WILEY-VCH WILEY

Chemical Engineering

PROFESSIONAL SCIENCE SAMPLER

INCLUDING

4

Green Engineering

Christopher L. Kitchens[1] and Lindsay Soh[2]

[1] *Clemson University, Clemson, South Carolina, USA*
[2] *Chemical and Biomolecular Engineering Department, Lafayette College, Easton, Pennsylvania, USA*

4.1 Introduction: Green Engineering Misconceptions and Realizations

During your career you have likely come across a colleague or co-worker who held the opinion that any greener alternative to a current process, chemical, or product is 1) more expensive, 2) doesn't work as well, or 3) is only motivated by external regulation [1]. This misconception is a primary argument of those opposing green innovation, but highlights that any greener alternative will not be adopted unless it is economically justified and does not sacrifice function [2]. Examples of such innovation are broad reaching across many aspects of the chemical industry as evidenced by the Presidential Green Chemistry Challenge Award winners [3]. It must also be realized that 1) the economics must be evaluated over the entire life cycle and 2) it is always better to incorporate greener alternatives in the design stage, rather than having to retrofit existing processes. Often different design stages of chemical processes are conducted independently thus hindering their systemic evaluation and possibly squandering opportunities for improvement. For example, the choice of raw materials, solvents, or reaction pathway is done based on operating costs that maximize the economic differential between product and consumables, including energy. If this is done before a rigorous evaluation of the capital and operating costs associated with separations, waste generation and treatment, fugitive emissions, safety measures, and risk assessment, then it is quite possible that a greener and more economical alternative is overlooked.

In order to avoid this opportunity loss and not handicap a process design with initial design choices, it is imperative to incorporate both guiding green engineering design principles as well as quantifiable green chemistry metrics with conventional design tools in the initial design stages [4]. While the economic advantage of a greener alternative may not be evident in preliminary design, advantages may emerge later during risk assessment analysis. Risk assessment is a systematic process of evaluating the probability of any event with adverse implications and the potential loss associated with each event, including economic loss. A comprehensive risk assessment by itself requires significant effort, time and expense, thus it is prohibitive to conduct during the early stages of a process design. Understanding the potential implications of design choices on risk assessment is reasonable and if done correctly, can streamline the overall design process by avoiding scenarios that require significant risk assessment. This is the concept of inherently safer design; Green Engineering Principle #1 [5].

Green engineering is defined by the U.S. Environmental Protection Agency as "the design, commercialization, and use of processes and products in a way that reduces pollution, promotes sustainability, and minimizes risk to human health and the environment without sacrificing economic viability and efficiency [6]." This is a working definition that has evolved to focus on a risk aversion viewpoint that weighs both human health

Green Techniques for Organic Synthesis and Medicinal Chemistry, Second Edition. Edited by Wei Zhang and Berkeley W. Cue.

and the environment [7]. The definition also places emphasis on sustainability from an economic standpoint, which reinforces the concept that any green engineering alternative must also be economical.

4.2 12 Principles of Green Engineering

Complementing the seminal publication by Anastas and Warner that defined the 12 Principles of Green Chemistry [8]. Anastas and Zimmerman later published the 12 Principles of Green Engineering, which are presented in abbreviated form [5]. Each of these benchmark principles focuses on different life-cycle stages of the engineering design or improvement process spanning from raw material and energy sources to end of life; systems design to process intensification. In all, the 12 Principles can be categorized into reducing hazard, increasing efficiency, minimizing excess, and increasing sustainability.

Many of the Principles are more straight forward to understand, others are less obvious, such as (3) design for separations, (5) output pulled, (6) reduce complexity, (9) minimize material diversity, and (8) avoid excess capacity. Typically, 70% of the materials and energy in a process are dedicated to separations, and generally the separation is designed based on the upstream process. A *design for separations* approach seeks to enhance efficiency by crafting the upstream such that the downstream separation requires less energy and materials. The end result may require increased upstream cost (e.g., in a reactor system), but the reduced separations cost will result in a decreased net cost. The concept of *"output pulled"* is derived from Le Chatelier's Principle, where a process should be developed in the direction which exploits equilibrium. In other words, a process will proceed more efficiently along a minimization of the Gibbs free energy, or it is easier to paddle down a river with the current versus against the current. *Reducing complexity*, *minimizing material diversity, and reducing capacity* can each be viewed as steps toward risk reduction. In such cases, increased complexity leads to higher probability that an unforeseen adverse event can occur; increased material diversity leads to potential chemical incompatibilities or reduced ability for material reuse; and excess capacity leads to greater potential for accidental spills or releases.

The 12 Principles of Green Engineering

1) Strive for inherently nonhazardous materials and energy inputs.
2) Prevent waste rather than treat.
3) Design separations and purifications to minimize energy and materials consumption.
4) Maximize mass, energy, space, and time efficiency.
5) "Output Pulled" rather than "Input Pushed".
6) Reduce complexity.
7) Design for durability, not immortality.
8) Avoid unnecessary capacity or capability.
9) Minimize material diversity.
10) Integration of energy and material flows.
11) Design for commercial afterlife.
12) Renewable rather than depleting.

The 12 Principles of Green Engineering serve as guidelines by which to develop a mindset conducive toward inherently greener products and processes. While these Principles inherently address several conventional design goals – maximizing profits, increasing energy efficiency, and use of safeguards to manage hazards – adherence to these Principles can provide additional guidance toward greener, safer, and more sustainable products or processes.

4.3 Green Chemistry Metrics Applied to Engineering

A practicing chemical engineer may say: "In my plant we strive to maximize efficiency, recover and recycle solvents and reactants, recover waste heat, use catalysts, and minimize toxic releases to the environment. Are we not Green Engineers?" Reply: YES. These are aspects of Green Engineering that are driven by economics. Now ask yourself: What happens at the product's end of life? Where are the raw materials sourced from? What happens when an accidental exposure or release does occur? Can we do better? YES. How can we do better? In order to answer this question, we need metrics to be able to perform quantitative evaluation of existing processes or products and new alternatives.

A metric is a quantifiable standard of measure that can be used for comparison. Green chemistry metrics can be used to determine if one product or process is "greener" than another (Comparative) or identify aspects of a product or process that can be improved (Improvement). Many green chemistry metrics exist, as discussed in previous chapters; the challenge is to choose the most appropriate.

The types of metrics that can be implemented span from very simplistic to extremely complex. Simple metrics are useful for rapid, transparent, and direct comparisons of similar scenarios, which can be used to easily display advantages of one reaction, process, or product over another. These green chemistry metrics include atom economy, E-factor, effective mass yield, reaction mass efficiency, environmental index, and so on. Conventional metrics that are on the same level of complexity include reaction yield, which is a product of reaction conversion and selectivity or cost index. Cost index (CI) is a baseline economic analysis that compares the cost of products to reactants: $\mathrm{CI} = \sum -\nu * \textit{Cost per pound or mole}$; where ν is the mass or mole stoichiometric factor. Similar to the Green Chemistry metrics, the cost index can be very useful in an initial cursory comparison of two very similar scenarios but neglect significant details as a trade-off for ease of implementation. As an example, Environmental Index (EI) can be used is a similar manor to CI where $\mathrm{EI} = \sum |\nu| * \textit{Toxicity Factor}$; where *Toxicity Factor* can be defined as the inverse of a conventional toxicity measures such as a threshold limit value (TLV), permissible exposure limit (PEL), or recommended exposure limit (REL). For EI comparisons, a smaller number is preferred and thus the reciprocal of the toxicity measure is used. Furthermore, for more toxic compounds, there can be little difference in the numerical value but the reciprocal will have greater difference. The inverse is true for less toxic compounds. Also, the absolute value of the stoichiometric factor is used because unlike the CI, toxicity can result from exposure to reactants or products and they are always present.

Traditional Tools	Green Chemistry Tools
• Level 1 – Yield and Selectivity – Row Materials Cost – Energy Requirements – Product Price – Waste Treatment	• Level 1 – E-Factor – Atom Economy – Effective Mass Yield – Reaction Mass Efficiency – Environmental Index
• Level 2 – Process Optimization – Process Integration	• Level 2 – Routes to Exposure – Hazard Assessment
• Level 3 – Process Design – Risk Analysis	• Level 3 – Risk Analysis – Life Cycle Assessment – Sustainability Analysis

A second level of Green Chemistry metrics contains an increase in complexity as well as depth of evaluation. These include a higher level of specific hazard assessment and potential routes of exposure. Specific hazard

analysis can include methods such as What If Analysis (WIA), Fault Tree Analysis (FTA), or Failure Mode and Effect Analysis (FMEA). FTA is a deductive, top-down method while FMEA is an inductive, bottom-up method. Regardless of the method, the result is identification of individual potential routes of failure with a certain probability of resulting in an adverse effect. The magnitude of the adverse effect for chemical hazards is a function of the toxicity, route to exposure, and exposure concentration. Exposure concentration is determined by a balance of time dependent release rate, transport, and degradation rate. Compilations of these metrics are a component of a full risk assessment, which is categorized as a Level 3 metric.

4.3.1 Maleic Anhydride Production Example

Maleic anhydride (MA) is classified as a commodity chemical with global production in excess of 1.5 million tons per year [1, 9]. It is primarily used in the manufacture of polyester resins for commercial applications in construction, marine, automotive, textiles, and consumer products applications. It is also widely used as a chemical precursor in the agricultural and chemicals industries, as well as in the food, consumer goods, and personal care products industries.

MA was first produced industrially in the 1930s by vapor-phase catalytic oxidation of benzene over a vanadium oxide and molybdenum oxide catalyst. Due to the ease of oxidation and high selectivity for MA, the benzene process dominated global production until the early 1980s. Rapidly increasing costs and growing environmental concerns associated with benzene use motivated a transition to an alternate feedstock. Furthermore, benzene conversion is inherently inefficient due to the conversion of a C_6 feedstock to a C_4 product (lower atom economy). This resulted in the exploration of C_4 feedstocks, and a process to produce MA from low-cost n-butane. Monsanto built the first butane to MA plant in 1974 and by the mid-1980s, 100% of U.S. production was from butane. Despite having lower selectivity, the butane process was preferred due to the lower feedstock cost, decreased environmental impact, and increased atom economy. The increased atom economy results in a competitive mass yield for butane with the lower selectivity. Advancements in catalysts for the butane reaction increased activity to enable conversion of benzene-based plants without capacity loss. Further advancements led to more energy efficient, solvent-based isolation and purification methods [9].

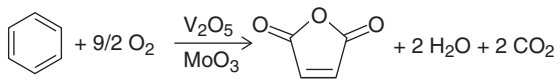
4.3.2 Level 1 Green Chemistry Metrics

Figure 4.1 shows the two chemical reactions for converting benzene and n-butane to MA. Both reactions are catalytic oxidation reactions with a single reaction step that can be evaluated using Level 1 green chemistry metrics. The benzene reaction has an E-factor of 1.26, atom economy of 0.44 and EI of 5.6 based on the TLV values. In contrast, the n-butane reaction has an E-factor of 0.73, atom economy of 0.58, and EI of 4.0. Based on these metrics, the n-butane process is preferred, but stoichiometric factor, conversion, selectivity, and yield are not accounted for.

Quinone is a by-product of the benzene oxidation reaction and to avoid quinone production, the reaction is run with excess oxygen at a 13.5:1 ratio rather than the stoichiometric 4.5:1 ratio. This excess results in lower MA yield due to CO_2 formation but reduces quinone and results in a stoichiometric factor of 2.3 and a yield of 62%. The n-butane route has a decreased MA selectivity of 55% and n-butane conversion of 85%, which results in a yield of 47%, but the use of excess reagents is not needed. The reaction mass efficiency (RME) metric can be used to compare these two routes in order to take into account, not only yield, but also excesses and other materials consumption. In this example the n-butane route (RME 0.270, assuming stoichiometric feeds and material recycle) has a more favorable RME than the benzene route (RME 0.122) despite the lower reaction yield. While a more detailed metric, the RME is still limited in scope and does not account for many other factors that will influence the comparative analysis, especially scale-up, and purification.

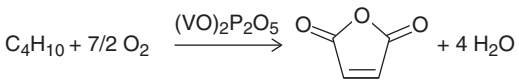
An alternate approach to the Level 1 metrics is to apply the same metrics to an industrial scale process, rather than chemical reactions and benchtop reaction yields, however, industrial scale data or simulations

benzene to maleic anhydride



	Benzene	Oxygen	Maleic Anhydride	Water	Carbon Dioxide
MW (g/mol)	78.1	32.0	98.1	18.0	44.0
Stoichiometric Mass Ratio	0.80	1.47	1.0	0.37	0.90
TLV (ppm)	0.5	–	0.25	–	5000
PEL (ppm)	1–5	–	0.25	–	–
REL (ppm)	0.1	–	0.25	–	–

n-butane to maleic anhydride



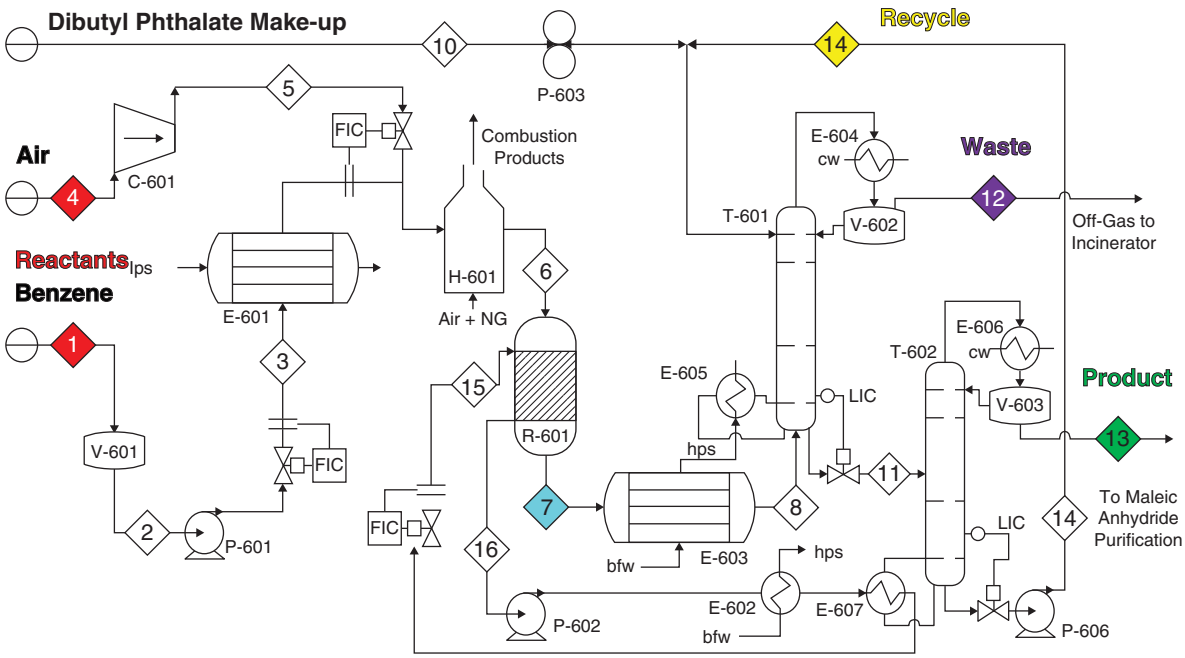
	n-butane	Oxygen	Maleic Anhydride	Water
MW (g/mol)	58.1	32.0	98.1	18.0
Stoichiometric Mass Ratio	0.59	1.14	1.0	0.73
TLV (ppm)	1000	–	0.25	–
PEL (ppm)	800	–	0.25	–
REL (ppm)	800	–	0.25	–

Figure 4.1 Comparison of the benzene and n-butane methods of MA synthesis with information needed for green chemistry metrics. Threshold Limit value (TLV), permissible exposure limit (PEL), and recommended exposure limit (REL) were obtained from safety data sheets.

may be difficult to obtain. Figure 4.2 shows an example process flow diagram (PFD) and process stream composition flows for an industrial scale benzene conversion to MA obtained from Turton *et al.* [1]. Benzene is mixed with excess air before being heated to 460°C and fed into a vapor-phase reactor where the exothermic catalytic oxidation reaction occurs. Excess air is used to suppress by-product formation and avoid the lower flammability limit. The reaction product is then sent to an absorber and separation tower that uses dibutyl phthalate to isolate the MA.

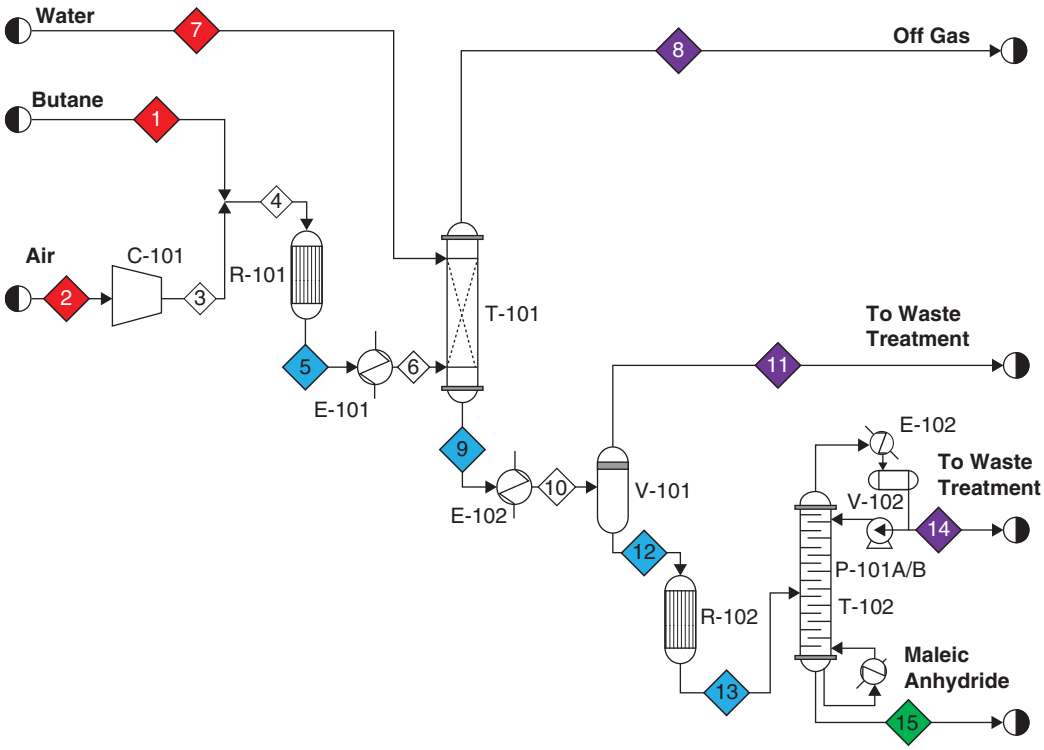
A mass-based E-factor of 31.3 can be determined by taking the ratio of waste (stream 12) to product (stream 13). Stream 12 includes the inert N_2 from the air stream which could be argued is not a waste material. Without the N_2 inert, the E-factor becomes 7.5. Further analysis of the PFD gives a MA yield of 62% around the primary reactor, an overall yield of 59%, and a stoichiometric factor of 2.34. Assuming a materials recovery parameter of 1, gives an RME of 0.114.

Figure 4.3 is a PFD for the n-butane process. An E-factor of 20.5 is calculated from the ratio of waste streams 8, 11, and 14 to product stream 15. Neglecting the N_2 inert gives and E-factor of 8.18. The MA yield around the primary reactor is 78.7%, the overall yield is 74.6%, and the stoichiometric factor is 1.97. Assuming a materials recovery parameter of 1, yields a RME of 0.218.



Stream No.	1	4	6	7	8	9	10	11	12	13	14
Temperature (°C)	30	30	460	608	260	329	320	194	84	189	329
Pressure (kPa)	101	101	235	220	215	80	100	82	75	70	80
Total (kmol/h)	42.3	2790	2832.3	2825.2	2825.3	500.1	0.1	526.2	2797.9	26.2	500
Total (kg/h)	**3304**	**80490**	83794	83794	83794	139191.6	30.6	141866	81225	**2597**	139269
Flowrates (kmol/h)											
Maleic Anhydride	–	–	0	**26.3**	26.3	–	–	24.8	0.5	**24.8**	–
Dibutyl Phthalate	–	–	0	–	–	500.5	0.1	500.4	0.1	–	500.4
Nitrogen	–	2205	2205	2205	2205	–	–	–	2205	–	–
Water	–	–	0	91.5	91.5	–	–	–	90.5	–	–
Oxygen	–	**585**	585	370.2	370.2	–	–	–	370.2	–	–
Benzene	**42.3**	–	42.3	**2.6**	2.6	–	–	–	2.6	–	–
Quinone	–	–	–	0.7	0.7	–	–	0.4	0.4	0.4	–
Carbon Dioxide	–	–	0	129	129	–	–	–	129	–	–
Maleic Acid	–	–	0	–	–	–	–	1	–	1	–

Figure 4.2 Process flow diagram for the benzene to maleic anhydride process with composition information for the chemical streams throughout the process. Data for this figure and the PFD were obtained from Turton *et al.* [1]. *Source*: Adapted from Turton, 2008.



Stream No.	1	2	5	7	8	9	11	12	13	14	15
Temp. (°C)	20	20	410	45	59.9	60	120	120	160	97.1	162
Press. (kPa)	275	101	275	170	170	170	101	101	101	101	101
Vapor Fraction	0	1	1	0	1	0	1	1	0.7	0	0
Flowrate (kg/h)	2000	41174	43174	11216	37621	16769	13640	3128.8	3128.8	602.5	2526.3
Flowrates (kmol/h)											
Nitrogen	–	1115	1115	–	1115	–	–	–	–	–	–
Carbon Monoxide	–	–	0.78	–	0.78	–	–	–	–	–	–
Oxygen	–	297	196	–	196	–	–	–	–	–	–
Carbon Dioxide	–	–	2.3	–	2.3	–	–	–	–	–	–
Butane	34.4	–	6.1	–	–	6.1	6.1	.002	.002	.002	.–
Water	–	25.2	138	623	–	138	734	0.7	27.6	27.3	0.25
Formic Acid	–	–	0.39	–	–	0.39	0.38	.004	.004	.004	.002
Acrylic Acid	–	–	0.39	–	–	0.39	0.37	0.02	0.02	.008	0.01
Maleic Anhydride	–	–	27.1	–	–	27.1	–	–	26.8	1.1	25.7
Maleic Acid	–	–	–	–	–	–	0.29	26.9	–	–	–

Figure 4.3 Process flow diagram for the n-butane to maleic anhydride process with composition information for the chemical streams throughout the process. This figure is adapted from Turton and Shaeiwitz [2]. *Source*: Adapted from Turton, 2008 and Shaeiwitz, 2017.

Table 4.1 Summary of Level 1 Green Chemistry metrics applied to the maleic anhydride reactions and process flow diagrams for the benzene and n-butane processes.

Metric	Benzene Process	Butane Process
Chemical Reaction		
E-factor (MW)	1.27	**0.734**
Atom Economy	0.454	**0.577**
Environmental Index	5.60	**4.00**
Yield	**0.62**	0.468
Reaction Mass Efficiency	0.122	**0.270**
Chemical Process (PFD)		
E-factor (mass)	**7.5**	8.18
Yield	0.62	**0.787**
Reaction Mass Efficiency	0.114	**0.281**
Environmental Index	5.46	4.0

Table 4.1 provides a summary of the Level 1 Green Chemistry metrics. When applied to the industrial scale PFD, the n-butane process has the advantage in yield, RME, and environmental index. The benzene process has a lower E-factor when N_2 is neglected, which results from the extensive water use in the purification stage. Potential for process improvement may include recycle of the water, as is done with the dibutyl phthalate in the benzene process.

Level 1 metrics can be very useful in a comparative analysis or identification of opportunities for improvement when complemented with conventional metrics that include energy consumption, and cost analysis. These metrics are also congruent with the Green Engineering Principles; specifically, Principles 1 through 4. Additional Level 1 metrics can also be envisioned which account for factors such as environmental persistence, bioaccumulation, bioconcetration, and so on.

4.3.3 Level 2 Green Chemistry Metrics

While conventional Level 2 design metrics may be seen as equivalent to process optimization or intensification, Green Chemistry metrics focus more on quantifying hazards and exposure in the occupational setting as well as the environment. The added complexity and diversity of contributing factors associated with the upper-level metrics often results in less quantitative tools and a greater proportion of qualitative assessment, which eventually boils down to cost and risk. Specific methods that can be used to conduct Level 2 analysis are well developed and broadly applicable techniques that are amenable to this level of evaluation.

RISK: In the context of a chemical plant, risk assessment can be evaluated as a cross product between hazard and exposure where the probability of an adverse event occurring and the severity of implications is a function of both the type of hazard and route or likelihood of exposure.

HAZARD: Hazards can exist in many forms as physical (e.g., pressure, temperature, velocity, electrical) or chemical. Within the chemical realm, hazards include toxicity, reactivity (corrosive, oxidizing, or explosive), flammability, or radioactivity. The hazard assessment essentially distills down to the different species present and the inherent hazards, which are a function of the physical properties (volatility, boiling point, water solubility, Henry's constant or octanol-water partition coefficient) or chemical properties (toxicity, reactivity, etc.), which can be measured or predicted. The above-mentioned hazards are what we can

consider as conventional measures; from the green engineering standpoint, we must also consider environmental measures such as aquatic toxicity, bioaccumulation, bioconcentration, or potential for ecosystem disruption (global warming, ozone deletion, smog formation, acid rain, or eutrophication).

Now, to put the complexity of the issue into perspective, the CAS registry currently has more than 120 million chemicals and materials known. The 2016 amendment to the 1976 Toxic Substances Control Act (TSCA) brought the number of inventoried compounds to more than 67,600 [10]. On the other hand, it is reasonable to say that there are probably around "only" 10,000 different chemicals commonly use in the chemicals industry. This is not outrageous but when you begin to consider the interactive properties of each of these chemicals – from reactivity to mixture properties – the problem of assessing hazards becomes significant or impossible if relying on tabulated recorded data. An alternative to tabulated data is the use of property estimation methods to predict physical properties as well as hazards associated with green engineering; organism toxicity, fate in the environment, bioconcentration, or bioaccumulation [4]. EPI Suite [11] is an example of a predictive tool provided by the U.S. EPA that uses chemical functional group or structure activity relationship models to predict physical properties as well as empirically derived models that correlate properties like Henry's constant or octanol-water partition with aquatic toxicity, bioaccumulation, biodegradation, environmental persistence, or environmental fate. For certain hazard associated properties, a certain quantifiable number is an adequate representation, but not always. Case in point relates to toxicity where an LC50 or recommended exposure limit can be measured or estimated, but this does not disclose the type of toxicity or information contained in a full dose-response curve. Ultimately, the many tools available to assess hazards can be useful for preliminary design, however, the limitations of each value should be considered.

EXPOSURE: The other half of the risk equation is exposure. Considerations include the route of exposure (inhalation, ingestion, and adsorption), duration (acute versus chronic), and magnitude or amount of the exposure within a period of time. Again, conventional measures can be used to assess exposure within the confines of a chemical plant by understanding the design around chemical storage and transportation, containment safeguards, ventilation, and waste treatment. Furthermore, there are tools available to assess and quantify fugitive emissions based on the plant design and operations [4].

Expanding beyond the conventional to encompass green engineering tools requires adapting fate and transport models into an environmental system. Consider the case of a large-scale accidental release of a known hazardous compound. Exposure assessment will require determination of the fate, which is dictated by the physical properties as well as transport and kinetics. For example, the volatility or Henry's Constant will dictate the relative partitioning between air and water, while the water solubility or octanol-water partition coefficient will dictate the partitioning between the water and soil. The octanol-water partition coefficient can also be predictive of bioaccumulation, plant uptake, or skin/gut/gill absorption.

Environmental transport is also a significant piece of the equation. The environment is a complex system with convective flows of air and water that will distribute the hazardous compound from the original site. Thus, the exposure concentration will be a function of the distance, time, environmental conditions, and partitioning between the air, water, soil, and silt. Degradation kinetics also determine a chemical's fate in the environment. Atmospheric persistence is governed by hydroxyl radical oxidation reactions. Similarly, aquatic degradation is controlled by hydrolysis reactions. In the simplest assessment, kinetic rate equations for different chemical compounds can be estimated in order to estimate the overall biodegradability in the form of an index or environmental half-life. Other methods are available to model the aerobic and anaerobic biodegradation of compounds [4]. Once assembled, it is possible to perform a continuum-based modeling approach to solve a collection of partial differential equations that account for transport and reaction kinetics in order to determine concentrations for any scenario.

Chemical fate and transport in the environment is a complex field of study and a vital component to green engineering [12]. For a more cursory evaluation, a tool such as the EPA's EPI Suite [11] is an excellent option

for comparative analysis of different compounds that may be considered for a given chemical synthesis or industrial process. For the case of MA synthesis, it is easy to compare n-butane versus benzene raw material or water versus dibutyl phthalate for the separation. However, another question may be to evaluate an alternative for MA all together or maybe even bioderived routes from renewable succinic acid.

4.3.4 Level 3 Green Chemistry Metrics

The third level of metrics are all-encompassing and extremely rigorous. These include life-cycle assessment (LCA), risk analysis, and sustainability analysis. These tools are equivalent in scope to a conventional full process design evaluation. In-depth discussion of these methods is beyond the scope of this chapter, however, there are many resources available for these topics. As an alternative, we will now focus on one aspect of the chemicals industry that can significantly impact the sustainability of a process: solvent use. We will illustrate how the Principles of Green Engineering can be used to aid in the design and overall sustainability of processes employing solvents across different life-cycle stages.

4.4 Use of Green Solvents in the Chemical Industry

Solvent applications range from product formulation to industrial manufacturing and processing. In consumer and commercial products alone (non-industrial) – the estimated amount of organic solvent usage is over 972,000 tons per year [13]. Including the solvent used for manufacturing across a product's life cycle dramatically increases this number. It has been estimated that solvents can typically account for 80% to 90% of mass utilization for batch processing of pharmaceuticals and fine chemicals [14]. While solvent use in continuous and larger scale bulk chemical production would be reduced, it still has considerable impact on green chemistry and engineering metrics as well as life-cycle impacts for a process [15].

Throughout the chemical processing industry, the use of solvents is commonplace in order to facilitate unit operations and attain product specifications. Solvents have traditionally played a key role in chemical processes for a number of reasons – they may lead to higher quality product, reduce by-product formation, enable product/catalyst separations, reduce the necessary number of process steps, and even be used as a reactant itself [16]. Frequently solvents are used in large excess of solutes and reactants in order to maximize yields. As such, they can greatly contribute to waste production and life-cycle impacts [17]. Calculated E-factors for the production of chemicals have shown that generally the mass of waste is 1 to 5 times the mass of chemical produced for bulk chemicals and, 5 to 50 times for fine chemicals and 25 to 100+ times for pharmaceuticals [15]. Of this waste it was found that upward of 80% may be attributable to solvents [17]. From the engineering perspective, more efficient process design can be implemented to minimize impacts by decreasing solvent requirements and/or maximizing recyclability.

In addition to efficient processing, green design of solvent applications must also consider solvent properties including physicochemical properties, which affect the functionality, acute hazard, and environmental implications associated with solvents. These properties include solubility, polarity, hydrophilicity, and volatility which will affect risk from both a hazard and exposure perspective. These hazards include flammability, explosion risk, and acute toxicity with exposure routes through workplace and consumer contact. These properties also impact environmental implications of solvents associated with fate and transport. As such these properties must be considered across a product life cycle to determine risks to environmental health including greenhouse gas, bioaccumulation, and ecotoxicity metrics [18]. In this context, the principles of green engineering provide guidelines for choosing better solvent alternatives and utilizing solvents more efficiently. The examples below illustrate how the principles can influence solvent design and utilization.

4.4.1 Waste Prevention

One of the best ways to reduce impacts associated with solvent use is to minimize the use of solvent. These reductions can include incorporating solvent-free approaches and/or increasing process efficiency to decrease

losses and maximize solvent recycling. Examples of solvent-free reactions in organic reactions, multiple component condensations, and heterocyclic syntheses using catalysts [19] and strategies such as, microwave and ultrasound irradiation [20], eutectic melt [21], and mechanochemistry [22]. It should be noted, however, that while no solvent is added to these systems, that reagents may be hydrated or form compounds such as acetic acid that could have a solvent effect [22]. While solvent-free options are expanding, the utility of solvents for improvements in yields, phase mediation, and separations often cannot be circumvented. Guided process design can aid in making these reactions and separations more efficient. For instance, the use of flow chemistry, microfluidics [23], and reactor design [24] may aid in reaction yields while minimizing ancillary solvent usage.

Given the unavoidable role of solvents for certain applications, judicious solvent choice that weighs hazard, exposure, recyclability, and life-cycle impacts is required. As such, the following principles consider how solvents as well as the systems/processes they are involved in can be designed to minimize their impacts. These principles guide us to choose solvent alternatives that are inherently nonhazardous, have minimal end-of-life impacts, and are renewably sourced. Furthermore, process specific principles address embedded complexity, designing for separation, and process integration for minimization of energy and materials. Examples of both conventional and neoteric solvents help to illustrate trade-offs across the solvent life cycle from sourcing, manufacturing, use, and end-of-life.

4.4.2 Inherently Non-Hazardous

While hundreds of organic solvents are available, water is an obvious choice as it is inherently benign with further advantages in that it is readily abundant and cheap. Aqueous-phase chemistries and extractions are well precedented with water being the choice solvent in biological systems [25]. The use of pure water may be limited by its highly polar behavior as well as its utility with heterogeneous catalysts [26]. These limitations may lead to the use of large volumes of water and/or utilization of surfactants to facilitate compatibility [27]. The recycling or disposal of water typically requires treatment for contaminants; these types of separations processes may be energy intensive particularly for large volumes and may require monitoring to ensure compliance. The use of surface-active agents may facilitate localized high concentrations of reagents within micelles, enabling reactions in an aqueous environment. While low concentrations are typically necessary, surfactant recovery may prove difficult [25]. As such life-cycle impacts of the production and disposal of surfactants must be considered for each application. However, designer surfactants, including bio-derived options, for micellar catalyst-enabled reactions is an growing field that could improve impacts associated with their use [28]. Developing chemistries in water also include the use of enzymes, nature's catalyst for selective and low-impact reactions. High-pressure/temperature and supercritical water has been explored as a powerful solvent for both extractions and reactions [27–29]. At ambient conditions, water acts as a solvent for ionic species, however, at and near supercritical conditions, the polarity shifts and water acts as a non-ionic species solvent. This range may be exploited for a number of different applications ranging from hydrolysis reactions to biomass extractions of non-polar, polar, and ionic compounds [30]. The high critical point of water at 647K and 22 MPa, however, requires significant energy contributes for heating and pressurization and must be considered from an energy use and cost (both capital and operating) perspective before adoption.

Despite advances in solvent-free and water-based solvent applications, organic solvent use is still prevalent and necessary for many reactions and separations processes. Over the last couple of decades, much attention has been paid to finding suitable, less hazardous solvent alternatives, particularly for the high solvent intensity industry of pharmaceuticals. As such, several solvent selection guides have been published through various companies including Sanofi [31], GlaxoSmithKline [32], (GSK) Pfizer [33], and AstraZeneca [34]. These guides use a variety of metrics to attempt to quantify the greenness of solvents for medicinal chemistry applications. The metrics are generally broken up into (process) safety, human health, and environmental categories though GSK has added categories for both waste and life-cycle assessment [35].

Table 4.2 summarizes the main parameters that are considered in the development of the Pfizer, Sanofi, and GSK guides. A common feature of all the guides is to utilize available environmental, health, and safety (EHS)

Table 4.2 Solvent properties to consider for EHS and life-cycle impacts.

	Safety	Human Health	Environment	Waste	Life Cycle
Pfizer	**Process Safety** Flammability (vapor pressure) Explosion risk Electrical Conductivity/Static Charge Peroxide formation potential	**Worker Safety** Carcinogenicity Mutagenicity Reprotoxicity Skin Absorption/ Sensitization Acute Toxicity	Ecotoxicity Water contamination Ozone Depletion Photoreactive Potential		
Sanofi	Flammability (flash & boiling point) Explosion risk (GHS) Autoignition temperature Electrical Conduct./ Static Charge Peroxide formation potential	Carcinogenicity Mutagenicity Reprotoxicity Skin Absorption/ Sensitization Acute Toxicity Boiling Point GSK	Energy demand (boiling point) Aquatic Ecotoxicity Bioaccumulation Ozone depletion		
GSK	**Flammability and Explosion Potential** Vapor Pressure Electrical Conductivity/ Static Charge Boiling Point Flash point Autoignition Temperature **Reactivity and Stability** Peroxide formation potential Explosion risk (NFPA reactivity rating) Self reaction potential Acidity/basicity	**Exposure** Occupational exposure limits **Hazard** GHS risk phrases Carcinogenicity Mutagenicity Reprotoxicity	**Aqueous** Acute and chronic ecotoxicity (GHS) Biodegradation **Air** Photochem O_3 creation potential Vapor Pressure	**Waste score:** Incineration (water solubility, emissions, and enthalpy of combustion) Recycling (ready drying/separation from water, safety aspects) Biotreatment (BOD, VOC potential, K_{ow}) VOC emissions	**LCA score:** Net mass of materials consumed Gross Energy usage Total water consumption Total organic carbon Fossil fuel feedstocks POCP equivalents Greenhouse gas emissions Acidification Eutrophication

data from a variety of sources. For example, many use the Globally Harmonized System (GHS) for health and safety data as a rich source of information for many chemicals including solvents providing EHS data for toxicity and flammability/explosion risks in their safety scores [36]. Complete information, however, may be lacking particularly for newer solvents. Furthermore, information may also be inconsistent between sources and must be verified [37]. Note that while all the guides use many mutual EHS metrics, other metrics, particularly in the environment category are not considered or weighed equally among different rankings. Furthermore, metrics such as boiling point, volatility, and water solubility are physicochemical properties that generally relate to the exposure rather than inherent hazard of the solvent; both of which should be

considered for full risk assessment. In utilizing these properties within multiple metrics, their impact becomes incorporated in various metrics and could compound or confound their importance. For instance, in the case of boiling point/vapor pressure, high boiling point solvents have lower risk in terms of air emissions and exposure but the commensurate low vapor pressure means that more energy would be required for distillation for reuse [31]. Further comparisons of these solvent guides can be found published elsewhere [18, 35, 38].

The importance of the chemicals industry for sustainability is reflected in REACH, recent regulation in the EU that was "adopted to improve the protection of human health and the environment from the risks that can be posed by chemicals, while enhancing the competitiveness of the EU chemicals industry [39]." Motivated by REACH, AstraZeneca has recently formulated a guide that includes 272 different chemicals [34]. The framework surveys physical parameters (e.g., boiling point, density, refractive index), solubility parameters, chemical properties (e.g., flash point, water solubility) of the solvents, though comprehensive data for all solvents is not available. Their works seeks to provide a prediction tool for determining best alternatives using principal component analysis and potentially providing unconventional solvent solutions. While these guides provide a number of alternatives to current solvents, there are not always drop-in replacements that are readily available. For example, the AstraZeneca guide highlights the lack of ready alternatives for dipolar aprotic solvents such as N,N-dimethylformamide (DMF), N,N-dimethylacetamide (DMA) [34]. Furthermore, these alternatives typically still have significant life-cycle impacts and as such only provide a step toward sustainable processing. As such neoteric alternatives are being developed that are high functioning but also minimize impacts according to the principles of green engineering. Ranging from bio-based solvents to ionic liquids and carbon dioxide, the hazards associated with these neoteric alternatives must still be carefully considered across the process life cycle.

4.4.3 Renewable Rather Than Depleting

Biomass-derived solvents stem from a variety of feedstocks from lignocellulosics to oleaginous materials [40]. As such they may span a large range of necessary solvent properties [35, 41]. Glycerol is a particularly interesting feedstock as it is a co-product of biodiesel production that has low value and has even been ascribed as waste. The valorization of glycerol as a solvent and/or derivatization to produce other solvents or chemicals adds economic incentive and can decrease environmental impacts associated with the production of biodiesel [42]. While there is much potential for renewably based solvents, not all bio-derived solvents are in fact non-hazardous [40]. EHS metrics should be evaluated to make sure that these solvents are inherently benign. Furthermore, life-cycle metrics can be used to analyze their potential impacts including all stages such as production and end of life. Several studies have looked at the potential impacts of some of these bio-based solvents indicating that like petroleum-derived solvents, they also have the potential to have significant toxicity and are potentially not biodegradable. For example, in the production of hexamethylenediamine, several routes were evaluated via life-cycle assessment in order to compare bio-based versus fossil-based methods for manufacturing. The study found the fossil-based route to be more economical and with generally lower life-cycle impacts than the bio-based alternatives [43]. The study found trade-offs for certain alternatives between life-cycle metrics particularly related with eutrophication impacts due to the use of fertilizers. This trade-off is often associated with bio-based products and could be potentially mitigated by more effective nutrient application or more complete bio-based feedstock utilization allowing for a spread allocation of impacts across a number of products [44].

In addition to conventional techniques, advancements in computational strategies have recently been used for new solvent development. *In silico* design strategies can be used to generate feasible molecular candidates from renewably based building blocks and predict their functional properties. This powerful technique can reduce time and waste during solvent development including selecting from a potentially large spectrum of candidates. These method may be particularly valuable for certain solvent categories (e.g., polar aprotic, asymmetric halogenated hydrocarbons) where no or few good alternatives exist [45]. For example, the dissolution of nitrocellulose was evaluated for potential green solvent alternatives to conventional solvents like

butyl acetate, xylene, and toluene, all of which are flammable and pose other safety risks. Using this *in silico* approach, diacetin, a glycerol-derived compound, was isolated as a strong candidate with low EHS impacts. Computation approaches can certainly aid in the effort to establish solvent alternatives. Unfortunately, for many of these compounds, only empirical data for all parameters is available and thus estimates are used for physicochemical properties and life-cycle impacts – development of this type of data would enable more accurate and models and better alternatives to be established.

4.4.4 Design for Commercial After-Life

The fate of solvents after their direct use may have significant weight on their life-cycle impacts and must be evaluated with regards to a number of metrics (Table 4.2). While dozens of possible metrics could be used (from eutrophication to ozone depletion or emissions potentials) certain key metrics are distinctly important given specific applications. For example, the feedstock/materials extraction phase is particularly important for bio-derived products since the use of fertilizers and water can lead to high eutrophication potential and water use.

To some extent, the environmental fate and transport of solvents can be estimated via physicochemical properties of the compounds themselves [46]. High volatility/low boiling point solvents are more likely to partition to air. Partitioning can also be determined by such correlations as Henry's law or the octanol-water partition coefficient which may be used to estimate water/air partitioning, bioaccumulation, and to some extent soil sorption. The compartmentalization of these compounds will also determine the potential exposures for such end-point impacts as aquatic ecotoxicity and biodegradability.

Despite the importance of determining end-of-life impacts, taking all of these joint factors into account is difficult and the weighting between each of the impacts elucidates trade-offs that should be considered during solvent selection. As such, environmental impacts are the most varied set of parameters evaluated among the solvent selection guides (Table 4.2). All account for ecotoxicity and ozone depletion by some means, though the accounting for other metrics vary. In the case of Sanofi, the major environmental assessment is done via boiling point correlation where high boiling point components are penalized as they require high energy for recycle. This phenomenon is also noted in the GSK guide [32]. This approach, while limited also starts to address some of the life-cycle impacts that would need to be addressed for solvent reuse. Previous researchers have recommended the avoidance of distillation as a separation technique in order to cut down on life-cycle impacts associated with the high energy use [46]. The GSK guide is the only one that extensively accounts for life-cycle impacts in its guide [32].

4.4.5 Separation and Purification to Minimize Energy Consumption and Materials Use

Separations are particularly energy intensive operations. It has been estimated that separations processes can consume up to 40% of energy use in chemicals manufacturing and may contribute to more than 50% of operating costs [47]. Distillation is the most prevalent industrial separation process and represents a large portion of process energy. As such, in the search for green solvents, one criteria often considered is the vapor pressure/boiling point [46]. A low boiling point requires less energy for solvent distillation, however, the commensurate high-vapor pressure has potential ramifications for inhalation exposure and atmospheric life-cycle endpoints. Current focus on novel means for separation include membranes, crystallization, and chromatography, however, these processes still require significant improvements in order to decrease energy consumption and make them cost-competitive to distillation.

Switchable solvents are a promising solvent alternative that are designed for ready separation. These solvents contain a property that is easily flipped given an external trigger. For example, some switchable solvents can flip polarity or hydrophilicity using CO_2 as the switch [48]. The change in property can be useful for both reactions (drawing products into separate phases) and extractions. For example, a switchable hydrophilicity solvent was used for soybean oil extraction. Typically, this type of oil extraction is done with hexane which has

toxicological consequences and requires distillation. Using a switchable solvent in the hydrophilic form, oil is first extracted from soy bean flakes. Water is then added to separate the solvent and oil. In order to recover the solvent, CO_2 is then sparged through the water/solvent mixture to switch the solvent to the hydrophobic form and recover the solvent which can then be switched back to its original hydrophilic form [49]. While these applications are promising, the use of SHS is not currently commercially represented, however, design of these solvents for particular applications with further improvements in recovery and yields could aid in the adoption of these solvents for alternative separations [50].

From a separations perspective, another interesting green solvent candidate is high-pressure carbon dioxide either in liquid or supercritical states. CO_2 has been used for extractions, reaction mediation, and chromatography. One major advantage is that its properties are potentially tunable with varying pressure and temperature as well as the use of co-solvents [51]. The use of pure CO_2 for extractions is growing in popularity for food applications as this solvent is completely non-toxic and can easily be removed by bringing the pressure down to ambient conditions [52]. The use of co-solvents such as methanol or ethanol extends the solubility range of CO_2. Adding a secondary solvent does potentially require downstream separations, however, these co-solvents are used in much smaller quantities than conventional solvent operations thus decreasing separations burdens. Despite the promise and prevalence of research applications using CO_2 few have come to commercial fruition. Common reasons for this include risk aversion due to high capital costs and operating costs associated with the energy input needed for CO_2 compression during pressure swings needed for product recovery. However, recent studies have shown that pressurization is not the most energy-intensive step for CO_2 extraction (chilling and heating also play a significant role) but that total energy is comparable to that of distillation processes [53]. New methods that may improve the energy use of CO_2 extractions include improving recyclability without large changes in pressure (e.g., using membranes or adsorption) and process integration.

4.4.6 Integration and Interconnectivity with Available Energy and Materials Flows

Throughout a process, solvents are integrated into numerous unit operations where the integration of energy and materials flows would enable gains in efficiency. In the previous example, using supercritical CO_2, heat integration could significantly reduce the energy needed for heating and chilling of the supplied CO_2. Since the critical temperature of CO_2 is 304K, operating temperatures could be supplied by waste heat. Integrated process flows could also be used for switchable solvents where waste CO_2 produced from various processes could be used in a switchable solvent system [48]. Another notable example of process integration is reactive distillation where products are being separated as they are formed, creating a better driving force for better yields and higher purities [54].

The possibility of mass/energy integration with respect to both conventional and neoteric solvents could be evaluated circumstantially. For instance, the use of solvents in an integrated biorefinery would ideally produce a number of different products while utilizing selective solvents that could produce a variety of products [35]. Specifically, during the production of biodiesel the process includes a reactor and several distillation columns to remove product from by-product (glycerol) and excess methanol for recycle. It was shown that by integrating heat exchangers and water flows from distillation columns and reactors in various biodiesel production scenarios, that the energy and water consumption was significantly decreased for production [55]. This example provides insights into how appropriate process design could aid in the more sustainable use and recycling of system components including solvents.

4.4.7 Conserve Complexity

Minimizing the complexity of a process from the original design could help to circumvent unnecessary losses. As such decreasing the complexity of components used as well as the number of process steps required could aid to minimize process impacts. From the perspective of solvents this may refer to the number of steps needed

Table 4.3 Life-cycle considerations for green engineering with solvents.

Raw materials sourcing	Production	Utilization	Recycling/ Disposal
Fossil versus biobased	Synthesis steps	Exposure to workers and consumers	Distillation/separations
Renewable versus depleting	Intermediate exposures	Functionality	Disposal method
Accessibility	Energy and resource consumption	Separations	End-of-life impacts (biodeg., ecotox., etc.)

to produce the solvent itself or those needed for the use/re-use of the solvent. Ionic liquids (IL) serve as a prime example of embedded complexity in the production of solvents. Often considered green solvents due to their low vapor pressure (minimizing inhalation exposure and facilitating product distillation), the actual impacts of IL are sometimes called to question due to their potentially high toxicity [56] and large number of synthesis steps [46, 57].

With regards to ionic liquids, the evaluation of these potential solvents touches many aspects of green engineering, highlighting many of the previously mentioned principles. A life-cycle analysis comparing the use of IL ([Bmim][BF_4]) to conventional solvents for two reactions showed that conventional methods had better or comparable life-cycle impacts to the IL [57]. While these results indicated that the evaluated IL was not the most attractive alternative, it also highlights where improvements to IL separation efficiency and recyclability would improve the life-cycle outcome. As such, recent attention has been paid to the production of lower impact IL. For example, renewable/readily available materials have been used for production of IL [58] using low-impact syntheses. Furthermore, many IL have traditionally shown high human and ecotoxicity; however, several new IL are being developed from biological materials enabling their use for food applications and reducing the hazard from exposure to humans or environment [59]. Highly recyclable IL with low cost are also being developed [60]. The recyclability has a strong impact on the cost; for example, a technoeconomic analysis showed that even for a high recycle rate of 99.6%, IL costs are estimated at 39% of the minimum ethanol selling price [61]. Other advances in the field of IL include developing tunable formulations depending on the anion and cation identity. Tunability could lead to better selectivity and efficiency, thus allowing for lower impacts [62].

As the previous examples have highlighted, there is much opportunity in the field of solvents to find and implement greener alternatives. These solvents must be evaluated across the entire life cycle from sourcing to disposal of the solvent in terms of an array of indicators that can be categorized into environmental, efficiency, economic, and energy endpoints [63]. Table 4.3 highlights factors to be considered at each life-cycle stage with respect to the design and use of green solvents. As a consequence of innovations in this field, there have been several Presidential Green Chemistry Awards that have been awarded to applications specific to solvents from water to CO_2 and will be discussed below.

4.5 Presidential Green Chemistry Awards

To find examples of 12 Principles of Green Chemistry and Green Engineering being employed in research and on an industrially relevant scale, one most look no further than the winners of the Presidential Green Chemistry Challenge Awards, established in 1996. As of 2016, it is estimated that the award-winning technologies have eliminated 826 million pounds of hazardous chemicals each year, saved 21 billion gallons of water each year, and eliminated 7.8 billion pounds of CO_2 equivalents each year [3]. In each award-winning technology, the 12 Principles of Green Chemistry and Green Engineering are evident, green chemistry metrics are used to demonstrate the magnitude of the advantages, and undoubtedly, every adopted technology was

also economically beneficial. Here are a few examples of award-winning technology focus on green solvent usage.

In 2011, Bruce Lipshutz received the academic award for the use of water as a solvent using surfactants in order to allow for chemical reactions in water and circumventing the need for production of organic solvents. While the surfactants still need to be produced, the quantities are much smaller than those of organic solvents.

Eastman Chemical won the Greener Synthetic Pathways award in 2009 for creating a solvent-free enzymatic production of esters for their cosmetic and personal care products. Purportedly the process can save over 10 liters of organic solvent per kilogram of product and has benefits in terms of product purity, cost, and life-cycle impacts.

The 2006 Small Business award was given to Arkon Consultants and NuPro Technologies, Inc., for their work on the development of a system that reduces solvent use and replaces solvents with greener alternatives. As a result, the system reduces explosion potential and emissions during solvent recycling. Typically, in flexographic printing, a mixture of chloro-saturated cyclic, or acyclic hydrocarbons (such as xylene) are used for washing of their printing plates. Firstly, new solvents were developed using bio-based solvents which also improved EHS properties (flash point, toxicity, explosion potential, workplace exposure) for their solvents. They also developed a recycling system using filtration and centrifugation to lower workplace exposure to the solvents as well as improve recycling.

The 2004 Academic award was given to Professors Charles Eckert and Charlie Liotta for their work combining reactions and separations in tunable solvent systems. Using both CO_2 and water, they have provided a body of work that takes advantage of the differential properties of high pressure systems, tuning the conditions for optimal reaction and separations conditions including the use of phase-transfer catalysts. These techniques also allow for improved catalyst recycling and have been used for a variety of applications in the chemical and pharmaceuticals industries.

The 1998 Green Reaction Conditions award was given to Argonne National Laboratory for their synthesis of lactate esters, sugar-derived solvents, for the replacement of more hazardous solvents such as halogenated constituents. The award is for their membrane reaction/separation process that uses pervaporation membranes and catalysts for the liquid and vapor product separations that allows for ammonia separation and utilization from the other reaction constituents acids/esters/alcohols. The process was able to decrease waste production while making the production of this renewable solvent more economical.

4.6 Opportunities and Outlook

In closing, the opportunities for green engineering innovations in the chemicals industry are many and the outlook is bright. There are numerous metrics-based tools for evaluating how green a process or product is, and in many cases the difficulty is determining the best set of metrics and evaluating the entire picture. There is significant opportunity for innovation in this field, particularly as our natural resources continue to be depleted, global population is growing, societies are advancing, and the global environmental burden increases. Green engineering is a key aspect of sustainability and must be at the forefront of our minds as we design new process, build new plants, and revolutionize our industry.

References

1 Turton, R., Bailie, R. C., Whiting, W. B., Shaeiwitz, J. A. (2008) *Analysis, Synthesis and Design of Chemical Processes*. Pearson Education.

2 Shaeiwitz, J. A., Turton, R. (2017) Production of maleic anhydride. In *Large Scale Process Design*, Department of Chemical Engineering, W. V. U., Ed. http://www2.cemr.wvu.edu/~wwwche/publications/projects/large_proj/maleic.PDF (15 June 2017).

3 American Chemical Society Presidential Green Chemistry Challenge Awards. (n.d.) https://www.epa.gov/greenchemistry (15 June 2017).
4 Allen, D. T., Shonnard, D. R. (2001) *Green Engineering: Environmentally Conscious Design of Chemical Processes*. Pearson Education.
5 Anastas, P. T., Zimmerman, J. B. (2003) Design through the 12 principles of green engineering. *Environ. Sci. Technol.*, **37** (5), 94A–101A.
6 Environmental Protection Agency. (n.d.) https://www.epa.gov/green-engineering/about-green-engineering#definition. (accessed December **12**, 2016).
7 (a) Abraham, M. A., Nguyen, N. (2003) "Green engineering: Defining the principles" — results from the sandestin conference. *Environmental Progress*, **22** (4), 233–236; (b) Winterton, N. (2001) Twelve more green chemistry principles. *Green Chemistry*, **3** (6), G73–G75.
8 Anastas, P. T., Warner, J. C. (1998) *Green Chemistry: Theory and Practice*. Oxford University Press: New York, p. 30.
9 Felthouse, T. R., Burnett, J. C., Horrell, B., Mummey, M. J., Kuo, Y.-J. (2000) Maleic anhydride, maleic acid, and fumaric acid. In *Kirk-Othmer Encyclopedia of Chemical Technology*, John Wiley & Sons, Inc.
10 U.S. Congress. (2016) Frank R. Lautenberg Chemical Safety for the 21st Century Act.
11 EPA, U. S. (2017) *Estimation Programs Interface Suite™ for Microsoft® Windows, v 4.11*, Washington, DC, USA.
12 Hemond, H. F., Fechner, E. J. (2015) *Chemical Fate and Transport in the Environment* Third Edition ed.; Academic Press: Boston, pp. 1–73.
13 EPA. (1996) *Environmental Protection Agency: Consumer and Commercial Solvent Use (Final Report)*.
14 Constable, D. J. C., Jimenez-Gonzalez, C., Henderson, R. K. (2007) Perspective on solvent use in the pharmaceutical industry. *Org. Process. Res. Dev.*, **11**.
15 Sheldon, R. A. (2007) The E factor: Fifteen years on. *Green Chemistry*, **9** (12), 1273–1283.
16 Welton, T. (2015) Solvents and sustainable chemistry. *Proc R Soc A*, **471**.
17 Jiménez-González, C., Constable, D. J., Ponder, C. S. (2012) Evaluating the "greenness" of chemical processes and products in the pharmaceutical industry—A green metrics primer. *Chemical Society Reviews*, **41** (4), 1485–1498.
18 Tobiszewski, M., Tsakovski, S., Simeonov, V., Namieśnik, J., Pena-Pereira, F. (2015) A solvent selection guide based on chemometrics and multicriteria decision analysis. *Green Chemistry*, **17** (10), 4773–4785.
19 Singh, M. S., Chowdhury, S. (2012) Recent developments in solvent-free multicomponent reactions: a perfect synergy for eco-compatible organic synthesis. *RSC Advances*, **2** (11), 4547–4592.
20 Martins, M. A., Frizzo, C. P., Moreira, D. N., Buriol, L., Machado, P. (2009) Solvent-free heterocyclic synthesis. *Chem. Rev.*, **109** (9), 4140–4182.
21 Cave, G. W., Raston, C. L., Scott, J. L. (2001) Recent advances in solventless organic reactions: towards benign synthesis with remarkable versatility. *Chemical Communications*, **21**, 2159–2169.
22 James, S. L., Adams, C. J., Bolm, C., Braga, D., Collier, P., Friščić, T., Grepioni, F., Harris, K. D. M., Hyett, G., Jones, W., Krebs, A., Mack, J., Maini, L., Orpen, A. G., Parkin, I. P., Shearouse, W. C., Steed, J. W., Waddell, D. C. (2012) Mechanochemistry: opportunities for new and cleaner synthesis. *Chem Soc Rev.*, **41**.
23 (a) Irfan, M., Glasnov, T. N., Kappe, C. O. (2011) heterogeneous catalytic hydrogenation reactions in continuous-flow reactors. *ChemSusChem*, **4** (3), 300–316; (b) Razzaq, T., Glasnov, T. N., Kappe, C. O. (2009) Continuous-flow microreactor chemistry under high-temperature/pressure conditions. *European Journal of Organic Chemistry*, 2009 (**9**), 1321–1325.
24 Sheldon, R. A. (2012) Fundamentals of green chemistry: efficiency in reaction design. *Chemical Society Reviews*, **41** (4), 1437–1451.
25 Lipshutz, B. H. Gallou, F. Handa, S. (2016) The evolution of solvents in organic chemistry. *ACS Sustainable Chemistry & Engineering*.

26 Lipshutz, B. H. Ghorai, S. (2009) Transition metal catalyzed cross-couplings going green-in water at room temperature. *ChemInform*, **40** (41), i.

27 Kruse, A. Dahmen, N. (2015) Water–A magic solvent for biomass conversion. *The Journal of Supercritical Fluids*, **96**, 36–45.

28 Foley, P. Kermanshahipour, A. Beach, E. S. Zimmerman, J. B. (2012) Derivation and synthesis of renewable surfactants. *Chemical Society Reviews*, **41** (4), 1499–1518.

29 Brunner, G. (2009) Near critical and supercritical water. Part I. Hydrolytic and hydrothermal processes. *The Journal of Supercritical Fluids*, **47** (3), 373–381.

30 Weingärtner, H. Franck, E. U. (2005) Supercritical water as a solvent. *Angewandte Chemie International Editio*n, **44** (18), 2672–2692.

31 Prat, D. Wells, A. Hayler, J. Sneddon, H. McElroy, C. R. Abou-Shehada, S. Dunn, P. J. (2016) CHEM21 selection guide of classical- and less classical-solvents. *Green Chem*, **18**.

32 Alder, C. M. Hayler, J. D. Henderson, R. K. Redman, A. M. Shukla, L. Shuster, L. E. Sneddon, H. F. (2016) Updating and further expanding GSK's solvent sustainability guide. *Green Chem*.

33 Kim Alfonsi, K., Colberg, J., Dunn, P. J., Fevig, T., Jennings, S., Johnson, T. A., Kleine, H. P., Knight, C., Nagy, M. A., Perry, D. A., Stefaniak, M. (2008) Green chemistry tools to influence a medicinal chemistry and research chemistry based organisation. *Green Chem*, **10**.

34 Diorazio, L. J., Hose, D. R. J., Adlington, N. K. (2016) Toward a more holistic framework for solvent selection. *Org Process Res Dev*, **20**.

35 Soh, L., Eckelman, M. J. (2016) *Green Solvents in Biomass Processing*.New York: Pearson.

36 Secretariat, U. N. E. C. f. E. (2009) *Globally Harmonized System of Classification and Labelling of Chemicals (GHS)*. United Nations Publications.

37 Prat, D., Hayler, J., Wells, A. (2014) A survey of solvent selection guides. *Green Chem*, **16**.

38 Byrne, F. P., Jin, S., Paggiola, G., Petchey, T. H. M., Clark, J. H., Farmer, T. J., Hunt, A. J., Robert McElroy, C., Sherwood, J. (2016) Tools and techniques for solvent selection: green solvent selection guides. *Sustainable Chemical Processes*, **4** (1), 1–24.

39 ECHA. European Chemicals Agency: REACH. (n.d.) https://echa.europa.eu/regulations/reach (18 October 2016).

40 Li, Z., Smith, K. H., Stevens, G. W. (2015) The use of environmentally sustainable bio-derived solvents in solvent extraction applications—a review. *Chin J Chem Eng*, **24**.

41 Hansen, C. M. (2007) *Hansen solubility parameters: a user's handbook*. CRC Press.

42 Garcia, J. I., Garcia-Marin, H., Pires, E. (2014) Glycerol based solvents: Synthesis, properties and applications. *Green Chemistry*, **16** (3), 1007–1033.

43 Dros, A., Larue, O., Reimond, A., De Campo, F., Pera-Titus, M. (2015) Hexamethylenediamine (HMDA) from fossil-vs. bio-based routes: An economic and life cycle assessment comparative study. *Green Chemistry*, **17** (10), 4760–4772.

44 Montazeri, M., Soh, L., Pérez-López, P., Zimmerman, J. B., Eckelman, M. J. (2016) Time-dependent life cycle assessment of microalgal biorefinery co-products. *Biofuels, Bioproducts and Biorefining*.

45 Moity, L., Molinier, V., Benazzouz, A., Joossen, B., Gerbaud, V., Aubry, J.-M. (2016) A "top-down" in silico approach for designing ad hoc bio-based solvents: application to glycerol-derived solvents of nitrocellulose. *Green Chemistry*, **18** (11), 3239–3249.

46 Jessop, P. G. (2011) Searching for green solvents. *Green Chemistry*, **13** (6), 1391–1398.

47 Bassi, A., Yudken, J. (2009) Potential challenges faced by the US chemicals industry under a carbon policy. *Sustainability*, **1** (3), 592–611.

48 Jessop, P. G., Mercer, S. M., Heldebrant, D. J. (2012) CO_2-triggered switchable solvents, surfactants, and other materials. *Energy & Environmental Science*, **5** (6), 7240–7253.

49 Phan, L., Brown, H., White, J., Hodgson, A., Jessop, P. G. (2009) Soybean oil extraction and separation using switchable or expanded solvents. *Green Chemistry*, **11** (1), 53–59.

50 Vanderveen, J. R., Durelle, J., Jessop, P. G. (2014) Design and evaluation of switchable-hydrophilicity solvents. *Green Chem*, **16**.

51 Beckman, E. J. (2004) Supercritical and near-critical CO_2 in green chemical synthesis and processing. *The Journal of Supercritical Fluids*, **28** (2), 121–191.

52 King, J. W. (2014) Modern supercritical fluid technology for food applications. *Annual review of food science and technology*, **5**, 215–238.

53 Martin, L., Skinner, C., Marriott, R. (2015) Supercritical extraction of oil seed rape: Energetic evaluation of process scale. *The Journal of Supercritical Fluids*, **105**, 55–59.

54 Harmsen, G. J. (2007) Reactive distillation: The front-runner of industrial process intensification: a full review of commercial applications, research, scale-up, design and operation. *Chemical Engineering and Processing: Process Intensification*, **46** (9), 774–780.

55 Martín, M., Grossmann, I. E. (2012) Simultaneous optimization and heat integration for biodiesel production from cooking oil and algae. *Ind. Eng. Chem. Res.*, **51** (23), 7998–8014.

56 Patinha, D. J. S., Tomé, L. C., Florindo, C., Soares, H. R., Coroadinha, A. S., Marrucho, I. M. (2016) New low-toxicity cholinium-based ionic liquids with perfluoroalkanoate anions for aqueous biphasic system implementation. *ACS Sustainable Chemistry & Engineering*, **4** (5), 2670–2679.

57 Zhang, Y., Bakshi, B. R., Demessie, E. S. (2008) Life cycle assessment of an ionic liquid versus molecular solvents and their applications. *Environ. Sci. Technol.*, **42** (5), 1724–1730.

58 Pena-Pereira, F., Namieśnik, J. (2014) Ionic liquids and deep eutectic mixtures: Sustainable solvents for extraction processes. *ChemSusChem*, **7** (7), 1784–1800.

59 Toledo Hijo, A. A., Maximo, G. J., Costa, M. C., Batista, E. A., Meirelles, A. J. (2016) Applications of Ionic Liquids in the Food and Bioproducts Industries. *ACS Sustainable Chemistry & Engineering*.

60 Gore, R. G., Myles, L., Spulak, M., Beadham, I., Garcia, T. M., Connon, S. J., Gathergood, N. (2013) A new generation of aprotic yet Brønsted acidic imidazolium salts: effect of ester/amide groups in the C-2, C-4 and C-5 on antimicrobial toxicity and biodegradation. *Green Chemistry*, **15** (10), 2747–2760.

61 George, A., Brandt, A., Tran, K., Zahari, S. M. N. S., Klein-Marcuschamer, D., Sun, N., Sathitsuksanoh, N., Shi, J., Stavila, V., Parthasarathi, R. (2015) Design of low-cost ionic liquids for lignocellulosic biomass pretreatment. *Green Chemistry*, **17** (3), 1728–1734.

62 Peleteiro, S., Rivas, S., Alonso, J. L., Santos, V., Parajó, J. C. (2012) Utilization of Ionic Liquids in Lignocellulose Biorefineries as Agents for Separation, Derivatization, Fractionation, or Pretreatment. *J. Agric. Food Chem.*, **63** (37), 8093–8102.

63 Ruiz-Mercado, G. J., Smith, R. L., Gonzalez, M. A. (2012) Sustainability indicators for chemical processes: I. Taxonomy. *Ind. Eng. Chem. Res.*, **51** (5), 2309–2328.

5

Process Intensification in the Chemical Industry: A Review

Stefano Curcio
Department of Engineering Modeling, University of Calabria, Rende, Italy

5.1 Introduction

It is difficult to give an appropriate and exact definition of the term "Process Intensification" (PI). Often, process intensification is associated with attractive but vague expressions like "cheaper, smaller, cleaner" or "making more with less". However, all these purely descriptive and qualitative attributes do not contribute to defining the term "process intensification" in a rigorous way. The problem with the PI definition is actually related to the identification of the limit (if any), which allows discrimination between real process intensification and classical process optimization. Generally speaking, PI technologies tend to completely re-design conventional unit operations and, also, make use of novel equipment or processing methods such as multifunctional reactors, micro reactors, enhanced heat exchangers, alternative energy forms, and so on. The recent developments in both climate change and energy supply have confirmed the necessity of faster and wider application of innovative PI-technologies, which can offer significant advantages like decrease of capital and operating expenses, selectivity improvements, lead time reduction and safety improvement.

In recent years, the development and the exploitation of PI methods has seen a worldwide hastening. Starting from the mid-1980s, the interest of both academia and industry in process intensification increased to a significant extent, as testified by the number of articles published in this field. A literature search performed by *Web of Science*, considering

Sustainable Development in Chemical Engineering – Innovative Technologies, First Edition.
Edited by Vincenzo Piemonte, Marcello De Falco and Angelo Basile.

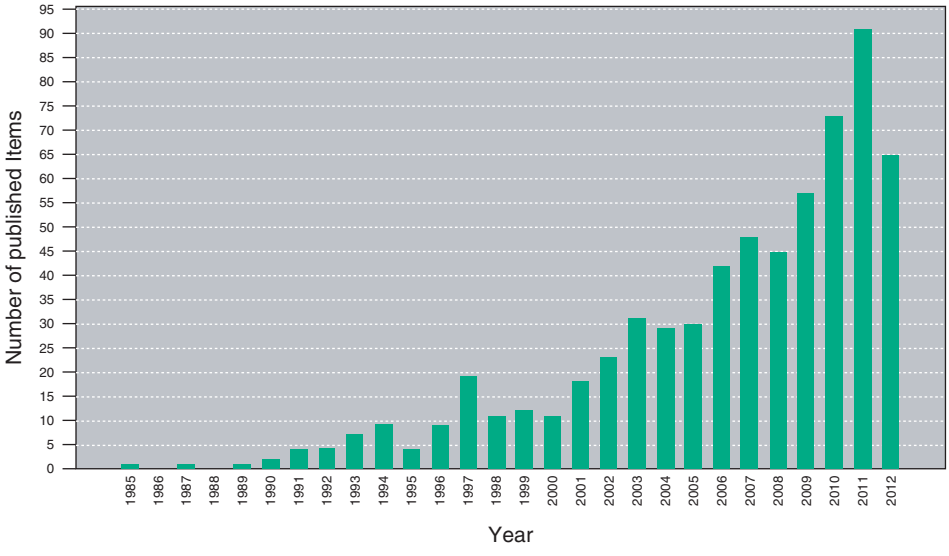


Figure 5.1 *Number of published papers – keywords: ''Chemical'' AND ''Process Intensification''. Data collated from www.webofknowledge.com*

the words "chemical" AND "process intensification" returned the result shown in Fig. 5.1: the number of publications in the last few years seems to stabilize in the range 45–90 papers per year, with a peak in 2011. In addition, it is worthwhile remarking that a scientific journal, namely *Chemical Engineering and Processing: Process Intensification*, focuses on PI and on different related fields.

Another fundamental reference in PI field is represented by the European Roadmap for Process Intensification, which presents a valuable review of PI technologies analyzed from an industrial standpoint.

5.2 Different Approaches to Process Intensification

PI methodologies have been extensively applied to improve the performance of several chemical processes. PI has the potential to positively affect a process by increasing its efficiency via a reduction in energy consumption, costs, volumes handled, waste generated, and, by improving the process safety without sacrificing product quality (Reay *et al*., 2008). Retrofit is a fundamental concept in PI and can be defined as the improvement of an existing plant by substituting or improving some of its constituent unit operations while fitting the rest of the plant and some of the process variables, namely the chemicals processed by the system, the reactions that may be occurring, the product purities that need to be achieved, a sub-set of equipment and operations in the process (Lutzea *et al*., 2010). Actually, a proper discrimination between drastic improvements and improvements as just a result of qualitative changes has to be done. Microreaction technology, monolithic reactors and reactive separations are classical examples of a drastic enhancement, since they allow, as compared to traditional apparatuses, improved

mass and heat transfer rates, thus achieving significant improvements as compared to traditional technologies. Alternative forms of energy supply for chemical reactions such as microwaves and ultrasound, the use of new reaction media such as ionic liquids, micro-emulsions and supercritical phases, or the use of new auxiliary agents such as phase transfer catalysts belong, instead, to the second category, even if they are also capable of improving the existing processes. The principles associated with PI have been classified (Lutzea *et al*., 2010) considering three different kinds of enhancements, which can be achieved through: (1) the integration of operations; (2) the integration of functions; and (3) the integration of phenomena. In this way, PI is accomplished by adding and enhancing a phenomenon within a function and/or operation for improved equipment design in order to maximize the driving force or to overcome the limitations occurring in the corresponding conventional process. In addition to these three principles, a targeted enhancement of a phenomenon can be considered as well; in this way, PI is achieved through improved use of the respective phenomena/functions to overcome the limitations occurring in a given conventional operation. Van Gerven and Stankiewicz (2009) defined four principles to achieve a completely intensified process: (1) to maximize the effectiveness of intra-and intermolecular events; (2) to optimize the driving forces at every scale and maximize the specific surface area to which these forces apply; (3) to maximize synergistic effects; (4) and to give each molecule the same processing experience. Finally, another possibility for the classification of process intensification measures is to divide them into three superordinate levels (Freund and Sundmacher, 2008). The most detailed level is the so-called *phase level*, at which molecule populations, which build up a thermodynamic phase, are considered. In a whole process, all the thermodynamic phases are embedded into apparatuses, or into individual process spaces. This is the so-called *process unit level*. Usually, the process consists of several such process units. The interconnection between the individual apparatuses and thus the overall process flowsheet can finally be analyzed at the superordinated *plant level*.

5.3 Process Intensification as a Valuable Tool for the Chemical Industry

From an industrial standpoint, PI has been successfully exploited in many different sectors, as discussed in the European Roadmap for Process Intensification.

In petrochemical industry, for instance, energy definitely represents a relevant part of total delivered product costs. Energy efficiency significantly affects the cost competitiveness and sustainability of petrochemical industry, which, in recent years, has already attained considerable improvements in energy efficiency. Given the large size of the petrochemical industry and the highly competitive market in which it operates, it is worthwhile combining reliability and predictability in order to achieve a decrease in energy expenses. Unexpected production interruptions do indeed have serious impacts on operational and logistic costs and may easily erode profits. In this context, PI may play an important role either in reducing energy costs or in lowering capital needs for investments, keeping safety and process reliability at the levels required by both legislation and market. PI does indeed aim at promoting innovation, creating technology

breakthroughs for any industrial sector and, therefore, also for petrochemical industry processes (Harmsen, 2010). As far as energy efficiency is concerned, it is expected that the exploitation of process intensification methodologies may determine at least a 20% improvement by 2050 (Sanders *et al*., 2012). This will be achieved by implementing new technologies, for example, new processes, bio-based and bio-catalytic routes, novel materials, hybrid reactors, novel separation systems, and so on. Like all new technologies, however, PI faces several barriers, which need to be overcome to effect an efficient PI implementation. These barriers can be summarized as: high cost to retrofit PI technologies in current production processes; risks of commercializing breakthrough technology; scale-up of PI; lack of PI knowledge, unfamiliarity with PI technologies and, finally, a long development path (Ren, 2009).

In the pharmaceutical/fine chemicals industry the use of process intensification methods is called to increase the selectivity of reactions (Yadav *et al*., 2003) and, thus, the material yield, reducing the lead time (i.e., the time between raw materials delivery and the completion of the product) of the entire production process. The achievement of these goals would also determine savings in energy consumption and contribute to related sustainability needs. Currently, the typical selectivity of chemical reactions is about 80%; PI technologies can increase selectivity up to 90%. For a classic multi-step process, this would achieve a rise in material yield from 30–60%, which significantly contributes to the sustainability of the whole process. Moreover, current processes in the fine chemicals industry may have up to 50 process steps, with a lead time of several months. The exploitation of improved design methods based on PI will allow operating integrated and continuous process steps, thus reducing these lead times to a significant extent. It is indeed expected that a multipurpose serial production train, based on PI technologies, can result in 50% reduction of production costs within 10–15 years. However, this vision can be achieved if PI technologies, whose efficiency has been proved on lab-scale plants only, are properly scaled-up and industrialized. However, a successfully implementation of PI technologies in the pharmaceutical/fine chemicals industry is actually dependent on a detailed insight, at molecular level, of both kinetics and thermodynamics of the reactions participating in the process (Gernaey *et al*., 2012). This knowledge is essential to formulate reliable models, which are called to predict the influence of process and operating variables on system behavior. A dependable model is indeed necessary for the implementation of a proper control system since it allows investigating how the process outputs may change with time under the influence of changes of the external disturbances and manipulated variables. Another important factor that has to be accounted for regards the development of effective on-line analytic techniques aimed at monitoring and validating the "intensified" continuous processes. Also in this case, PI faces several barriers (Lutzea *et al*., 2010): suppliers for industrial applications of PI technologies are lacking; the design of novel PI apparatuses on pilot-/industrial scale is expensive and technologically uncertain; the capital cost of novel PI modules per production capacity today is often higher than that of traditional reactors. In addition, efficient solutions for downstream processing in combination with PI reactors where the production of multiple products is combined in a single unit, are difficult to implement.

Agro-food industry is characterized by huge volumes or great flow rates of (very) diluted streams. Cost competitiveness of agro-food industry is significantly affected by two factors, namely the energy costs for processing and the wastes treatment cost.

Both these costs can be reduced exploiting process intensification methodologies (Doré *et al*., 2011) provided that an optimization, performed over the whole agro chain and considering – at least – crop harvesting, transportation, preliminary processing, refining, derivatization process, has been preliminarily implemented. The production chain and processing of agro-food industry is indeed constrained by the limited stability of crops and derived materials. PI can potentially achieve different kinds of optimization. Typical examples are: separation in (or close to) the field of certain crop components determines significant energy and transportation cost savings; preservation of crop components right after harvesting reduces the seasonality effect, leading to much higher capacity utilization of process equipment and related capital costs; valorization of non-foodstuff competitive crop components into the so-called second-generation bio-fuels can be achieved through the exploitation of new crops (waste oils, lignocellulosic materials, whey, etc.) (Sansonetti *et al*., 2009; De Paola *et al*., 2009) and novel harvesting technologies. For instance, using a PI standpoint, milk separation into water, proteins and fats can be performed on-site (i.e., at the farm) with low energy-consuming micro separators. In this way, product transportation to and handling at the factory can be limited to the relevant components, proteins and fats, saving energy and reducing CO_2 emissions by avoiding the unnecessary transportation of water. Different factors have to be analyzed before implementing PI technology in the agro-food industry: crops with short storage limits such as milk, fruit juices, sugar beets, and vegetables do indeed determine particular problems; final product quality is strongly affected by the variability of the agro material due to source and seasonality; finally, the significant increase of crops dedicated for bio-ethanol and bio-diesel production is determining a competition for valuable lands utilization, thus determining the necessity of both intensified crops and novel process approaches. Food industry performance is also affected by market conditions, which request accurate cost awareness that directly turn into technology improvements so as to achieve higher yields. Product quality and food safety are definitely essential issues, which need to be accounted for in relation to consumer and legislation requirements. Food companies, therefore, need updated product innovation in order to satisfy consumer trends and changing demand. In addition, energy efficiency increases, plant safety, and flexibility are considered as important factors, which may contribute to the implementation of proper PI methodologies. Different technologies, having a potential high impact on food technology, have been actually identified. In particular, it is expected that PI will lead to an overall energy efficiency increase of 30–40% by using: (1) milder processing techniques which will result also in a better product quality; (2) more selective treatment of raw materials and ingredients; (3) more effective removal of micro traces of selected compounds; (4) increased processing flexibility (Ammar *et al*., 2012). Moreover, it is expected that a decrease of (bio)-fouling in processing equipment and an increase in cleaning efficiency will determine a 20% capacity growth and a 60% energy saving. An important barrier to PI development in the agro-food industry is the limited availability of true process experts who, generally, concentrate on optimizing the existing plants and pay little attention to new, green, improved processes. This is partially to be ascribed to the fact that novel PI-based technologies are difficult to be incorporated in existing and well-established factories. Food industry is indeed a rather traditional sector not prone to adopting new technologies based, in particular, on PI methods (Fitzpatrick and Ahrné, 2005).

5.4 PI Exploitation in the Chemical Industry

In this section, a description of some process units designed on the basis of PI concepts will be presented, pointing out on their major features, on the advantages determined by the exploitation of these PI units and, in some cases, on the existing barriers that are currently limiting their spread on an industrial scale. Many of these units have already replaced the traditional apparatuses and improved the performance of conventional processes; some others are at a development stage, but it is expected that they may replace the traditional units in the future, hopefully in the next 10–15 years. The following critical review on each of the considered apparatus was readapted from the European Roadmap for Process Intensification where more detailed information can be found.

5.4.1 Structured Packing for Mass Transfer

Structured packing is used in various mass-transfer operations, including distillation, liquid/liquid mixing, absorption, and extraction. It offers high interfacial mass transfer areas and good phase dispersion resulting in high number of transfer units. Many different designs of structured packing are available; most of them are realized in order to form an open honeycomb structure with inclined flow channels and a relatively high exposed surface area (Bessoua *et al*., 2010).

5.4.2 Static Mixers

Static mixers are pipe inserts which generate radial mixing or, for multiphase systems, produce fine bubbles or droplets (Lobry *et al*., 2011). Static mixers are particularly useful for the continuous processing of chemicals; their main limitation is sensitivity to clogging. Compared to conventional mixing systems they are characterized by a very high-energy dissipation rate, resulting in very compact and energy efficient units.

5.4.3 Catalytic Foam Reactors

Solid foam catalytic reactors use a solid foam structure as the support for depositing a catalyst. These materials combine high void fraction and high surface area. Catalytic foam reactors (Palma *et al*., 2012) represent a valid alternative to fixed bed reactors, since they are characterized by lower pressure drop and, hence, lower energy consumption; in addition, in the case of metallic foams, heat transfer is enhanced and more equal temperature profiles are obtained. The major disadvantages are represented by the inherently higher costs of the foam-based catalyst and by the relatively low surface area for deposition of the catalytic material. This technology is still in an early stage of development.

5.4.4 Monolithic Reactors

Monolithic catalysts are made of ceramic materials or corrugated metal sheets and consist of a many narrow parallel channels whose walls are used for catalyst deposition. Monoliths are characterized by narrower residence time distribution, which corresponds, as compared to conventional units, to higher selectivity, higher specific geometric areas, lower mass transfer resistances and, hence, smaller reactor volume, very low

pressure drops and therefore, lower energy requirement (Edvinsson and Cybulski, 1994). Monolithic reactors are used in gas-phase cleaning in environmental applications; however, a cost reduction of monoliths and the exploitation of novel materials allowing higher catalyst loading are both required to achieve a wider spread of this technology.

5.4.5 Microchannel Reactors

Microreactors are chemical reactors of extremely small dimensions and usually a sandwich-like structure, consisting of a number of layers constituted by micromachined channels whose diameter may range between 20–400 μm (Fig. 5.2). Microreactors allow very high heat transfer rates (Regatte and Kaisare, 2011), not attainable by other apparatus. Very low reaction-volume-to-surface-area ratios make microreactors attractive for reactions involving poisonous or explosive reactants. Compared to traditional reactors, heat dissipation in microreactors is significantly enhanced by passing the reaction fluid through very small channels, realized in high thermally conductive metal blocks which, depending on the endothermic or exothermic nature of the reaction, are heated or cooled, respectively. The major advantage of microreactors is definitely represented by the excellent control of reaction temperature (Kockmann *et al*., 2011). The main drawbacks, instead, are high pressure drops, relevant clogging tendency, and high unit cost. Microreactors may be successfully exploited in both fine chemistry and the pharmaceutical industry.

5.4.6 Non-Selective Membrane Reactors

In a reactor equipped with a non-selective membrane, the membrane is used to provide a support for the deposition of a catalyst (Shuit and Ong, 2012; Basile *et al*., this book).

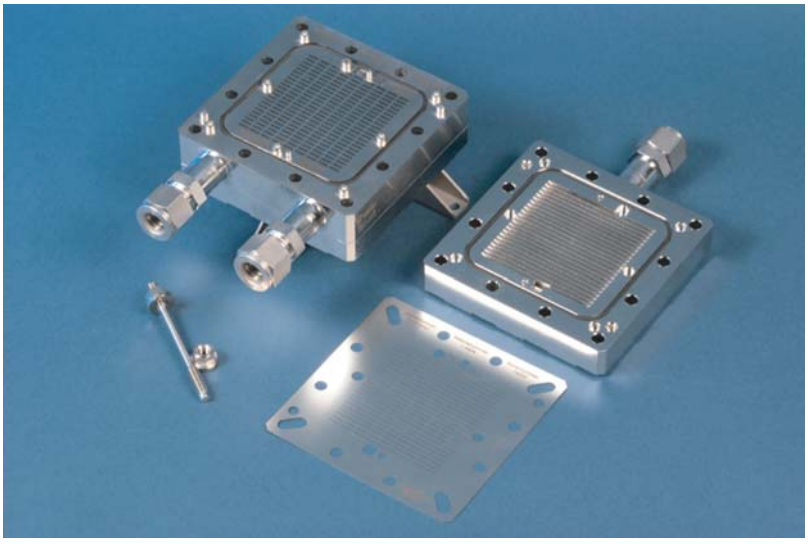


Figure 5.2 *Microchannel reactor. Reproduced from www.vinci-technologies.com*

The reaction, therefore, can take place either inside the pores of the membrane or on its external surface; the membrane is aimed at offering a very effective contact between reactants and catalyst, thus resulting in a short contact time catalytic reactor. An attractive exploitation of non-selective membrane reactor is the controlled dosing of one reactant along the length of the reactor (Rahimpour, 2009), which allows avoiding high local concentrations. The problems that need to be solved for a complete utilization of membrane technology on industrial scale are still represented by the system costs and the membrane stability.

5.4.7 Adsorptive Distillation

Adsorptive distillation is a three-phase mass transfer operation performed in two columns (Mujiburohman *et al*., 2006); the first one is used to achieve the actual separation, the second one to attain the adsorbent regeneration. The adsorbent is a fine powder, which is fluidized and circulated by an inert carrier. Adsorptive distillation is capable of enhancing separation efficiency, especially in the case of azeotropes or close-boiling components and can be exploited, especially in the fine chemical industry, to perform continuous processes. One significant improvement of this operation is actually represented by the suspension absorptive distillation, where the solid absorptive material acts as an extractant. This technology, however, is still in a very early stage of development and no demonstration of its feasibility or of the actual advantages over traditional alternatives has been yet provided.

5.4.8 Heat-Integrated Distillation

As compared to a classic distillation column, a dividing wall column is capable of delivering pure side fractions, thus reducing the number of necessary distillation columns in a separation sequence. A dividing wall column represents an example of Heat-Integrated Distillation Columns (Kiss and Suszwalak, 2012; Markowski *et al*., 2007) where the integration is achieved by the combination of the rectifying and the stripping columns in a single arrangement. Dividing wall columns have been effectively used in several chemical processes and can be considered to be a mature technology, which allows, in terms of energy and investment costs, savings even higher than 30%.

5.4.9 Membrane Absorption/Stripping

Membrane absorption occurs when a gaseous component is selectively transported through a membrane and, then, dissolved in an absorbing liquid (Fig. 5.3). One of the most important uses of membrane absorption is represented by CO_2 capture from flue gas (Feron and Jansen, 1997; Belaissaoui *et al*., 2012). A natural extension of membrane adsorption is represented by either the membrane-based absorption-desorption process, where two liquids are on both sides of the membrane, or the membrane stripping process, in which selected components are removed from the liquid phase through the membrane by a stripping gas. Membrane absorption is actually a bubble-free gas-liquid mass transfer unit operation, which presents several advantages in the case, for instance, of shear-sensitive biological systems.

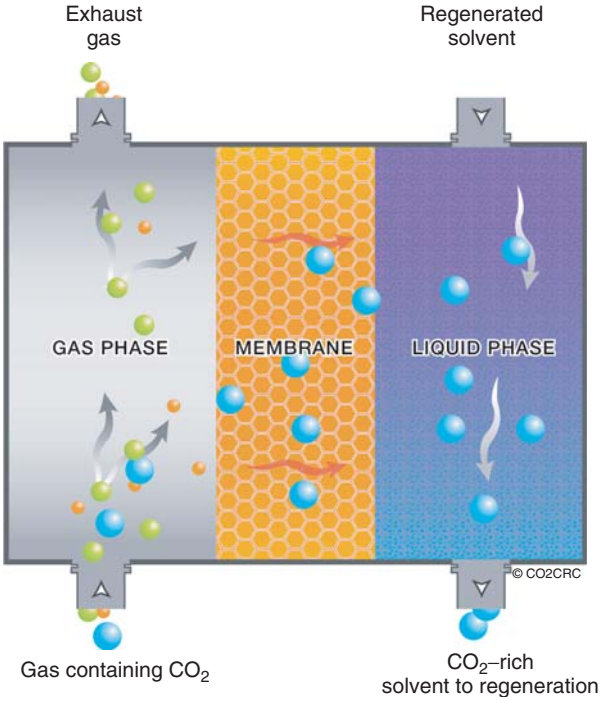


Figure 5.3 *Membrane absorption schematic. Reproduced with permission. Copyright © CO2CRC 2011*

5.4.10 Membrane Distillation

Membrane distillation (Calabrò *et al*., 1994) is a thermally driven unit operation consisting in transporting a volatile component of a liquid feed stream, as a vapor, through a porous membrane; the vapor is then condensed on the other side of the membrane (Fig. 5.4). This technology offers the potential of performing very efficient separations; however, energy (actually a lower amount as compared to conventional

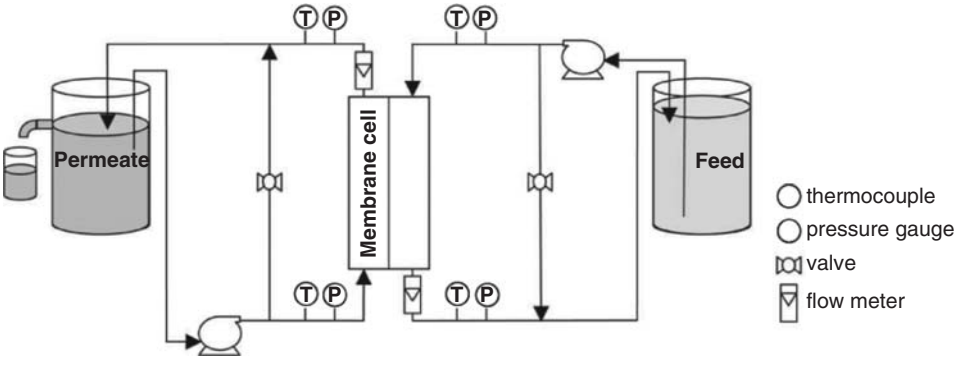


Figure 5.4 *Membrane distillation schematic. Reproduced from www.omicsonline.org*

distillation) is needed for vaporization. Another advantage is represented by the absence of concentration polarization, which, instead, afflicts other membrane operations.

5.4.11 Membrane Crystallization

Membrane crystallization is a novel crystallization technique, used – for instance – for protein crystals growing with enhanced crystallization kinetics. Membrane crystallization allows obtaining better crystal forms under operating conditions that are unsuitable for conventional crystallization processes. This technology is actually given as the combination of crystallization and membrane distillation (Di Profio *et al*., 2010) and is based on a rather simple principle: the solvent evaporates (by steam/vacuum) at the membrane interface, migrates through the pores of the membrane and eventually condenses on the opposite side of the membrane. The membrane is capable to activate heterogeneous nucleation starting at low super saturation, thus enhancing the kinetics of crystallization, even for large molecules, like proteins (Di Profio *et al*., 2006). Since all solvent must be evaporated, the process can be rather energy-intensive, even if it operates at a low temperature-difference as compared to the established technology, thus potentially resulting in energy savings. Moreover, due to the lower temperature differences, the crystallization process is carried out at milder conditions and in a more controlled way, which could be of great interest for the pharmaceutical industry.

5.4.12 Distillation-Pervaporation

A pervaporation membrane can be coupled with a conventional distillation column, thus resulting in a hybrid membrane/distillation process (Naidu and Malik, 2011), where the membrane can be placed either on the overhead vapor or on the feed of distillation column. Examples of potential improvements of this technology as compared to conventional processes are breaking an azeotrope, without the addition of a solvent, increasing the capacity of a distillation column and improving quality of the bottom and overhead products. The potential of the technology has been already proved in the case of alcohols dehydration and of the separation of isomeric hydrocarbons.

5.4.13 Membrane Reactors

Membrane reactors represent a very effective system in equilibrium limited reactions, since the products are continuously and selectively removed, thus favoring the forward reaction according to Le Chatelier's principle (Gallucci *et al*., 2008). Two well-known examples of membrane reactors consist in hydrogen conducting membranes, based on palladium (Basile *et al*., 2008), and oxygen conducting membranes, based on perovskites (Evdou *et al*., 2008). A very wide range of principles and characteristics are actually known; this leads to a broad range of operating conditions that can be used to operate a membrane reactor. Figure 5.5 shows a palladium membrane reactor for the direct synthesis of phenols from benzene.

5.4.14 Heat Exchanger Reactors

In a heat exchanger reactor (Anxionnaz *et al*., 2008, Despènes *et al*., 2012) the reaction occurs close to a heat exchange surface, which is aimed at removing (or supplying for

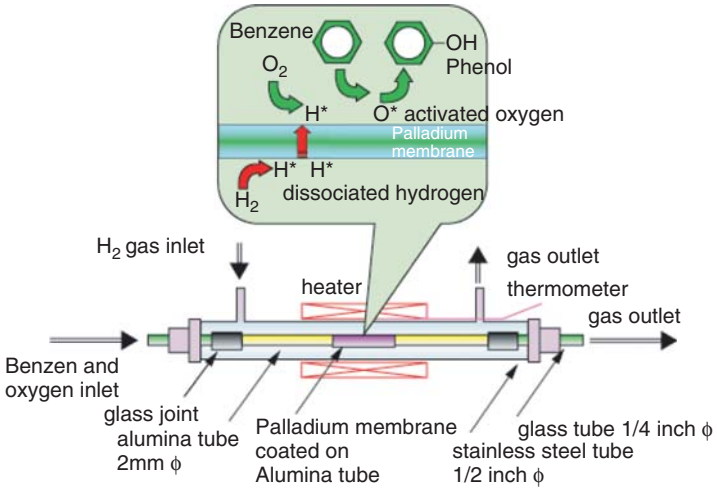


Figure 5.5 *Palladium membrane reactor for the direct synthesis of phenols from benzene in a single-stage reaction. Reproduced from www.aist.go.jp*

an endothermic reaction) the heat generated by the reaction. In this way, an accurate control of the operating temperature is actually attained. Both non-catalytic and catalytic heat-exchanger reactors show interesting promise for (very) fast reactions characterized by high heat of reaction.

5.4.15 Simulated Moving Bed Reactors

The simulated moving-bed reactor combines continuous countercurrent chromatographic separation with chemical reaction (Kundu *et al*., 2009). This hybrid, not energy-intensive process is competitive with traditional processes in which reaction and separation are performed in different units. Higher conversions and better yields can be attained by separating the reaction product(s) in an equilibrium reaction. This technology is applied in those processes where chromatographic separation is a necessary step, that is, when high quality of separation is required and the product has a high value (fine chemicals and pharmaceuticals).

5.4.16 Gas-Solid-Solid Trickle Flow Reactor

In the Gas-Solid-Solid Trickle Flow Reactor fine adsorbent trickles through a fixed bed of catalyst (Kuczynski *et al*., 1987); one or more products are selectively removed from the reaction zone. In case of methanol synthesis, for examples, conversions significantly higher than the equilibrium conversions are achieved under the same operating conditions. The major advantages of this technology are energy savings, increased conversion, and high potential for innovations as given by the utilization of multifunctional catalysis or of structured packing (Muzen and Cassanello, 2007). A current barrier is represented by the fact that, generally, solid recycling is less attractive than liquid or gas recirculation.

5.4.17 Reactive Extraction

Reactive extraction encompasses simultaneous reaction and liquid-liquid separation (Dussan *et al*., 2010). Reactive extraction may be exploited in multi-reaction systems to attain significant improvements in both yield and selectivity of desired products. This leads to a reduction of both recycle streams and wastes. The combination of reaction and liquid-liquid extraction can be also used for the separation of waste byproducts, which are difficult to remove be conventional methods.

5.4.18 Reactive Absorption

Reactive absorption (Noeres *et al*., 2003) is actually a mature technique used in the production of nitric or sulfuric acid, for the removal, by amine solutions, of the CO_2 contained in the flue gas, for gas desulfurization, in which H_2S is removed and converted to sulfur, and for the separation of light olefins and paraffins (Ortiz *et al*., 2010). Reactive absorption is the most applied reactive separation technique.

5.4.19 Reactive Distillation

Reactive Distillation is performed in a conventional distillation column where some species react in the presence of a structured catalyst (Noeres *et al*., 2002); the obtained products are continuously separated by fractionation, thus favoring the forward reaction; in some cases, even total conversion can be reached. The major benefits of this operation are: lower energy requirements, higher yields, good product purity, and lower capital investment. Reactive distillation may also be exploited as an effective separation method in the case of mixtures containing reactive and non-reactive components with near boiling point (Lai *et al*. 2007). Both homogeneous and heterogeneous catalysis can be applied, even if some problems regarding the catalyst development (performance, kinetics, stability, morphology, coating procedures, etc.) are still to be solved.

5.4.20 Membrane-Assisted Reactive Distillation

In some cases characterized by particular limitations, that is, the presence of an azeotrope, reactive distillation cannot fulfill the desired process performance and another unit operation has to be combined. This could be achieved by coupling a reactive distillation unit to a pervaporation module (Buchaly *et al*., 2007) so as to further purify the distillate. Two commercial applications have been reported so far, that is the production of methyl borate and the production of fatty acid esters (Kiatkittipong *et al*., 2011).

5.4.21 Hydrodynamic Cavitation Reactors

The energy associated with a liquid in motion can be successfully utilized to promote cavitation, which is aimed at intensifying reactions and other operations, such as homogenization or emulsification. Figure 5.6 shows a typical hydrodynamic cavitation chamber. Hydrodynamic cavitation can be obtained by allowing the liquid pass through a throttling valve, an orifice plate or any other mechanical constriction (Gogate and Kabadi, 2009). If the operating pressure falls below the cavitation pressure, millions of microcavities are obtained. These bubbles then implode when pressure is

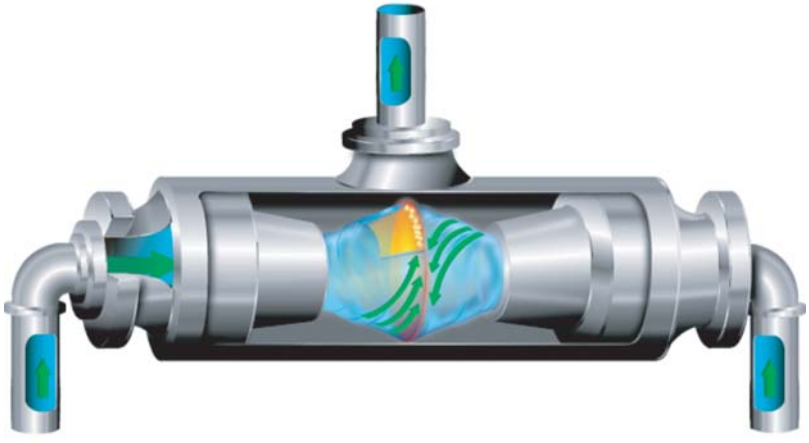


Figure 5.6 *Hydrodynamic cavitation chamber. Reproduced from vrtxtech.com*

increased. Hydrodynamic cavitation may improve the performance of several industrial transformations, as compared to conventional processes (Gogate and Pandit, 2005). For example, cavitated corn slurry exhibits higher yields in ethanol production as compared to uncavitated corn slurry processing. A significant improvement is achieved also in the mineralization of bio-refractory compounds (Gogate, 2002), which would otherwise need extremely high temperature and pressure conditions to remove the free radicals generated during processing.

5.4.22 Pulsed Compression Reactor

A novel chemical reactor concept derived from PI approach allows a technological breakthrough in syngas production (Roestenberg *et al*., 2011). This novel concept completely modifies the traditional standpoints since, instead of formulating improved catalysts, which may allow decreasing the operating temperature, no catalyst is used and the reaction is performed at very high temperatures, with peak values in the order of 1500–5000 K (Roestenberg *et al*., 2010). The reactor consists of a double-ended cylinder and a free piston, which separates the cylinder into two compression-reaction chambers. The piston reciprocates compressing the feed gas, until it reacts, in turn in the lower and in the upper chamber. The reciprocation is maintained by the released reaction energy. This technology, even if it is an early stage of development, could provide an excellent potential for energy savings (reaction energy is directly applied); moreover, CO_2 and NO_x emissions are reduced since feed heating is actually not required. Cost effectiveness is also improved by narrower reaction product distribution (less downstream processing) and more compact reactor area design.

5.4.23 Sonochemical Reactors

In a liquid system the exposure to ultrasound determines, in a time period of a few milliseconds, the formation, growth, and subsequent collapse of microbubbles (Parvizian *et al*., 2011). The micro-implosions are followed by high energy release, which leads

to local generation of extremely high temperatures and pressures, which can be up to 5000 K and 2000 bar, respectively (Sutkar and Gogate, 2009). The use of ultrasound significantly enhances the rate of any chemical reaction, thus increasing the product yield. For chemical syntheses, this technology is in the early stages of its development; however, it may have potentially interesting applications in food processing, biotechnology and environmental protection.

5.4.24 Ultrasound-Enhanced Crystallization

Sonocrystallization is a non-invasive method based on the exploitation of ultrasound to control the point of nucleation and the number of nuclei formed in a crystallization process (Nalajala and Moholkar, 2011). This generally enhances the crystal yields, improves the product properties, namely handling and appearance, reduces the crystal's agglomeration and increases the process reproducibility. An additional advantage is given by the formation of ultra-fine, nano-structured materials; moreover, a significant energy and cost saving can be attained since expensive milling or recrystallization steps are avoided. This technology provides very interesting perspectives in both pharmaceutical and food industry, but reasonably will need about 10–15 years of fundamental research to reach full commercial application.

5.4.25 Electric Field-Enhanced Extraction

Electric fields are commercially employed to increase the process rates and to control the droplet size in pain spraying and in surface coating processes. Other well-known applications regard gas cleaning, emulsion breaking, ink-yet spraying, fuel spraying, crude oil desalting and bulk chemical washing. Electric fields can also enhance liquid-liquid or solid-liquid extraction (Grimi *et al*., 2007, Gachovska *et al*., 2010) and improvements of about 200%, as compared to conventional processes, were reported. Typical advantages of this technology are energy savings and equipment size reduction; however, even if well developed for some specific applications, no new innovative products are to be expected in the future.

5.4.26 Induction and Ohmic Heating

Inductive heating is a process where electric currents are induced within a food or any other product due to the presence of oscillating electromagnetic fields generated by electric coils (Sastry and Barach, 2000). Inductive heating is different from microwave heating due to the frequency and the nature of the source. In microwaves in fact, the frequency is specifically assigned and a magnetron (not coils) is used to promote heating. Ohmic heating (also known as Joule heating), electro-conductive heating, or direct electrical resistance heating, involves the direct passage of electric current through a material for the purpose of heating it. Ohmic heating necessarily involves electrodes that are put in contact with the product to create a continuous electrical circuit. The most promising application of Ohmic heating is the continuous sterilization of foods (Goullieux and Pain, 2005) by high-temperature-short-time processing, which yields a significant quality improvement. Figure 5.7 shows a typical ohmic heater exploited in the food industry.

Figure 5.7 *Ohmic heater for the food industry. Reproduced from www.kasag.ch*

5.4.27 Microwave Drying

In general, microwave frequencies range from 0.3–300 GHz, but, in order to avoid any interference with radar and telecommunication applications, the industrial and domestic microwave appliances operate at a standard allocated frequency, most often at 2.45 GHz. Molecules with a permanent dipole moment (e.g., water) can rotate in a fast changing electric field; in addition, in substances where free ions or ionic species are present, the energy is also transferred by the ionic motion in an oscillating microwave field. As a result of both these mechanisms, the substance is heated directly and almost regularly. The heating extent depends on dielectric properties, namely the dielectric constant and the loss tangent, of the substance to be heated. Some materials absorb the microwave energy very easily; others are transparent or impermeable to it; this may allow a selective heating of materials/products (Holtz *et al*., 2010). Microwave-enhanced drying is used on industrial scale in food, wood (Bartholme *et al*., 2009), textile and pharmaceutical industries (Mcloughlin *et al*., 2000). The speed of the MW drying allows preventing unwanted degradation of some thermo-labile components that it is wanted to preserve in the dried material. Microwave heating allows remarkable energy savings compared to conventional processes. Figure 5.8 shows an industrial-scale microwave oven exploited to achieve wood drying.

5.4.28 Microwave-Enhanced Separation and Microwave Reactors

Microwave heating proved to enhance also some extraction operations, particularly the extraction of pharmaceutical ingredients from plant material. A limited number of papers

Figure 5.8 *Microwave oven for wood drying. Reproduced from Jiyuan Electric Co., Ltd.*

regarding the enhancement effects of microwaves on membrane separation and distillation have been published as well. The ability of microwave heating to accelerate chemical reactions and to improve the product yield has been extensively reported (Monsef-Mirzai *et al*., 1995). For example, in microwave-assisted oxidative coupling of methane on alumina supported La_2O_3/CeO_2 catalyst conversion occurred at temperatures about 250°C lower than those used by the conventional heating. However, no consolidated data exist on the actual energy/cost savings achievable by microwave reactors.

5.4.29 Photochemical Reactors

Photochemical reactors use the energy of light to initiate or catalyze reactions (Gupta *et al*., 2011). The basic principle is that light quanta are absorbed by chemical species which are excited and become more reactive with regards to other compounds present. The light can originate either from the Sun or from artificial sources, namely medium-pressure mercury or a xenon lamp. In the case of solar photochemical reactors, the energy has to be concentrated so as to attain sufficient efficiency. In non-catalyzed photochemical reactions the light energy is absorbed by a reagent or by a sensitizer, which transfers the electronic energy to the reagent or undergoes a reversible redox reaction with the reagent. Currently, the major applications of photochemical reactors are in production of protective and decorative coatings, inks, packaging, and electronic materials. In other fields, like chemical, pharmaceutical, and food industry the applications are rather rare. The main benefits of this technology regard the use of low temperatures, which means significant energy savings, and very high conversion/yield/selectivity.

5.4.30 Oscillatory Baffled Reactor Technologies

The Oscillatory Baffled Reactor technology generally uses a cylindrical column containing equally spaced orifice baffles and superimposed fluid oscillation (Vilar *et al*., 2008); this allows plug flow conditions even at low (laminar) flow rates, thus inducing enhanced mass and heat transfers as compared to conventional stirred tank reactors. Vortices are generated when fluid flows through the baffles; the continuous generation and cessation of eddies creates uniform mixing in each baffled cell (Smith and Mackley, 2006). The major advantages of this technology regard a significant energy/utility savings, higher yields and less side product. In addition, capital cost savings are achieved through much more compact designs.

5.4.31 Reverse Flow Reactor Operation

A well-known example of the integration of reaction and heat transfer in a multifunctional single unit is reverse-flow reactors (Simeone *et al*., 2012) where one or more process variables are intentionally and continuously perturbed (see Fig. 5.9); this dynamic operation results in process improvements, which cannot be achieved by steady state operation. In the case of exothermic processes the periodic flow reversal, typical of such systems, allows for a very good utilization of the heat of reaction, which is maintained within the catalyst bed and, after the reversion of the flow, is used to pre-heat the stream of cold reactants entering the reactor. So far, the reverse-flow reactors have been used in the SO_2 oxidation (Gosiewski, 1993), in the total oxidation of hydrocarbons contained in the off-gases and in the NO_x reduction (Botar-Jid *et al*., 2007). The major benefits of this technology are represented by energy savings, increased conversion selectivity, and enhanced productivity.

5.4.32 Pulse Combustion Drying

The term pulse combustion originates from the intermittent combustion of the solid, liquid, or gaseous fuel in contrast to the continuous combustion that occurs in conventional burners (Zbiciński *et al*., 2001). Such periodic combustion generates pressure,

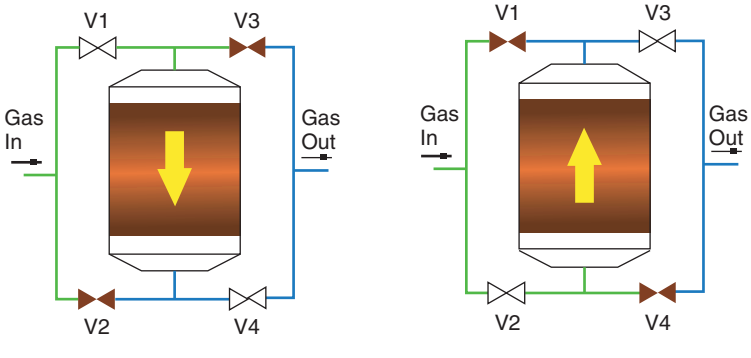


Figure 5.9 *Reverse flow reactor operation. Reproduced from www.matrostech.com*

velocity, and temperature waves which propagate from the combustion chamber to the drying chamber. Due to such wave propagation, the rates of heat and mass transfer and, consequently the drying rate, are significantly improved as compared to conventional processes. The pulse combustion dryer consists of a pulse combustor combined to a spray dryer or a rotary kiln or a fluid-bed dryer (Zbiciński, 2002). Pulse combustion drying represents an energy-efficient and environmentally-friendly technology; as compared to traditional spray dryers, it indeed provides lower energy consumption, lower capital cost, and lower CO_2 emission. In spite of these advantages, the application of pulse combustion is not widespread.

5.4.33 Supercritical Separation

Supercritical fluids have some properties, for example, density, that are quite similar to those of a liquid, and some other properties, such as low viscosity, low surface tension, and high diffusivities for solutes, which makes them similar to a gas (Gere *et al*., 1997). This is due to the fact that above the supercritical point there is no difference between the phases. Supercritical fluids, due to their unique properties, may significantly improve the performance of some conventional separation processes, like for example extraction. In fact, some compounds which are almost insoluble in a fluid at ambient conditions can become soluble in the fluid at supercritical conditions (see Fig. 5.10). Currently CO_2 is mostly used but other compounds, namely hexane, pentane, and ammonia, can be

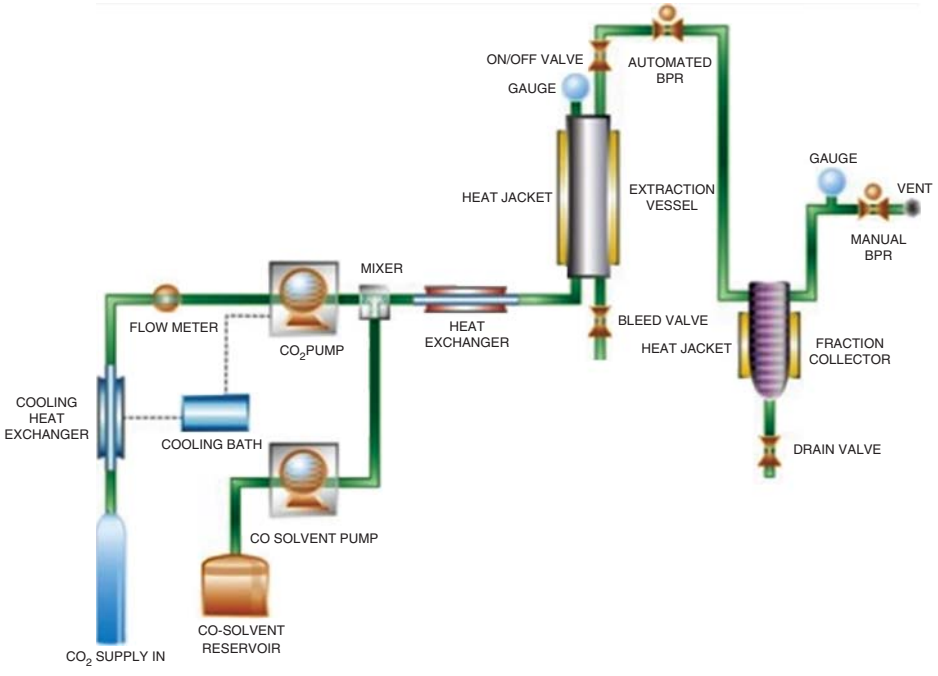


Figure 5.10 *Supercritical fluid extraction system. Reproduced from www.waters.com*

exploited as well. Supercritical separations using CO_2 are well established in extraction of natural materials (Reverchon, 1997). By using CO_2 in supercritical conditions, that is, when the substance is like a liquid, high value compounds can be extracted from mixtures; after pressure reduction, the supercritical fluid becomes a gas and the compounds can be recovered in a pure/not dissolved form. The major benefit is therefore represented by the avoidance of a traditional solvent and its associated waste problems, that is, emissions to the atmosphere or waste treatment.

5.5 Conclusions

Process intensification is a valuable tool for the development of more sophisticated and more efficient processes aimed at sustainable production in industry. Process intensification methodologies have already offered substantial benefits in material and energy efficiency, and the contribution of PI concepts was fundamental in designing novel types of equipment and of processes. Further improvements can be attained if operation and control of a complete process are considered simultaneously and systematically together with different PI design options. In order to find true optimal configuration it is also necessary to exploit a rigorous methodology, which allows determining intensified options by stepwise reduction of the search space through constraints, performance evaluation and objective function. In this context, a set of PI metrics accounting for economic, environmental, safety and process performance has to be developed to compare, on the basis of precise modeling techniques, different possible options, which allow intensifying a process through a standard set of performance measures.

References

Y. Ammar, S. Joyce, R. Norman, *et al*., Low grade thermal energy sources and uses from the process industry in the UK (2012), *Applied Energy*, **89** (1), 3–20.

Z. Anxionnaz, M. Cabassud, C. Gourdon and P. Tochon, Heat exchanger/reactors (HEX reactors): Concepts, technologies: State-of-the-art (2008), *Chemical Engineering and Processing: Process Intensification*, **47** (12), 2029–2050.

M. Bartholme, G. Avramidis, W. Viöl and A. Kharazipour, Microwave drying of wet processed wood fibre insulating boards (2009), *European Journal of Wood and Wood Products*, **67** (3), 357–360.

A. Basile, F. Gallucci and S. Tosti, Ch. 8: Synthesis, Characterization, and Applications of Palladium Membranes (2008), in: *Inorganic Membranes: Synthesis, Characterization and Applications, in Membrane Science and Technology*, Vol. **13**, 255–323, Elsevier B.V., Amsterdam.

A. Basile, A. Iulianelli and S. Liguori, Process intensification in the chemical and petrochemical industry, Ch. 6, this book.

B. Belaissaoui, D. Willson and E. Favre, Membrane gas separations and post-combustion carbon dioxide capture: Parametric sensitivity and process integration strategies (2012), *Chemical Engineering Journal*, **211–212**, 122–132.

V. Bessoua, D. Rouzineaua, M. Prévosta, *et al*., Performance characteristics of a new structured packing (2010), *Chemical Engineering Science*, **65** (16), 4855–4865.

C. C. Botar-Jid, P.Ş. Agachi and D. Fissore, Comparison of reverse flow and countercurrent reactors in case of selective catalytic reduction of NO_x (2007), *Computer Aided Chemical Engineering*, **24**, 1331–1336.

C. Buchaly, P. Kreis and A. Górak, Hybrid separation processes – Combination of reactive distillation with membrane separation (2007), *Chemical Engineering and Processing: Process Intensification*, **46** (9), 790–799.

V. Calabrò, B. L. Jiao and E. Drioli, Theoretical and experimental study on membrane distillation in the concentration of orange juice (1994), *Industrial & Engineering Chemistry Research*, **33** (7), 1803–1808.

M. G. De Paola, E. Ricca *et al*., Factor analysis of transesterification reaction of waste oil for biodiesel production (2009), *Bioresource Technology*, **100**, 5126–5131.

L. Despènes, S. Elgue, C. Gourdon and M. Cabassud, Impact of the material on the thermal behaviour of heat exchangers-reactors (2012), *Chemical Engineering and Processing: Process Intensification*, **52**, 102–111.

G. Di Profio, E. Curcio and E. Drioli, Controlling protein crystallization kinetics in membrane crystallizers: effects on morphology and structure (2006), *Desalination*, **200** (1–3), 598–600.

G. Di Profio, E. Curcio and E. Drioli, Membrane Crystallization Technology (2010), in *Comprehensive Membrane Science and Engineering, Volume 4: Membrane Contactors and Integrated Membrane Operations*, ISBN: 978-0-08-093250-7, Elsevier B.V., Amsterdam.

T. Doré, D. Makowski, E. Malézieux, *et al*., Facing up to the paradigm of ecological intensification in agronomy: Revisiting methods, concepts and knowledge (2011), *European Journal of Agronomy*, **34** (4), 197–210.

K. J. Dussan, C. A. Cardona, O. H. Giraldo, *et al*., Analysis of a reactive extraction process for biodiesel production using a lipase immobilized on magnetic nanostructures (2010), *Bioresource Technology*, **101** (24), 9542–9549.

R. K. Edvinsson and A. Cybulski, A comparative analysis of the trickle-bed and the monolithic reactor for three-phase hydrogenations (1994), *Chemical Engineering Science*, **49** (24 Part 2), 5653–5666.

A. Evdou, L. Nalbandian and V. T. Zaspalis, Perovskite membrane reactor for continuous and isothermal redox hydrogen production from the dissociation of water (2008), *Journal of Membrane Science*, **325** (2), 704–711.

P. H. M. Feron and A. E. Jansen, The production of carbon dioxide from flue gas by membrane gas absorption (1997), *Energy Conversion and Management*, **38** (Supplement), S93–S98.

J. J. Fitzpatrick and L. Ahrné, Food powder handling and processing: Industry problems, knowledge barriers and research opportunities (2005), *Chemical Engineering and Processing: Process Intensification*, **44** (2), 209–214.

H. Freund and K. Sundmacher, Towards a methodology for the systematic analysis and design of efficient chemical processes Part 1. From unit operations to elementary process functions (2008), *Chemical Engineering and Processing: Process Intensification*, **47**, 2051–2060.

T. Gachovska, D. Cassada, J. Subbiah, *et al*., Enhanced anthocyanin extraction from red cabbage using pulsed electric field processing (2010), *Journal of Food Science*, **75** (6), 323–329.

F. Gallucci, S. Tosti and A. Basile, Pd–Ag tubular membrane reactors for methane dry reforming: A reactive method for CO2 consumption and H2 production (2008), *Journal of Membrane Science*, **317** (1–2), 96–105.

D. R. Gere, L. G. Randall and D. Callahan, Ch. 11: Supercritical fluid extraction: Principles and applications (1997), in *Techniques and Instrumentation in Analytical Chemistry*, Vol. **18**, 421–484, Elsevier B.V., Amsterdam.

K. V. Gernaey, A. E. Cervera-Padrell and J. M. Woodley, A perspective on PSE in pharmaceutical process development and innovation (2012), *Computers & Chemical Engineering*, **42**, 15–29.

P. R. Gogate, Cavitation: an auxiliary technique in wastewater treatment schemes (2002), *Advances in Environmental Research*, **6** (3), 335–358.

P. R. Gogate and A. M. Kabadi, A review of applications of cavitation in biochemical engineering/biotechnology (2009), *Biochemical Engineering Journal*, **44** (1), 60–72.

P. R. Gogate and A. B. Pandit, A review and assessment of hydrodynamic cavitation as a technology for the future (2005), *Ultrasonics Sonochemistry*, **12** (1–2), 21–27.

K. Gosiewski, Dynamic modelling of industrial SO2 oxidation reactors. Part II. Model of a reverse-flow reactor (1993), *Chemical Engineering and Processing: Process Intensification*, **32** (4), 233–244.

A. Goullieux and J. P. Pain, Ch. 18: Ohmic Heating (2005), in *Emerging Technologies for Food Processing*, 469–505, D.-W. Sun (ed.), Elsevier Ltd., Oxford, ISBN: 9780126767575.

N. Grimi, I. Praporscic, N. Lebovka and E. Vorobiev, Selective extraction from carrot slices by pressing and washing enhanced by pulsed electric fields (2007), *Separation and Purification Technology*, **58** (2), 267–273.

V. K. Gupta, R. Jain, S. Agarwal and M. Shrivastava, Kinetics of photo-catalytic degradation of hazardous dye Tropaeoline 000 using UV/TiO 2 in a UV reactor (2011), *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, **378** (1–3), 22–26.

J. Harmsen, Process intensification in the petrochemicals industry: Drivers and hurdles for commercial implementation (2010), *Chemical Engineering and Processing: Process Intensification*, **49** (1), 70–73.

E. Holtz, L. Ahrné, M. Rittenauer and A. Rasmuson, Influence of dielectric and sorption properties on drying behaviour and energy efficiency during microwave convective drying of selected food and non-food inorganic materials (2010), *Journal of Food Engineering*, **97** (2), 144–153.

W. Kiatkittipong, P. Intaracharoen, N. Laosiripojana, *et al*., Glycerol ethers synthesis from glycerol etherification with tert -butyl alcohol in reactive distillation (2011), *Computers & Chemical Engineering*, **35** (10), 2034–2043.

A. A. Kiss and D. J. P. C. Suszwalak, Enhanced bioethanol dehydration by extractive and azeotropic distillation in dividing-wall columns (2012), *Separation and Purification Technology*, **86**, 70–78.

N. Kockmann, M. Gottsponer and D. M. Roberge, Scale-up concept of single-channel microreactors from process development to industrial production (2011), *Chemical Engineering Journal*, **167** (2–3), 718–726.

M. Kuczynski, M. H. Oyevaar, R. T. Pieters and K. R. Westerterp, Methanol synthesis in a countercurrent gas-solid-solid trickle flow reactor: An experimental study (1987), *Chemical Engineering Science*, **42** (8), 1887–1898.

P. K. Kundu, Y. Zhang and A. K. Ray, Modeling and simulation of simulated countercurrent moving bed chromatographic reactor for oxidative coupling of methane (2009), *Chemical Engineering Science*, **64** (24), 5143–5152.

I. K. Lai, S. B. Hung, W. J. Hung, *et al*., Design and control of reactive distillation for ethyl and isopropyl acetates production with azeotropic feeds (2007), *Chemical Engineering Science*, **62** (3), 878–898.

E. Lobry, F. Theron, C. Gourdon, *et al*., Turbulent liquid–liquid dispersion in SMV static mixer at high dispersed phase concentration (2011), *Chemical Engineering Science*, **66** (23), 5762–5774.

P. Lutzea, R. Gani and J. M. Woodley, Process intensification: A perspective on process synthesis (2010), *Chemical Engineering and Processing: Process Intensification*, **49**, 547–558.

M. Markowski, M. Trafczynski and K. Urbaniec, Energy expenditure in the thermal separation of hydrocarbon mixtures using a sequence of heat-integrated distillation columns (2007), *Applied Thermal Engineering*, **27** (7), 1198–1204.

C. M. McLoughlin, W. A. M. McMinn and T.R.A. Magee, Microwave Drying of Pharmaceutical Powders (2000), *Food and Bioproducts Processing*, **78** (2), 90–96.

P. Monsef-Mirzai, W. R. McWhinnie, M. C. Perry and P. Burchill, Reaction of phosphorylating reagents with substituted phenols and coals, using microwave heating (1995), *Fuel*, **74** (7), 1004–1008.

M. Mujiburohman, W. B. Sediawan and H. Sulistyo, A preliminary study: Distillation of isopropanol–water mixture using fixed adsorptive distillation method (2006), *Separation and Purification Technology*, **48** (1), 85–92.

A. Muzen and M. C. Cassanello, Flow regime transition in a trickle bed with structured packing examined with conductimetric probes (2007), *Chemical Engineering Science*, **62** (5),1494–1503.

Y. Naidu and R. K. Malik, A generalized methodology for optimal configurations of hybrid distillation–pervaporation processes (2011), *Chemical Engineering Research and Design*, **89** (8), 1348–1361.

V. S. Nalajala and V. S. Moholkar, Investigations in the physical mechanism of sonocrystallization (2011), *Ultrasonics Sonochemistry*, **18** (1), 345–355.

C. Noeres, A. Hoffmann and A. Górak, Reactive distillation: Non-ideal flow behaviour of the liquid phase in structured catalytic packings (2002), *Chemical Engineering Science*, **57** (9), 1545–1549.

C. Noeres, E. Y. Kenig and A. Górak, Modelling of reactive separation processes: reactive absorption and reactive distillation (2003), *Chemical Engineering and Processing: Process Intensification*, **42** (3), 157–178.

A. Ortiz, L. M. Galán, D. Gorri, *et al*., Kinetics of reactive absorption of propylene in RTIL-Ag + media (2010), *Separation and Purification Technology*, **73** (2), 106–113.

V. Palma, A. Ricca and P. Ciambelli, Monolith and foam catalysts performances in ATR of liquid and gaseous fuels (2012), *Chemical Engineering Journal*, **207–208**, 577–586.

F. Parvizian, M. Rahimi and M. Faryadi, Macro- and micromixing in a novel sonochemical reactor using high frequency ultrasound (2011), *Chemical Engineering and Processing: Process Intensification*, **50** (8), 732–740.

M. R. Rahimpour, Enhancement of hydrogen production in a novel fluidized-bed membrane reactor for naphtha reforming (2009), *International Journal of Hydrogen Energy*, **34** (5), 2235–2251.

D. Reay, C. Ramshaw and A. Harvey, *Process Intensification – Engineering for Efficiency: Sustainability and Flexibility* (2008), Elsevier Ltd.

V. R. Regatte and N. S. Kaisare, Propane combustion in non-adiabatic microreactors: 1. Comparison of channel and posted catalytic inserts (2011), *Chemical Engineering Science*, **66** (6), 1123–1131.

T. Ren, Barriers and drivers for process innovation in the petrochemical industry: A case study (2009), *Journal of Engineering and Technology Management*, **26** (4), 285–304.

E. Reverchon, Supercritical fluid extraction and fractionation of essential oils and related products (1997), *The Journal of Supercritical Fluids*, **10** (1), 1–37.

T. Roestenberg, M. J. Glushenkov, A. E. Kronberg, *et al*., Heat transfer study of the pulsed compression reactor (2010), *Chemical Engineering Science*, **65** (1), 88–91.

T. Roestenberg, M. J. Glushenkov, A. E. Kronberg, *et al*., Experimental study and simulation of syngas generation from methane in the Pulsed Compression Reactor (2011), *Fuel*, **90** (5), 1875–1883.

J. P. M. Sanders, J. H. Clark, G. J. Harmsen, *et al*., Process intensification in the future production of base chemicals from biomass (2012), *Chemical Engineering and Processing: Process Intensification*, **51**, 117–136.

S. Sansonetti, S. Curcio, V. Calabrò and G. Iorio, Bio-ethanol production by fermentation of ricotta cheese whey as an effective alternative non-vegetable source (2009), *Biomass & Bioenergy*, **33**, 1687–1692.

S. Sastry and J. T. Barach, Ohmic and inductive heating (2000), *Journal of Food Science*, **65** (s8), 42–46.

S. H. Shuit, Y. T. Ong, K. T. Lee, *et al*., Membrane technology as a promising alternative in biodiesel production: A review (2012), *Biotechnology Advances*, **30** (6), 1364–1380.

M. Simeone, L. Salemme and L. Menna, Methane autothermal reforming in a reverse flow reactor on Rh/Al 2O 3 catalyst (2012), *International Journal of Hydrogen Energy*, **37** (11), 9049–9057.

K. B. Smith and M. R. Mackley, An experimental investigation into the scale-up of oscillatory flow mixing in baffled tubes (2006), *Chemical Engineering Research and Design*, **84** (11), 1001–1011.

V. S. Sutkar and P. R. Gogate, Design aspects of sonochemical reactors: Techniques for understanding cavitational activity distribution and effect of operating parameters (2009), *Chemical Engineering Journal*, **155** (1–2), 26–36.

T. Van Gerven and A. Stankiewicz, Structure, energy, synergy, time the fundamentals of process intensification (2009), *Ind. Eng. Chem. Res.*, **48**, 2465–2474.

G. Vilar, R. A. Williams, M. Wang and R. J. Tweedie, On line analysis of structure of dispersions in an oscillatory baffled reactor using electrical impedance tomography (2008), *Chemical Engineering Journal*, **141** (1–3), 58–66.

G. D. Yadav, Y. B. Jadhav and S. Sengupta, Selectivity engineered phase transfer catalysis in the synthesis of fine chemicals: reactions of p -chloronitrobenzene with sodium sulphide (2003), *Journal of Molecular Catalysis A: Chemical*, **200** (1–2), 117–129.

I. Zbiciński, Equipment, technology, perspectives and modeling of pulse combustion drying (2002), *Chemical Engineering Journal*, **86** (1–2), 33–46.

I. Zbiciński, C. Strumillo, M. Kwapinska and I. Smucerowicz, Calculations of the pulse combustion drying system (2001), *Energy Conversion and Management*, **42** (15–17), 1909–1918.

1
Chemistry for Development

Stephen A. Matlin and Berhanu M. Abegaz

1.1
Chemistry, Innovation and Impact

The foundations of modern chemistry were laid in the 18th and 19th centuries and further extended in the 20th century. They encompassed the development of a theoretical framework for understanding and explaining the physical and chemical properties of atoms and molecules, together with the invention of increasingly sophisticated techniques for interacting with these entities in order to study and influence their structures and behaviors. These developments have given humanity a degree of mastery over its physical environment that surpasses the sum of achievements over the entire previous period of human history.

Chemistry's contributions to human advancement need to be seen in terms of its own core role as a physical science, but also as a "platform science" in the context of its relationships within the group of "natural sciences" that includes physics and biology. Chemistry provides the basis for understanding the atomic and molecular aspects of these disciplines and, through its interfaces with a range of pure and applied sciences, underpins the dramatic advances seen in recent decades in such diverse fields as medicine, genetics, biotechnology, materials and energy. Hence, this discussion of the role of chemistry in the process of development is framed in the broader context of the roles of science, technology and innovation more generally.

Innovation, which may operate in both technological and social fields [1], encompasses not only the birth of an idea or a discovery, but its application in practice – taking the outputs of research and invention and using them to put new goods, services or processes into use. While innovation is sometimes represented as a straightforward linear system (Figure 1.1), in reality this is an over-simplified model and innovation needs to be treated as a complex, highly nonlinear ecosystem, full of interdependences and feedback loops.

The Chemical Element: Chemistry's Contribution to Our Global Future, First Edition.
Edited by Javier Garcia-Martinez, Elena Serrano-Torregrosa

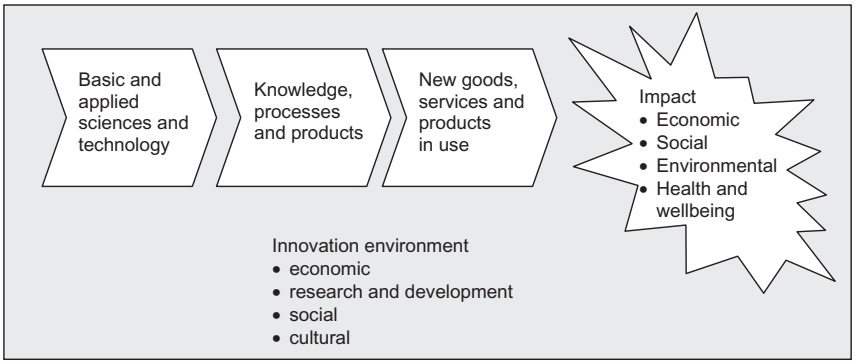


Figure 1.1 The chain of scientific innovation – from ideas to impact.

Chemistry may be involved not only in the initial stages of research (e.g., in areas such as agrochemicals and pharmaceuticals: chemical synthesis of new molecules for testing), but also in intermediate stages (e.g., product development, quality control) and in the evaluation of impact (e.g., health status assessment, environmental monitoring), thus contributing in key ways at every stage of the technological innovation chain.

Throughout the modern period of its development, chemistry has contributed enormously both to broad improvements in human wellbeing (including enhancements of health and quality of life) and to wealth creation for individuals and nations. Some landmark examples are summarized in Table 1.1. Early developments in electrochemistry and synergies with physics and engineering led to methods for producing electrical energy, which has impacted on virtually every aspect of human activity. Electrochemistry also provided the basis for the industrial transformation of many materials and, in particular, for the production of metals such as aluminum and important feedstocks such as caustic soda and chlorine. Industrial organic chemistry built on mid-19th century processes for manufacturing dyestuffs, but by the 20th century had expanded to include the synthesis of pharmaceuticals. In parallel with advances in public health (measures for reducing the spread of infectious diseases through improved water, sanitation and vaccination; and for improving health through ensuring optimal nutrition – in all of which areas chemistry has played a major role), pharmaceutical chemistry has contributed enormously to improving life expectancy and the quality of life through the treatment of infectious diseases and metabolic disorders and the control of pain. Chemistry has contributed to many of the advances in agriculture (e.g., fertilizers, plant growth regulators, pesticides) which have been characterized as a "green revolution" and which have helped to feed the world's population while it grew from about 1 billion to 6 billion during the 20th century. Moreover, chemistry has given the world a wide array of new materials, including polymers, plastics, semiconductors and superconductors, with applications from fabrics and structural materials to information and communications technologies and medical imaging.

Table 1.1 Landmark examples of chemistry breakthroughs contributing to health and wealth.

Date	Scientist	Breakthrough	Impact	Refs
		Industrial chemistry, electrochemistry, power and light		
1800	Alessandro Volta	Discovered that a continuous flow of electricity was generated when using certain fluids as conductors to promote a chemical reaction between the metals or electrodes.	Mass production of portable power sources enabled a vast range of applications– from automobiles to radios. The nickel–metal hydride battery, commercialized in 1990, provided a high energy density and absence of toxic metals. It has found numerous applications include mobile phones and laptop computers	[2]
1802	William Cruickshank	Designed the first electric battery capable of mass production.		
1839	William Grove	Invented the H_2/O_2 fuel cell.		
1859	Gaston Platé	Invented the first rechargeable battery, based on lead-acid chemistry.		
1806	Humphry Davy	Connected a very powerful electric battery to charcoal electrodes and produced "the most brilliant ascending arch of light ever seen".	Invention of the electric light bulb paved the way for replacement of polluting combustion processes for lighting and enabled mass lighting in homes, workplaces and public spaces.	[2]
1879	Thomas Edison	Discovered that a carbon filament in an oxygen-free bulb glowed but did not burn up.		
1820	André-Marie Ampère	Observed that wires carrying an electric current attracted or repelled one another.	Invention of the electric generator transformed life in industrialized countries, impacting on transport, work and leisure.	[2]
1831	Michael Faraday	Demonstrated that a copper disc was able to provide a constant flow of electricity when revolved in a strong magnetic field.		
1800	William Cruickshank	First description of electrolysis of brine	Electrolysis became an extremely important method of transforming materials and especially for the production of inorganic chemicals and compounds, either for use in their own right (e.g., see entry on aluminum below) as a source of feedstocks for the manufacture of other compounds, including organics. For example, the chlorine by-product from the electrolysis of brine was the starting point for the manufacture of organic compounds including solvents, pesticides and plastics	[2–4]
1833	Michael Faraday	Formulation of the laws that govern the electrolysis of aqueous solutions		
1861	Ernest Solvay (1838–1922)	Patented Solvay Process for manufacture of industrial soda using carbon dioxide, brine and ammonia. After the first commercial plant for the electrolysis of brine was built in 1891, caustic soda was increasingly produced directly by this method.		
1897	Herbert Dow	Dow persuaded a group of Cleveland investors to back him in building a chloralkali business in Midland, to be known as The Dow Chemical Co.		

Table 1.1 *Continued*

Date	Scientist	Breakthrough	Impact	Refs
1825	Hans Christian Oersted	Metallic aluminum first made by heating potassium amalgam with aluminum chloride.	Due to its low density, high thermal and electrical conductivity, non-magnetic character, high ductility and the capacity of the metal and its alloys to be cast, rolled, extruded, forged, drawn, and machined, aluminum became one of the most important and ubiquitous metals in the 20th century.	[5] [6]
1886	Charles Hall Paul Heroult	Electrolytic processes for the production of aluminum.		
1840s	August Hoffman	Discovery of aniline and process for its synthesis from benzene	Aniline dyes became the basis for the development of the dyestuffs industry in the 19th century, leading to major growth in chemical industries in the UK, France and Germany.	[7]
1856	William Perkin	Invention of mauve dye (aniline purple).		
		Medicinal chemistry and medicine		
1853	Charles Gerhardt	First synthesis of acetylsalicylic acid	Studies of microorganisms and the physiological effects of chemicals and work on the structural modification of natural products and synthetic chemicals in the 19th century laid the foundations for the pharmaceutical industry in the 20th century. Major classes of therapeutic agents soon emerged, including analgesics, anaesthetics, anti-infectives and anti-tumor agents. Growing understanding of the chemistry of metabolic processes and of the structures and functions of proteins and nucleic acids all contributed to the evolution of pharmacology and molecular biology as distinct sciences and to drug targeting and rational drug design.	[8–18]
1897	Felix Hoffmann	Investigated acetylsalicylic acid as a less-irritating replacement for salicylate medicines, e.g., for treating rheumatism.		
1860–1864	Louis Pasteur	Demonstrated that fermentation is caused by specific microorganisms and formulated the germ theory of disease – providing the basis for biotechnology and anti-microbial chemotherapy.		
1909	Paul Ehrlich	Synthesis of anti-syphilis organo-arsenical drug, Salvarsan.		
1928 1940	Alexander Fleming Howard Florey Ernst Chain	Discovery of penicillin, the first of a family of β-lactam antibiotics, and development of large-scale process for its production		
1932	Gerhard Domagk	Began testing Prontosil, leading to development of the sulfonamide antibiotics	Understanding of the metabolic roles of vitamins and hormones paved the way for a range of drug therapies for metabolic disorders and for development of hormonal contraceptives.	

Table 1.1 *Continued*

Date	Scientist	Breakthrough	Impact	Refs
1865	Heinrich Lissauer	Used potassium arsenite to treat chronic myelogenous leukemia – the first instance of effective chemotherapy for malignant disease. The modern era of cancer chemotherapy began with the study of the cytotoxic effects of nitrogen mustards on lymphoid tissues by Alfred Gilman, Louis Goodman and coworkers in the 1940s.	The pharmaceutical industry now employs well over half a million people and generates global sales in excess of US$ 700 billion per year. In the UK alone, the industry provided employment for 67 000 workers in 2007.	
1912	Casimir Funk	Published the "vitamine theory", based on observations of the effects of depriving animals of small amounts of essential dietary chemicals. Paved the way for the modern understanding of vitamins and their roles as key biochemical catalysts.		
1901	Jokichi	First isolation of epinephrine.		
1915	Takamine	First isolation in crystalline form of		
1921–22	Edward Kendall	thyroxine from thyroid gland. First isolation of insulin and		
1951	Frederick Banting John McLeod Charles Best	demonstration of its capacity to treat diabetes.		
	Carl Djerassi	Synthesized norethindrone, the first effective oral contraceptive.		
1949	Linus Pauling, Harvey Itano, S. J. Singer, Ibert Wells	Publication of "Sickle Cell Anemia, a Molecular Disease" – the first proof of a human disease caused by an abnormal protein and the dawn of molecular genetics.		
1953	James Watson Francis Crick	Discovery of the double helix structure of DNA – the foundation of molecular biology.		
1955	Frederick	First determination of the complete		
1958	Sanger	amino acid sequence of a protein – insulin.		
	Max Perutz John Kendrew	First three-dimensional structures of proteins solved by X-ray crystallography – hemoglobin and myoglobin.		

Table 1.1 *Continued*

Date	Scientist	Breakthrough	Impact	Refs
		Dentistry		
1826	Auguste Taveau	First to use amalgam as a dental restorative material	The development of safe and effective materials for dental restoration and anesthesia transformed dentistry, which had hitherto been a crude and extremely painful procedure.	[19]
1844–1846	Horace Wells William Morton	First uses of nitrous oxide and ether as general anesthetics for dental extractions		
1901	Frederick McKay	Began investigating the cause of widespread brown staining of teeth in Colorado Springs, which he discovered was associated with a dramatic absence of dental caries. Work by chemists in the 1930s eventually traced the cause to fluoride in drinking water.	Water fluoridation and the development of fluoride-containing toothpastes have further contributed to a dramatic improvement in oral health in many countries.	
1905	Alfred Einhorn	First synthesis of procaine (novocaine), a synthetic analog of cocaine without its addictive properties and the first safe local anesthetic for dentistry.		
		Agrochemistry		
1909	Fritz Haber	Invented the Haber process for nitrogen fixation, later scaled up by Carl Bosch. Fixation of nitrogen as ammonia, which can then be oxidized to make nitrates and nitrites, made possible the industrial production of many classes of compounds including nitrate fertilizers and explosives.	The Haber process now produces 100 million tons of nitrogen fertilizer per year, consuming 3–5% of world natural gas production (ca. 1–2% of the world's annual energy supply) and generating fertilizer which is responsible for sustaining one-third of the Earth's population.	[20] [21] [22] [23] [24] [25]
1874	Othmar Zeidler	First synthesis of DDT.	The development of plant growth promoters, crop protection agents and agents promoting animal health contributed to an agrochemicals industry with global annual sales of over of US$ 100 billion and, together with advances in plant breeding and agronomy methods, produced the green revolution of the 1960s–1980s which has helped feed the burgeoning population of the planet.	
1939	Paul Müller	Insecticidal properties discovered. DDT became the first commercial organochlorine insecticide. Prior to its banning on environmental grounds, DDT was a major weapon in the fight to eliminate malaria.		
1951	Geigy Chemical Co.	Introduction of carbamate insecticides		
1960s	Michael Elliott	Development of synthetic pyrethroid insecticides.		

Table 1.1 *Continued*

Date	Scientist	Breakthrough	Impact	Refs
		Analytical chemistry		
1901	Michael Tswett	First use of an adsorption column for the separation of plant pigments marked the birth of chromatography–later to develop into a family of 2- and 3-dimensional techniques involving combinations of gas, liquid and solid phases, for analytical and preparative scale separation of compounds.	The pioneering studies by a range of scientists, including botanists, physicists and physical chemists, led to the development of extremely powerful sets of techniques for separating chemical species, identifying them and measuring their concentrations. The evolving and often intertwined fields of analytical and separation sciences have been of fundamental importance, not only to the advance of chemistry itself but also to a wide range of areas including clinical and environmental sciences.	[26] [27] [28] [29] [30] [31]
1800	William Herschel	Discovered infrared radiation.		
1854	August Beer	Extending earlier work by by Pierre Bouguer and Johann Lambert, published what became known as the Beer-Lambert Law, defining the relationship between the extent of absorption of light and the properties of the material through which it is traveling.		
1859	Robert Bunsen Gustav Kirchoff	Developed the first spectroscope.		
1895	Wilhelm Röntgen	First systematic studies of X-rays, which later became the basis of X-ray medical diagnosis and X-ray crystallography.		
1913	Joseph Thomson	Invented mass spectrometry.		
1938	Isidor Rabi	First described and measured nuclear magnetic resonance in molecular beams. NMR became the basis of techniques for molecular structure elucidation and also medical imaging.		

Table 1.1 *Continued*

Date	Scientist	Breakthrough	Impact	Refs
		Polymers and plastics		
1839	Charles Goodyear	Discovered the process of vulcanization of natural rubber by heating with sulfur	Materials based on synthetic plastics and polymers became ubiquitous in the 20th century, finding applications in clothing, products from containers and appliance casings to non-stick pans, thermal and electrical insulators, components of transport machinery, medical and surgical devices, in space exploration and in much else.	[32] [33] [34] [35] [36]
1855	Alexander Parkes	Created the first plastic by treating cellulose treated with nitric acid.		
1909	Leo Hendrik Baekeland	Bakelite, the first synthetic polymer plastic, made from phenol and formaldehyde.		
1934	Wallace Carothers	First synthesis of a synthetic fiber, nylon, by co-polymerization of hexamethylene diamine and adipic acid		
		Solid state chemistry		
1833	Michael Faraday	Described first semiconductor effect, noting that electrical conductivity in silver sulfide increased with increasing temperature	Semiconductors based on silicon became the basis of solid state electronic devices including computers and provided the foundation for the modern digital age.	[37]

The value added by these products of chemistry and related sciences has contributed to the rapid growth in world GDP [38], especially in the industrially advanced countries during the second half of the 20th century (Figure 1.2). Knowledge-intensive and technology-intensive industries are estimated [39] to have accounted for 30% of global economic output, or some US$15.7 trillion, in 2007.

1.2 Poverty and Disparities in Life Expectancy

The benefits from advances in chemistry and other sciences have not been evenly distributed globally. The least industrially/technologically advanced countries have remained the poorest and people in the low- and middle-income countries (LMICs) have often fared worse than those in high-income countries (HICs), as illustrated by the dramatic relationship between poverty and life expectancy: the poor die young. Life expectancies around the world have increased very markedly over the course of the last century, but as they have done so the disparities between populations have grown larger [40]. However, the relationship between life expectancy

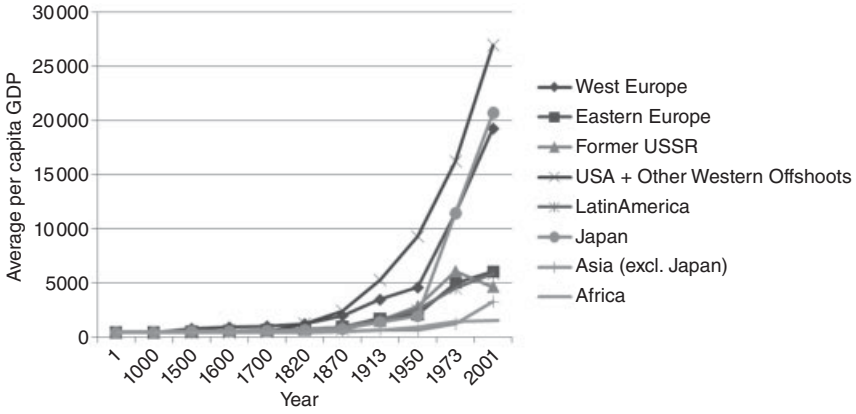


Figure 1.2 Per capita GDP: Regional and world averages, 1–2001 AD (millions 1990 international Geary-Khamis dollars). Data from [38], Table 8c.

and the average per capita income of the country is not a straightforward one and income is not the only factor involved. The economist Easterlin [41] concluded that much of the decline in mortality in the 20th century had its origin in technical progress – and in this context, "technical progress" refers to a combination of technological advances and their diffusion and uptake in different countries and the capacities of the countries themselves to conduct and apply research. Much of the variation in life expectancies seen between countries is explained by differences in the rate of this technical progress – for example, it explains two thirds of the variation in the decline in infant mortality over a 25 year period, whereas change in income explains only 9% [42, 43].

1.3 The Millennium Development Goals

In response to the unacceptable levels of poverty (Box 1.1) and growing disparities in health and wellbeing between people in different countries, the world's

Box 1.1 Poverty

What is poverty?

Poverty is hunger. Poverty is lack of shelter. Poverty is being sick and not being able to see a doctor. Poverty is not having access to school and not knowing how to read. Poverty is not having a job, is fear for the future, living one day at a time. Poverty is losing a child to illness brought about by unclean water. Poverty is powerlessness, lack of representation and freedom.

The World Bank [46]

governments met at the Millennium Summit [44] in New York on 6–8 September 2000, issuing the Millennium Declaration which led to agreement on a series of Millennium Development Goals (MDGs) [45] that were set for 2015 (Table 1.2). The targets were acknowledged to be extremely ambitious – but it was recognized that, for the first time in history, mankind had the capacity to substantially reduce or eliminate many sources of human suffering and to offer every person on the planet a basic level of existence that would be free from hunger, disease and discrimination in access to opportunities for development.

As stated in the report of the Task Force on Science Technology and Innovation of the Millennium Project [47]:

> "Since their adoption at the United Nations Millennium Summit in 2000, the Millennium Development Goals have become the international standard of reference for measuring and tracking improvements in the human condition in developing countries. The Goals are backed by a political mandate agreed by the leaders of all UN member states. They offer a comprehensive and multidimensional development framework and set clear quantifiable targets to be achieved by 2015."

The latest assessment shows that uneven progress has been made towards meeting the targets. Unmet commitments, inadequate resources, lack of focus and accountability and insufficient dedication to sustainable development have created shortfalls in many areas and without a major push forward many of the MDG targets are likely to be missed in most regions [48].

To achieve the goals will require a collective global effort harnessing political will, available resources and innovation in all areas, including the application of science and technology.

1.3.1 Goal 1: Reducing Poverty and Hunger

Economic growth, especially in the world's most populace country, China, resulted in hundreds of millions of people being lifted out of poverty during the last quarter of the 20th century [46]. Nevertheless, at the end of the century, out of a global population of 6 billion there were more than one billion people living on less than $1 a day, more than three billion living on less than $2 a day and nearly a billion suffering from hunger or severe malnutrition.

While many economically advanced countries produce an excess of food, some of which goes to waste, halving the proportions of those suffering poverty or hunger by 2015 is not merely a matter of redistributing available food. To overcome the net food shortage, allow for the expanding world population (already approaching 7 billion by 2010), ensure food security and independence from aid handouts, and respond to the agricultural impacts of climate change, it is necessary to expand agriculture throughout the world. Better applications of existing technologies and development of innovative new ones are essential

Table 1.2 Millennium development goals.

Goals	Targets
Goal 1 Eradicate extreme poverty and hunger	Halve, between 1990 and 2015, the proportion of people whose income is less than $1 a day
	Halve, between 1990 and 2015, the proportion of people who suffer from hunger
Goal 2 Achieve universal primary education	Ensure that, by 2015, children everywhere, boys and girls alike, will be able to complete a full course of primary schooling
Goal 3 Promote gender equality and empower women	Eliminate gender disparity in primary and secondary education preferably by 2005 and in all levels of education no later than 2015
Goal 4 Reduce child mortality	Reduce by two-thirds, between 1990 and 2015, the under-five mortality rate
Goal 5 Improve maternal health	Reduce by three-quarters, between 1990 and 2015, the maternal mortality ratio
Goal 6 Combat HIV/AIDS, malaria and other diseases	Have halted by 2015 and begun to reverse the spread of HIV/AIDS
	Have halted by 2015 and begun to reverse the incidence of malaria and other major diseases
Goal 7 Ensure environmental sustainability	Integrate the principles of sustainable development into country policies and programmes and reverse the loss of environmental resources
	Halve, by 2015, the proportion of people without sustainable access to safe drinking water and basic sanitation
	Have achieved, by 2020, a significant improvement in the lives of at least 100 million slum dwellers
Goal 8 Develop a global partnership for development	Develop further an open, rule-based, predictable, non-discriminatory trading and financial system (includes a commitment to good governance, development, and poverty reduction – both nationally and internationally)
	Address the special needs of the least developed countries (includes tariff-and quota-free access for exports, enhanced program of debt relief for HIPC and cancellation of official bilateral debt, and more generous ODA for countries committed to poverty reduction)
	Address the special needs of landlocked countries and small island developing states (through the Program of Action for the Sustainable Development of Small Island Developing States and 22nd General Assembly provisions)
	Deal comprehensively with the debt problems of developing countries through national and international measures in order to make debt sustainable in the long term
	In cooperation with developing countries, develop and implement strategies for decent and productive work for youth
	In cooperation with pharmaceutical companies, provide access to affordable, essential drugs in developing countries
	In cooperation with the private sector, make available the benefits of new technologies, especially information and communications

[49]–amounting to a second "green revolution" in which chemistry must play multiple important roles. Critical areas include improving plant varieties and methods for the efficient production, processing and preservation of foods that are healthy and nutritious.

The poverty goal is often referred to as an overarching goal, as it is intimately associated with the problems that are tackled in the other goals, including lack of gender equality, poor education and illiteracy, unacceptably high rates of maternal, neonatal and child mortality and of deaths from infectious diseases, lack of access to improved water and sanitation, and poor environment. However, it would be wrong to focus excessively on achieving this goal in the hope that the others will be met as a consequence. Other areas, such as education and health, have also been stressed as fundamental enablers of progress and the barriers to reaching each goal are varied and complex in nature, requiring individual attention. The reality is that effort is necessary across the whole range of issues highlighted in the MDGs. It must also be stressed that the MDGs are by no means comprehensive or complete, if a permanent shift in the trajectory of human development is to be achieved–for example, the MDGs make no direct reference to overcoming the challenges of unmet needs for reproductive health or the burgeoning rates of non-communicable diseases in LMICs.

1.3.2
Goal 2: Achieving Universal Primary Education

Education is a fundamental enabler of many other aspects of human development. Access to literacy, numeracy and knowledge transforms the lives of individuals, leading to better health and enhancing economic and social advancement, as well as contributing to national economic development. Yet, in the year 2000, there were more than 110 million children of primary school age out of school [50], a very low standard of education available for many children officially attending schools in some countries, and many hundreds of millions of adults who were illiterate. Moreover, education has exhibited a very high degree of gender discrimination, with a majority of those lacking access being girls and women. The high importance of education warrants the goal of ensuring universal primary schooling as a first step towards enabling access to secondary and further education.

1.3.3
Goal 3: Promoting Gender Equality and Empowering Women

Discrimination in access to education and health services and economic, social and political opportunities is experienced by girls and women in every part of the world. The fundamental right of females to equality of status, opportunity and treatment in all areas of human endeavor was established by a series of UN conventions and intergovernmental declarations during the 20th century [51–54], but at the close of the millennium the reality of women's and girls' experiences fell very far short of these standards in many parts of the world, and especially in many

low- and middle-income countries. The MDGs set specific targets for moving towards gender equity, to help drive the process forward. Among the areas highlighted for urgent action was access to education – and it is notable that, even for girls enrolled in education, they often experience barriers in access to science and technology education in many countries [55].

1.3.4
Goals 4 and 5: Reducing Maternal and Under-Five Child Mortality

The chances of a woman dying during pregnancy or childbirth or in the immediate post-partum period, or of a child dying in the first few years of life, can be a hundred-fold greater in some of the world's poorest countries than in some of the richest. The causes, which link to poverty, poor nutrition, lack of education and inadequate availability of and access to effective health services including emergency obstetric care, may be varied. However, they are well understood and it is unacceptable in the 21st century that women should continue to die in large numbers simply because they become pregnant or that infants should die because they are not provided with the means of survival. The latest assessments [56] show some progress, but maternal and child mortality levels in many LMICs remain unacceptably high (Figure 1.3) and many countries are still off track to meet the MDG targets.

1.3.5
Goal 6: Combating HIV/AIDS, Malaria and Other Diseases

The development of antibiotics and vaccines has contributed to a massive reduction in mortality and morbidity due to infectious diseases in high-income countries during the last hundred years. However, many LMICs continued to experience major problems with communicable diseases – especially those caused by tropical parasitic infections such as malaria, leishmaniasis, trypanosomiasis, schistosomiasis and Guinea Worm. The advent of the HIV/AIDS epidemic, which began spreading rapidly in countries in Africa and elsewhere in the 1980s and 1990s, transformed the situation into one of crisis, compounded by the concomitant resurgence of tuberculosis in increasingly drug-resistant forms. Meeting the targets for halting and rolling back the spread of these diseases requires not just better access to existing technologies but also, in many cases, innovations in the form of new diagnostics, drugs, vaccines and delivery systems – all areas where chemical sciences must make a core contribution.

1.3.6
Goal 7: Ensuring Environmental Sustainability

The broad concept of "sustainable development" – recognizing the finite nature of the world's physical and biological resources and the importance of protecting and preserving them while engaging in human activity on the planet – emerged during

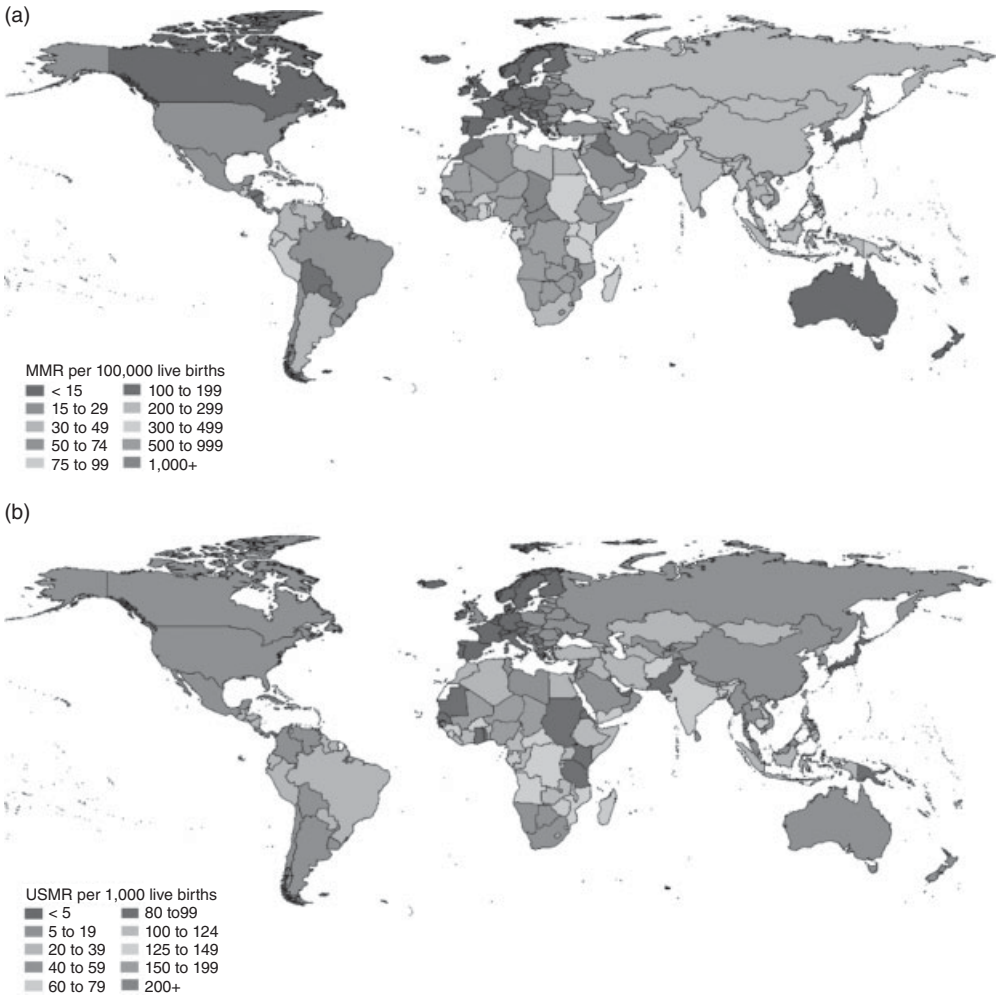


Figure 1.3 (a) Maternal mortality ratio (MMR) by country (per 100 000 live births), 2008 and (b) under-5 mortality rate (U5MR) by country (per 1000 live births), 2008 (from [56]). (Please find a color version of this figure in the color plates.)

the second half of the last century and was the focus of attention in world summits in Rio de Janeiro in 1992 [57] and in Johannesburg in 2002 [58]. The MDG targets represent an attempt to make some headway with these extremely challenging problems.

While there was still dispute at the end of the 20th century about the degree of climate change that the world would experience, there is now conclusive evidence that global warming is a real phenomenon and that climate change is already having, and will continue to have, increasingly severe impacts on many aspects of the human condition, including health, agriculture, the availability of fresh water,

human habitation and, especially for some low-lying countries, even their very existence [59].

A further important aspect of the changing human environment has been a major shift during the last century from rural to urban dwelling. In 2007, for the first time, the proportion of human beings living in urban dwellings reached 50% and the transition continues [60]. Since most of the increase in the world population expected to occur during the next half century (from 6 billion in 2000 to 9 billion in 2050) will take place in LMICs, and since cities in many of these countries already have a high proportion of their inhabitants living in slum conditions without adequate water or sanitation, the challenges for city planners and technologists are enormous.

1.3.7
Goal 8: Developing a Global Partnership for Development

Globalization (the increasingly rapid and less restricted movement of people, goods, services and information around the world) brings with it a growing global interdependence of people and economies. This has resulted in a pressing need for global systems governing a wide range of human activities that impact on health, trade, the environment and much else. The eighth MDG calls for effective global partnerships among all the relevant actors to address a range of concerns that were seen to be important at the opening of the new millennium, including the rules governing access to health technologies and to information and communications technologies.

One aspect of health technology that has attracted much attention has been the issue of how the rules governing intellectual property rights should be applied justly and humanely in the field of medicine, considering the high costs of anti-retroviral drugs for the treatment of people living with HIV/AIDS and other life-threatening diseases [61]. The eighth MDG looks to governments and pharmaceutical companies to cooperate in providing access to affordable essential drugs in LMICs.

1.4
Science, Technology and Development

Advances in science and technology (S&T) enabled countries in Europe and North America to industrialize rapidly in the 19th and 20th centuries. For example, industrialization in Belgium drew on the Solvay process for manufacture of soda, which helped to establish Belgium as one of the world's leading countries in the chemical industry sector (Box 1.2).

While this process in Europe and North America was under way, from as early as the end of the 19th century a number of less developed countries were beginning to recognize the importance of S&T, either for economic growth or to address serious health challenges such as epidemics. Some notable examples include:

Box 1.2 Chemistry and the industrialization of Belgium

The industrial revolution began in Belgium with the development of British-style machine shops at Liège (*ca.* 1807), and Belgium became the first country in continental Europe to be transformed economically. Like its English counterpart, the Belgian industrial revolution initially centered on iron, coal and textiles. During the 19th century the chemical industry added momentum to Belgium's industrial and economic development.

Ernest Solvay (1838–1922) developed a process for manufacturing industrial soda (sodium carbonate), in which carbon dioxide (from limestone) is mixed with sodium chloride solution and ammonia. The Solvay process, which was cheaper and more efficient than the old Leblanc process, was patented in 1861 and Solvay opened his first factory at Couillet in Belgium in 1863. Industrial soda is used in manufacturing glass, steel and detergents and demand was huge. By 1900, the Solvay process was used to manufacture 95% of the world's industrial soda, and Solvay had an extensive business empire, with factories in Europe and the USA. Today, around 70 plants using the Solvay process are in operation around the world.

As demand increased, in 1898 Solvay started producing caustic soda directly by the electrolysis of brine, a process which also yields chlorine and hydrogen and gradually replaced the older method. The chlorine was used to produce hydrochloric acid, sodium hypochlorite and, later, organic derivatives including vinyl chloride (the monomeric precursor of PVC plastics); while the hydrogen was oxidized to hydrogen peroxide. Solvay is now the world's leading producer of peroxygen-based products, with a global network of peroxygen production plants.

In 2007, on a per capita basis Belgium was the number one producer of chemicals in the world and the share of the chemical industry in Belgium's economy was even bigger than that in Germany [62, 63].

- A number of research institutions established branches or offshoots in less industrialized countries. In particular, France's Institut Pasteur, created in 1887 for the prevention and treatment of infectious diseases through research, teaching and public health initiatives, established an international network which currently counts 30 members spread over the five continents [64]. Early members of the network included Pasteur institutes in Vietnam (1891), Tunisia (1893), Algeria (1894), Madagascar (1898), Morocco (1911), Iran (1920), Senegal (1923) and French Guiana (1940).
- Conceived in 1896 by Jamsetji Nusserwanji Tata and finally born in 1909, the Indian Institute of Science was an early example of a research institute established in the British colonial period [65]. The South African Chemical Institute [66] was founded in 1912.
- Brazil's Federal Seropathy Institute, established in 1900 to produce serums and vaccines against the plague, in 1908 became the Oswaldo Cruz Institute and later the Oswaldo Cruz Foundation [67]. It has made huge contributions to combating disease, including through the production of medicines by its Farmanguinhos branch [68].

- Rubber Research Institutes were established to support the expanding demand for rubber products. For example, the origin of research on rubber in Sri Lanka dates from 1909, when a group of planters in the Kalutara District met and agreed to engage a chemist to study the coagulation of rubber [69]. Similarly, Rubber Research Institutes emerged in other rubber-growing countries, including Nigeria [70] (1900), Malaya [71] (1925) and India [72] (1946).

Since the mid-20th century, the importance of science and technology for development has increasingly been recognized by international agencies [73–75], development assistance partners [76] and the governments of LMICs [77–79].

Within the UN family, UNESCO is the UN specialized agency mandated to build institutional and human capacity in the basic and engineering sciences, which are seen as a prerequisite for social and economic development. UNESCO's activities focus principally on third-level, but also second-level, education and on research in mathematics, physics, chemistry, biology, biotechnology and basic medical sciences [80]. The UNESCO Science Prize is awarded biennially to "a person or group of persons for an outstanding contribution they have made to the technological development of a developing member state or region through the application of scientific and technological research (particularly in the fields of education, engineering and industrial development)." The first prize was awarded to Robert Simpson Silver in 1968 for his discovery of a process for the demineralization of sea water; several subsequent prizes have also been chemistry-related [81]. A partnership between UNESCO and L'Oréal, *Awards For Women in Science*, forms a core element of UNESCO national and international activities to foster gender equality and equity in science [80].

The work of UNESCO is reinforced by the recognition by the UN Development Programme (UNDP) of the importance of technology for the progress of the least developed countries [82].

A number of nongovernmental organizations (NGOs) have been established to promote the roles in development of S&T in general. Some notable examples include:

- **Academy of Sciences for the Developing World (**TWAS: **originally known as the** Third World Academy of Sciences**):** TWAS is an international NGO founded in 1983 in Trieste, Italy by a distinguished group of scientists from the South under the leadership of the late Nobel laureate Abdus Salam of Pakistan. Its principal aim is to promote scientific excellence and capacity in the South for science-based sustainable development [83].

- Third World Organization for Women in Science **(**TWOWS**):** Established as an NGO in Trieste in 1989, TWOWS is the first international forum to unite eminent women scientists from the South with the objective of strengthening their role in the development process and promoting their representation in scientific and technological leadership [84].

- International Association of Science and Technology for Development **(**IASTED**):** A non-profit organization devoted to promoting economic

and cultural advancement, IASTED was established in 1977. It organizes multidisciplinary conferences for academics and professionals, in both industrialized and developing countries, mainly in the fields of engineering, science, and education [85].

Perspectives on the nature of the development process itself have changed markedly during the last half-century. In the period of the 1950s–1970s, on the HIC side much of the development process was driven by and centered around post-colonial relationships and geopolitical cold-war maneuvrings, while LMICs themselves were beginning to seek ways to develop their own resources and capacities. There has been a movement away from HIC-driven approaches towards national self-determination and South–South cooperation and mutual reinforcement. Gradually there was a shift from a utilitarian perspective, which primarily focused on economic advancement as the main goal and saw human resources development, including S&T capacities, as a means of achieving this, to a human rights perspective which saw human development, equity and well-being as the primary objectives, with economic development being an important mechanism for enabling all people to achieve certain standards of health and freedom from want of basic needs as an inalienable right.

A series of world conferences in the 1990s, covering education, health, population, and sustainable development, culminated in the Millennium Declaration [86] in 2000. Reflecting the shifts in approach, the work of the Commission on Macroeconomics and Health, which reported to the World Health Organization in 2001, emphasized that health is an essential prerequisite for development, rather than the converse [87].

The role of science and innovation as drivers of development was examined in detail by the Task Force on Science, Technology and Innovation of the UN Millennium Project [47]. The Task Force, led by Calestous Juma and Lee Yee-Cheong, identified the important roles that science and technology can play in achieving the MDGs. It stressed the importance of S&T policies tailored to the specific needs and circumstances of each country and the need to create international partnerships that allow mutual learning.

The report of the Task Force outlined key areas for policy action, including:

- focusing on platform (generic) technologies
- improving infrastructure services as a foundation for technology
- improving higher education in science and engineering and redefining the role of universities
- promoting business activities in science, technology, and innovation
- improving the policy environment
- focusing on areas of underfunded research for development.

In a further study [88], Juma proposed that international development policy should be directed at building technical competence in developing countries rather than conventional relief activities. He argued that institutions of higher learning, especially universities, should have a direct role in helping to solve development challenges [89].

1.5
Chemistry and Development

Within the broader domain of S&T, chemistry has emerged as a key discipline able to contribute to development [90]. A number of NGOs, as well as programs within existing bodies, have been established to promote this contribution internationally. Two, in particular, are notable as examples of efforts at the global level to address major development needs and to build capacities for relevant chemistry in LMICs.

1.5.1
Chemical Research Applied to World Needs

At its meeting in Munich in 1973, the International Union of Pure and Applied Chemistry (IUPAC) considered ways in which it could foster opportunities for international cooperation. The result was the establishment of Chemical Research Applied to World Needs (CHEMRAWN) as a mechanism through which member nations of IUPAC could aid in identifying and solving important chemistry problems that have a direct impact on world needs [91]. The initial purposes proposed for CHEMRAWN were:

1) To identify human needs amenable to solution through chemistry with particular attention to those areas of global or multinational interest.

2) To serve as an international body and forum for the gathering, discussion, advancement and dissemination of chemical knowledge deemed useful for the improvement of humankind and our environment.

3) To serve as an international, nongovernmental source of advice for the benefit of governments and international agencies with respect to chemistry and its application to human needs.

The major activity of the CHEMRAWN Committee has been to organize a series of conferences, designed to identify and focus attention on world needs and to make recommendations for action to the global scientific community [92]. The highly ambitious nature of these conferences envisaged (Box 1.3) at the outset of CHEMRAWN illustrates the complexity and the importance of the potential roles of chemistry in development. In particular, four key elements remain the bedrock of achieving chemistry's potential in development almost four decades after the vision was first enunciated:

- A systems approach is essential to understanding and responding to human needs.
- Many interlocking systems are involved, requiring approaches that cross boundaries between S&T disciplines and social, economic, environmental and political sectors as well as needing engagement between governments, industries and academia.

Box 1.3 Chemical Research Applied to World Needs (CHEMRAWN)

"It was envisioned that CHEMRAWN activities would provide the basis for treating chemical-based human needs as systems. Thus, CHEMRAWN conferences by their very nature would be highly interdisciplinary and would take into account the social, economic, environmental and political factors, as well as the technical components involved. It was planned that these international conferences would attract world leaders from governments, industries and academia, and that the goal and focal point of the conference activities would be an attempt by recognized and influential world leaders to take an initial step toward developing a sense of future direction that would be of value to the world chemical community. Such direction would be provided in recommendations set forth in conference proceedings and made available to participants and policymakers and governments, industries, and academic institutions worldwide. Further, it was determined that CHEMRAWN conferences would provide continuity in areas where there is a persistent need."

Bryant Rossiter, first Chair of the CHEMRAWN Committee, quoted in [92].

- Engagement with politicians and the creation of a supportive policy environment are essential for advancing and sustaining the development agenda.
- All countries, including the less economically advanced, can contribute to the development process.

1.5.2 International Organization for Chemical Sciences in Development

The International Organization for Chemical Sciences in Development (IOCD) was the first international NGO specifically devoted to enhancing the role of the chemical sciences in the development process and involving chemists working in LMICs [93–96]. Its origins lay in a program established by the Special Programme of Research, Development and Research Training in Human Reproduction (HRP) at the World Health Organization (WHO) in the 1970s. Since many contraceptives appropriate for use in LMICs were not of major interest to the pharmaceutical companies whose markets were mainly in HICs, HRP-WHO sponsored a program to develop novel contraceptives outside the traditional pharmaceutical industry channels. In a project coordinated by the Belgian chemist Pierre Crabbé, the skills of groups of chemists in LMICs were engaged to synthesize compounds for biological evaluation. Over a number of years, several hundred novel steroids were synthesized, formulated and tested [97–99]. The success of this program [96] led to the idea that it might serve as a model for developing other drugs or even pesticides, while simultaneously stimulating capacity building in LMICs and enabling chemists in these countries to contribute to key S&T areas for development [100].

Building on this idea, Crabbé invited distinguished scientists from more than a dozen countries to meet at UNESCO, Paris in 1981, to consider how to give sustained support to the research work of chemists in developing countries. They recognized that many barriers hinder the efforts by scientists in LMICs to carry

out research, including inadequate laboratory equipment, lack of up-to-date books and journals, long periods of isolation from mainstream scientific activities, and so on. The vision of how these barriers might be lowered was to engage scientists from LMICs in collaborative research with scientists from HICs. IOCD was established to take forward the model [101]. Initially housed at UNESCO in Paris, IOCD soon moved to Mexico City, where it was given support by the Ministry of Health. The first group of elected officers were Glenn Seaborg (Nobel Laureate chemist, Berkely University, USA) as President; C.N.R. Rao, (Head of the Indian Institute of Science, Bangalore, India) and Sune Bergström (Nobel Laureate chemist, Karolinska Institute, Sweden) as Vice Presidents; and Elkan Blout, (Dean of the Harvard School of Public Health, USA) also as a Vice President and Treasurer. The involvement of these eminent scientists and of a range of other high-profile scientists from LMICs and HICs in the IOCD Advisory Council (including the father of the "green revolution", Nobel Laureate Norman Borlaug), was important in the early years in securing funding from a range of international organizations and foundations and in attracting prominent scientists to serve as leaders of IOCD's scientific Working Groups (Figure 1.4).

The first two IOCD Working Groups were aimed at the development of compounds for male fertility regulation and to treat tropical diseases. Modest grants were provided to facilitate the purchase of laboratory supplies and support research students in the collaborating LMIC laboratories, with collaborators in HIC laboratories assisting to overcome barriers to supply and providing back-up such as advanced spectroscopic and analytical services. While the work inevitably proceeded slowly in the first few years, a key spin-off was the establishment of networks of chemists collaborating across countries and many of the contacts and collaborations survived long after the projects themselves came to an end. IOCD was able to sponsor a number of site visits and training exchanges and a key event was a meeting of all the scientists involved in the IOCD programs in Oaxtepec, Mexico in 1986.

Figure 1.4 IOCD scientists meeting at Berkeley, California in 1986. From left to right: Carlos Rius, IOCD's first secretary; Pierre Crabbé, founder; Elkan Blout, first treasurer and one of three founding vice presidents; Carl Djerassi, one of the inspirations behind IOCD; Sune Bergström, a founding vice president; Sydney Archer, leader of the Tropical Diseases Working Group; (unknown); Glenn Seaborg, IOCD's first president and associate director of the Lawrence Berkeley National Laboratory; C.N.R. Rao, a founding IOCD vice president; and Joseph Fried, leader of the Male Fertility Regulation Working Group. (Please find a color version of this figure in the color plates.)

Pierre Crabbé was tragically killed in a car accident in 1987, but under its new Executive Secretary, Robert Maybury, IOCD continued to work and to grow, adding additional working groups on plant chemistry and on environmental analytical chemistry, and later a group on bioprospecting. The emphasis has gradually shifted away from active project funding for chemistry research programs to capacity building activities through organizing training workshops, supporting attendance at scientific meetings and supporting the networking efforts of scientists in Africa. Most recently, IOCD has adopted two long-term projects to help reinforce scientific capacities in LMICs: one on Books for International Development, which organizes the collection and transfer of books and journals to developing countries, with each shipment containing tens of thousands of items; and one on micro-scale chemistry, which helps support an international program that provides low-cost, small-scale equipment to enable students to gain hands-on practical skills in experimental chemistry even in very resource-poor settings [101].

1.6
Science and Technology for National Development

1.6.1
Investments in Research and Development

From 2002 to 2007, world R&D expenditure increased by 44%, from an estimated 788.5 billion PPP$ (purchasing power parity dollars) to 1137.9 billion PPP$. In relative terms, 1.7% of the world's Gross Domestic Product (GDP) was devoted to R&D in 2007 [55]. A number of Asian economies, including those of the Republic of Korea, Taiwan, Hong Kong and Singapore, became highly successful during the last half century and were able to maintain growth rates of 8–10% over a number of years. While many factors have been considered to contribute to the success of these "Asian tigers", important common threads have been an emphasis on higher education and on balanced investments across a range of business and technology sectors. This has enabled the economies to grow rapidly and to shift away from dependence on the export of raw materials and primary products, towards the production of high value-added products of S&T.

Given the importance of S&T for economic advancement and competitiveness in all countries [102], it is not surprising that, in recent years, there has been an increased focus on targeting specific levels of national investment in research and development (R&D) as a key driver of innovation. In 2002, the European Union (EU) set a target of reaching a level of 3% gross expenditure on R&D (GERD: also known as "research intensity") as a percentage of GDP by 2010. Of this 3%, it was projected that one third would come from public sector investment and two thirds from the private sector. By 2007, only Finland and Sweden had passed the 3% target, Austria, Denmark and Germany had reached 2.5% and France had reached 2%, while ten of the 27 EU member states were still investing below 1% [103, 104] (Figure 1.5a).

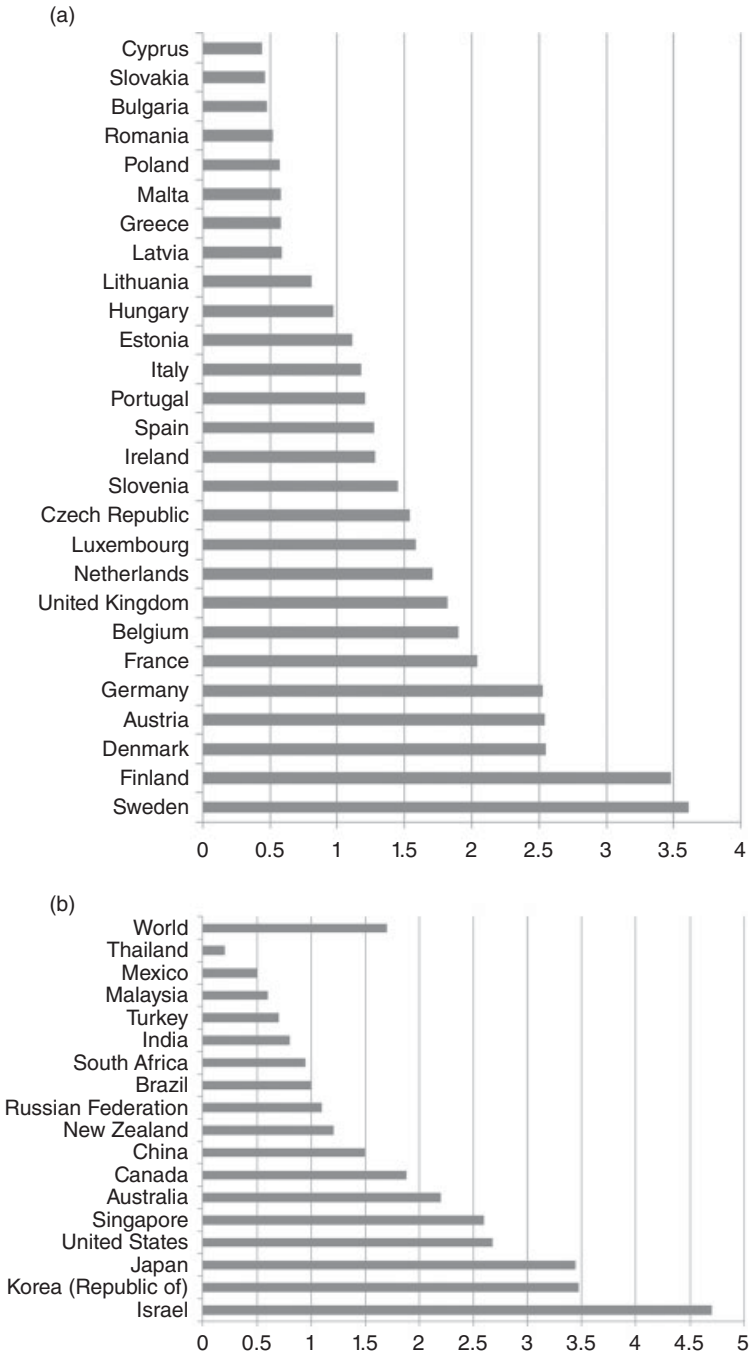
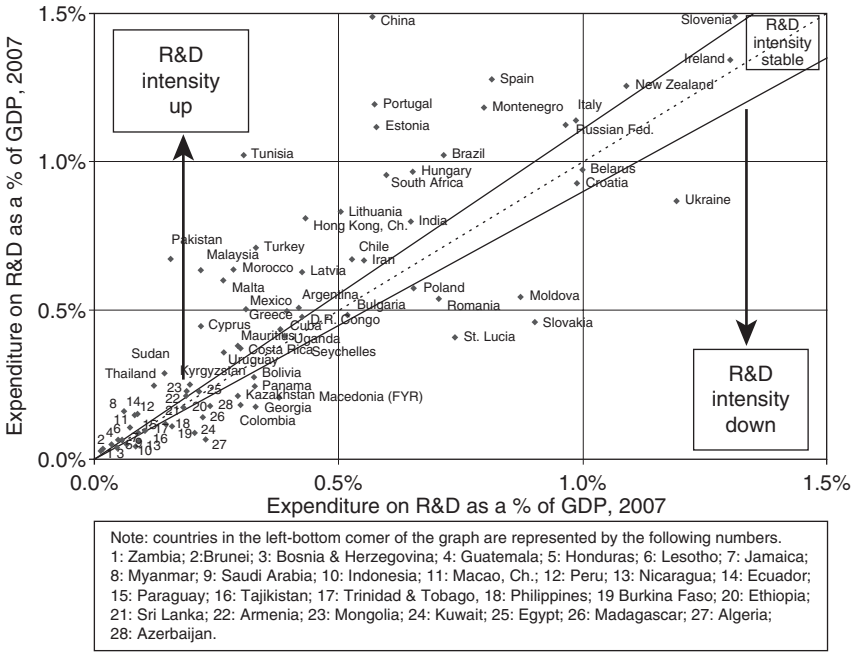


Figure 1.5 Gross domestic expenditure on R&D as a percentage of GDP for 2007. (a) European Union countries (data from [104]), (b) other countries (data from [104]).

Outside the European Union, gross expenditure on R&D among economically advanced countries and emerging economies also varies widely (Figure 1.5b):

- Recognizing the weakness of its performance in science, technology and innovation, the African Union (AU) has initiated efforts to increase its investments in R&D, spearheaded by the New Partnership for Africa's Development (NEPAD). At the first NEPAD Ministerial Conference on Science and Technology in November 2003, Ministers of Science and Technology of 20 AU countries reaffirmed their commitment to increasing public spending on R&D to at least 1% of GDP within five years and the AU commitment to this 1% target has been reiterated on a number of occasions and member countries are still working towards it [103, 105]. To date, only South Africa regularly measures and reports data on its research intensity, which had risen to 0.95% by 2007 – of which 56% came from the business enterprise sector [106]. Data on R&D investments in other African countries appears only sporadically and, for the few countries where information is available, suggests a range from <0.1% (Algeria) to 1% (Tunisia) in North Africa and 0.5% (Mozambique) or less in the rest of Africa. *Africa's Science and Technology Consolidated Plan of Action 2006-2010* was first elaborated in 2005 by the African Union/NEPAD and is being implemented with assistance from UNESCO, which has adopted three flagship projects: (i) capacity building in S&T and innovation policy; (ii) enhancing science and technology education; and (iii) the African Virtual Campus. NEPAD has instituted the African Science, Technology & Innovation Indicators Initiative (ASTII) and the establishment of the African Observatory for Science, Technology and Innovation (AOSTI). ASTII aims at the development and adoption of African common science, technology and innovation indicators, while AOSTI will ensure that the STI indicators and information gathering as well as collation, compilation and validation are standardized [107].
- Among the emerging economies, China has demonstrated a dramatic rate of increase in GERD, which almost tripled to 1.5% between 1996 and 2007 [55] (Figure 1.6).
- Israel, Japan and Korea all invest more than 3% of GDP in R&D. The USA has long recognized the strategic economic importance of investing in R&D but its research intensity has not kept pace with this leading group. In April 2009, President Barack Obama announced that the USA will devote more than 3% of its GDP to R&D, with policies that invest in basic and applied research, create new incentives for private innovation, promote breakthroughs in energy and medicine, and improve education in math and science. This represents the largest commitment to scientific research and innovation in American history [108].

The term "Innovative Developing Countries" (IDCs) has begun to be used to describe a number of countries which have been making strong advances in strengthening their S&T to support their own development. These include Argentina, Brazil, China, India, Indonesia, Malaysia, South Africa and Thailand. At a



Note: countries in the left-bottom comer of the graph are represented by the following numbers.
1: Zambia; 2:Brunei; 3: Bosnia & Herzegovina; 4: Guatemala; 5: Honduras; 6: Lesotho; 7: Jamaica; 8: Myanmar; 9: Saudi Arabia; 10: Indonesia; 11: Macao, Ch.; 12: Peru; 13: Nicaragua; 14: Ecuador; 15: Paraguay; 16: Tajikistan; 17: Trinidad & Tobago, 18: Philippines; 19 Burkina Faso; 20: Ethiopia; 21: Sri Lanka; 22: Armenia; 23: Mongolia; 24: Kuwait; 25: Egypt; 26: Madagascar; 27: Algeria; 28: Azerbaijan.

Figure 1.6 Changes in R&D intensity, 1996–2007.

meeting in 2005 to mark the 60th anniversary of South Africa's Council for Scientific and Industrial Research, leaders of science institutions from a number of these IDCs reached a consensus [109] on finding ways for S&T to play a part in sustainable development (Box 1.4). Some of the IDCs are now becoming significant development assistance partners, especially in Africa, including providing support for building higher education and research capacity [110].

1.6.2
Outputs from Investments in Research and Development

Not surprisingly, the higher levels of investments in R&D seen in many of the HICs and emerging economies correlate, at least to a degree, with science, technology and innovation outputs.

In particular, there is a strong correlation between levels of R&D investments and the densities of research workers [55] (Figure 1.7). For example, Asia represented 41.4% of world researchers in 2007 compared to 35.7% in 2002. This rise was mainly due to the increasing share of researchers in China from 14.0% to 20.1% between 2002 and 2007, reflecting the major increase in China's investments in R&D.

More funding and more researchers generally lead to higher outputs of scientific papers (Figure 1.8). The countries of the European Union, the United States and Japan collectively represent almost 70% of global R&D expenditure and these

Box 1.4 The Tshwane Consensus on Science and Development

The Emergence of Innovative Developing Countries (IDCs)

Many challenges remain. The world needs new and sustainable energy sources, protection from emerging diseases, and lower cost infrastructure. Moreover, the S&T environment has changed significantly in the last ten years. Knowledge production has been internationalized, access to money and skills has become increasingly competitive, and global technology and markets are changing with breathtaking speed.

A new set of actors has emerged in the quest to meet these challenges. Following sustained investment in education, research infrastructure and manufacturing in a number of developing countries, the IDCs have achieved high levels of economic progress and overall improvements in human wellbeing. How can these successes be generalized, and what role do the IDCs have in contributing to sustainable development?

The S&T leaders concluded that the IDCs can play a crucial role in developing innovative and appropriate solutions to global challenges, and at the same time strengthening their own S&T expertise. These leaders urged IDCs to coordinate their efforts, in order to increase investment in S&T aimed at the problems of developing countries. In particular, the leaders stressed the need for:

- developing nations, especially the poorest, to devote a proportion of their resources to S&T
- S&T leadership in developing countries to be strengthened and to define a clear set of priorities; this leadership needs to make a persuasive statement to the public that the scientific effort is essential and useful
- the political leadership of developing countries to press for a greater role in decision making on global development programs, including bilateral and multilateral aid; and to insist that a proportion of these resources be devoted to research and nurturing local scientific and technical capacity
- the benefits of S&T need to be extended to all; S&T efforts need to be increasingly directed to the creation of affordable and accessible products and services for poor people
- the strengthening of mechanisms, such as academies of S&T, for advising high levels of government on issues of S&T
- access to careers in S&T to be widened, and at the same time systems that reward and offer S&T careers to the most talented to be developed
- the broadening of the science education base within schools, technical colleges, universities, science councils, academies of sciences, government departments and industry; these institutions are fundamental to development and wealth creation. It is clear that an environment of excellent research is necessary to attract and retain young talent in scientific careers.

Although it is highly desirable for all countries, and especially developing countries to have functioning S&T systems, the symposium noted that this is not presently the situation, and in the interim several steps needs to be taken, including:

- the establishment of regional networks between national systems to overcome the lack of a critical mass, which is presently limiting the success of S&T in many developing countries
- the implementation of appropriate performance measures at all levels and for different types of S&T institutions in order to get the most out of the available resources

- the introduction of appropriate tax incentives and grants to encourage private sector participation in R&D; additional private sector resources for S&T could be accessed by addressing sources of market failure, including the preconditions for the entry of Technology Risk Capital.
- the close networking of universities, research councils and industry in order to promote innovation, entrepreneurship and wealth generation; mission-oriented clusters of institutions focused on identified priority issues must be established to aim at discovery, development and delivery.

In conclusion, the leaders noted that it is time for a number of important initiatives. It is time for developing and post-colonial societies to "name the ghosts" of science, technology, and higher education. While benefiting many people, S&T has also systematically excluded many groups. Governments and industries often use technologies in a way that harms both workers and the natural environment. Openness about these spectres will help to assure more equitable and constructive practices in the future.

It is also time for the IDCs to act collectively and think globally. An effective response to a number of shared global challenges, such as global climate change, infectious diseases and the loss of biodiversity, can only be achieved with the involvement of all countries, and especially the developing countries. The S&T systems of the innovative developing countries can play a crucial role in building such national capacity, and in shaping their own futures.

Extract from Tshwane Consensus [109]

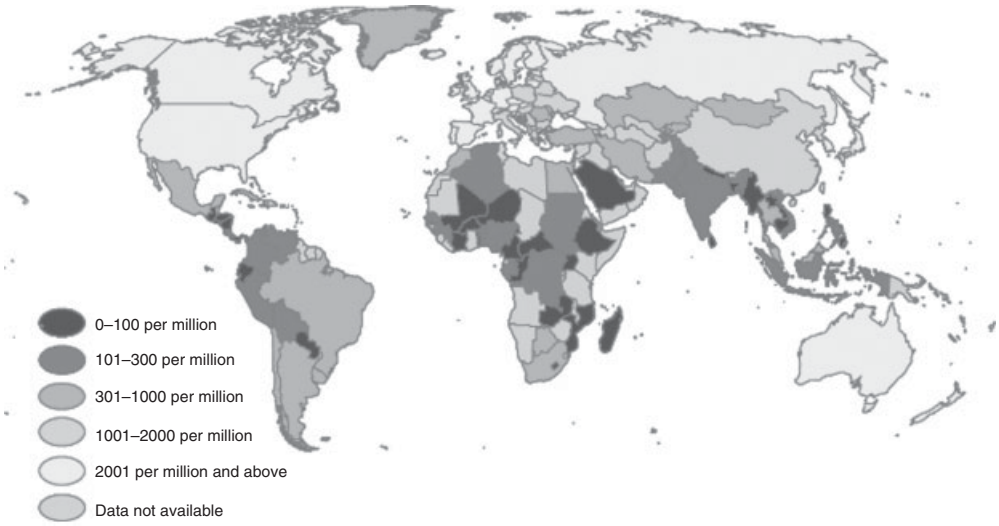


Figure 1.7 Researchers per million inhabitants, 2007 or latest available year. (Please find a color version of this figure in the color plates.)

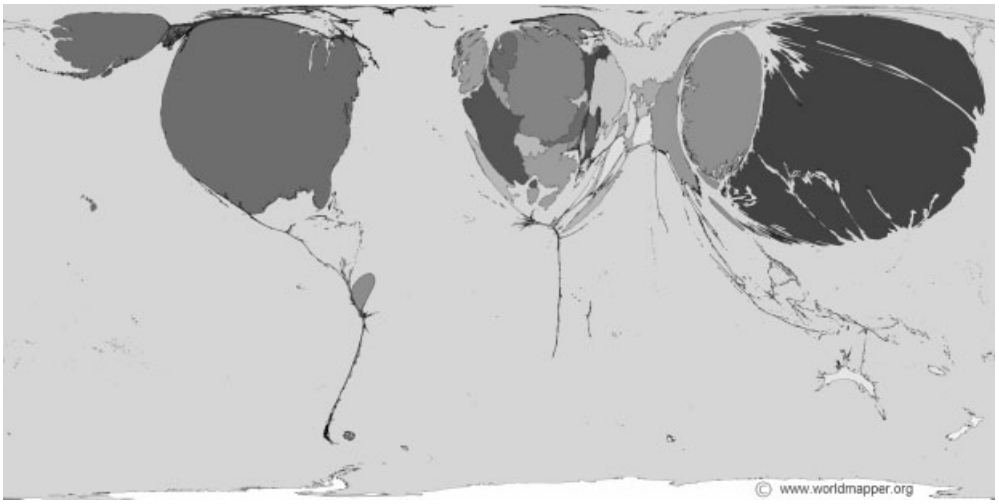


Figure 1.8 Scientific publications by countries, 2001. Territory size shows the proportion of all scientific papers published in 2001 written by authors living there. Scientific papers cover physics, biology, chemistry, mathematics, clinical medicine, biomedical research, engineering, technology, and earth and space sciences [111]. (Please find a color version of this figure in the color plates.)

countries publish roughly three times more scientific papers per person living there than in any other region [55, 111]. Bibliometric analyses of country outputs can be a useful tool for uncovering strengths and weaknesses in particular areas of science [112, 113]. For example, a 2003 study [114] in Malaysia noted that more papers were produced by Malaysian scientists in physical chemistry (10.16% of the total) than in any other area of science between 1955 and 2002, followed by agriculture (5.14% of the total).

The largest outputs of patents are predominantly made by HICs [111] (Figure 1.9). In 2002, 312 000 patents were granted around the world. More than a third of these were granted in Japan and just under a third were granted in the United States.

Migration of highly skilled workers ("brain drain") has a powerful effect on these outputs. The loss of such workers from the LMICs substantially lowers their capacities to innovate. One study [115] of migration identified 15 countries, especially in the Latin America/Caribbean and African regions, for which the percentage of highly skilled workers among the migrants was in the range 33–83% (Figure 1.10).

Migrants can bring a range of economic benefits to the receiving countries, including higher rates of innovation. Productivity gains in a number of destination places have been traced to the contributions of foreign students and scientists to the knowledge base. Data from the USA show that between 1950 and 2000, skilled migrants boosted innovation: a 1.3% increase in the share of migrant university graduates increased the number of patents issued per capita by a massive 15%, with marked contributions from science and engineering graduates and without any adverse effects on the innovative activity of local people [116, 117].

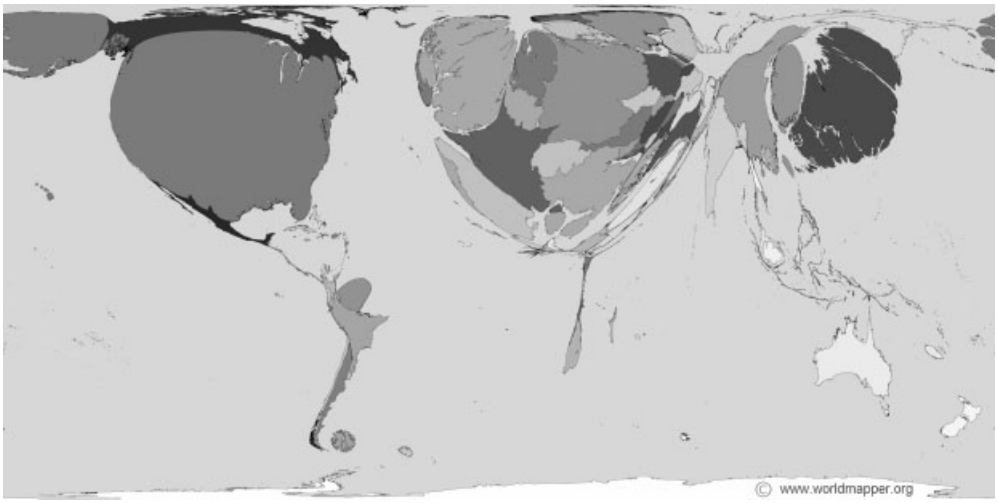


Figure 1.9 Patents granted by countries, 2002. Territory size shows the proportion of all patents worldwide that were granted there [111]. (Please find a color version of this figure in the color plates.)

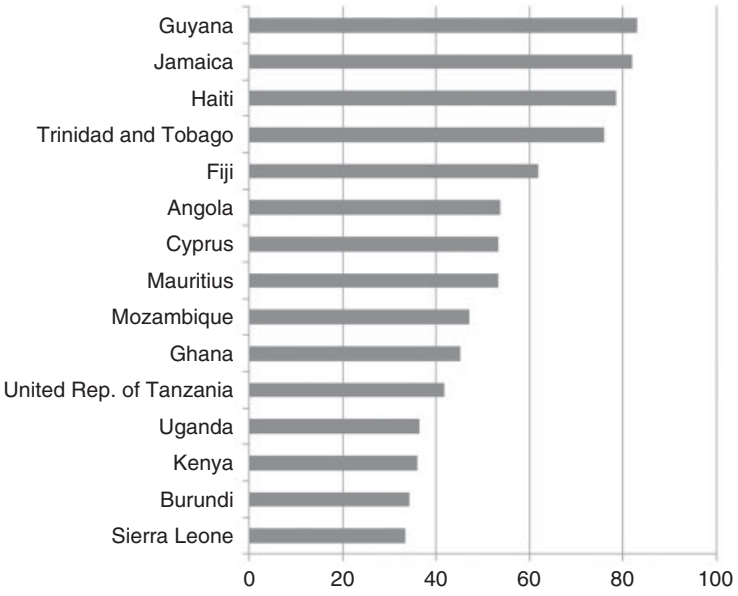


Figure 1.10 Percentages of highly skilled expatriates among total expatriates from selected non-OECD countries.

1.6.3 Connecting Science, Technology and Innovation

Scientific output in the form of publications and patents does not necessarily translate directly into innovation. The innovation environment in which scientists, inventors and entrepreneurs work plays a substantial role in determining how successful a country becomes in translating novel ideas into practical processes and products that contribute to national development [118].

A Rand study [119] which examined the scientific capability of 29 representative countries to adopt 16 technology applications divided the countries into four capability categories:

- **Advanced** – Australia, Canada, Germany, Israel, Japan, Korea, USA
- **Proficient** – China, India, Poland, Russia
- **Developing** – Brazil, Chile, Colombia, Indonesia, Mexico, South Africa, Turkey,
- **Lagging** – Cameroon, Chad, Dominican Republic, Egypt, Fiji, Georgia, Iran, Jordan, Kenya, Nepal, Pakistan

The study identified a number of major drivers and barriers to the capacity to adopt technological innovation:

- Cost and financing
- Laws and policies
- Social values, public opinion, and politics
- Infrastructure
- Privacy concerns
- Use of resources and environmental health
- R&D investment
- Education and literacy
- Population and demographics

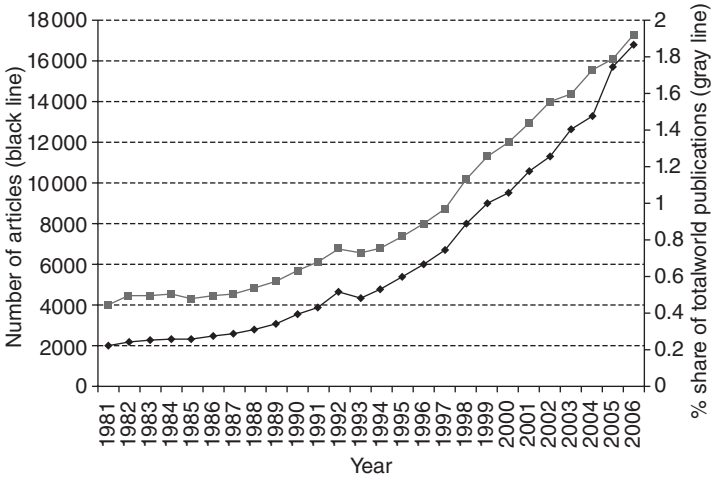
Clearly, therefore, countries need to do more than providing support for the pursuit of S&T if they wish to reap the economic and development benefits. Investments in science, including chemistry, must be coupled with national policies on the applications of science and creation of national environments that foster innovation. Brazil provides an example of a country that is pursuing this approach (Box 1.5) [120–125]. Areas now receiving considerable attention, which have a chemistry linkage, include bio-fuels and pharmaceuticals.

Among the emerging economies, China has demonstrated an extremely high growth rate in recent years, coupled with accelerated investments in R&D (Figure 1.6). An important aspect of China's development has been the capacity to take a long-term approach to investment and planning. It is remarkable that, in 2008, the Chinese Academy of Sciences published a 50-year science strategy as an extension to the Mid-to-Long-Term Plan for Development of Science and Technology (2006–2020) issued by the State Council of China [126].

While India's overall investment in S&T has lagged behind those of other emerging economies such as Brazil, China and South Africa, nevertheless, it

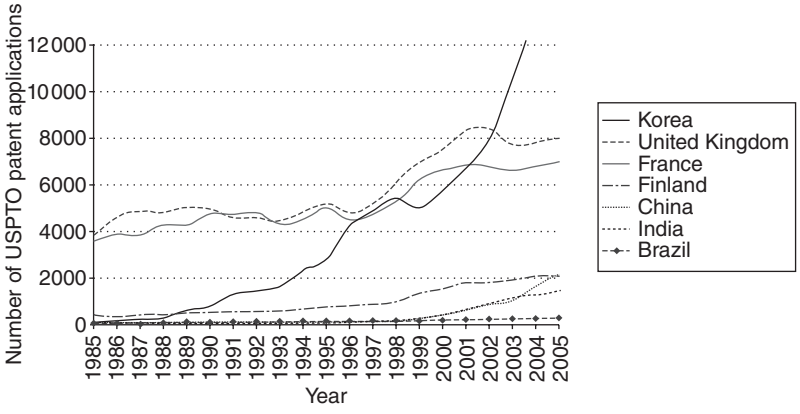
Box 1.5 Brazil's experience of promoting of science, technology and innovation (ST&I)

Brazil began systematically investing in S&T in the early 1950s, establishing national councils for research and for postgraduate education. During the military rule period (1964–1985), the funding system was consolidated with the creation of agencies for innovation and S&T; and a 1988 law required Brazil's states to create S&T funds. Following a serious health crisis in the 1980s, much greater attention began to be given to S&T to solve national problems and increased investment was soon followed by increasing scientific output.



Brazilian SCIE-indexed publications, total and share of world output, 1981–2006.

However, the environment for innovation remained unfavorable with laws that restricted the ability of university-based researchers to develop their discoveries. As a result, Brazil lagged behind other economies with a more open approach.



US Patent and Trademark Office patent applications, Brazil and selected countries, 1985–2005.

Continued

Brazil has taken some key steps to improve the environment for S&T and innovation:

- Sixteen Sectoral Funds were created between 1998 and 2008. They direct a fraction of the taxation of key industries to R&D projects to focus sector-specific research collaborations between enterprise, universities and research institutions, as well as to ensure the redistribution of research resources to help build capacity in less developed regions of the country.
- A new Innovation Law was adopted in 2004, to stimulate research innovation and remove barriers making it difficult for public sector researchers and private companies to collaborate.
- Brazil has significantly increased its investment in R&D in recent years, aiming to raise it from a level of around 1% of GDP to 1.5% by 2010 and with a prospect of increasing to 2% by 2020.
- Very importantly, the improved legal frameworks, investments and economic policies have been complemented in recent years by strong political will that recognizes that ST&I are at the center of economic development and social transformation.

See [120].

established a strong position in a number of chemistry-related fields. In particular, prior to accession to the World Trade Organization (WTO) in 2005 [127], India's non-recognition of product patents and innovative use of "reverse engineering" enabled it to establish itself as the "world's pharmacy", becoming the largest source of generic pharmaceuticals for other LMICs [128]. Having joined the WTO, the India pharmaceutical industry is being strongly encouraged to innovate and create its own intellectual property.

1.7 Capacity Building: Some Key Requirements for Chemistry's Role in Development

1.7.1 Evolution of Capacity Building Approaches in LMICs

Over the last few decades, the approach to capacity building in LMICs has evolved through several distinct stages [129]. Initially, the focus was on training of individual chemists – mainly by supporting students from LMICs to attend universities in HICs to obtain higher degrees and research experience. However, the graduates often found it difficult to pursue science careers on returning home and were frustrated by the lack of suitable laboratories, chemicals and equipment to follow their research interests. Some were assisted by maintaining close relationships with the HIC institutions where they had trained and by developing new North–South and South–South networks. Later, a more systematic effort emerged to develop high quality centers of advanced teaching and research in LMICs, creating their own cohorts of masters and PhD graduates and providing support and posi-

tions for those returning from advanced training abroad. However, chemistry and other academic disciplines have generally continued to have low esteem in many LMICs and to lack rewarding career pathways able to retain the brightest people. The importance of basic sciences like chemistry has been the subject of major emphasis in a number of conferences attended by scientists, policy makers and donor representatives [130, 131].

In the light of this mixed history of progress, but reflecting the strengthening recognition of the important roles that chemistry and other sciences need to play in the process of development, attention has turned, more recently, to addressing the entire system of teaching and research in a more comprehensive way.

Chemistry is taught within the overall framework of secondary and higher education. In many countries there are fragmented systems, sometimes involving separate ministries. Public sector teachers and lecturers in LMICs are often employed directly by the state as civil servants and subjected to a wide range of government regulations affecting salaries and terms and conditions of employment. Research is typically funded through a diverse and complex array of channels, including general university funding bodies, science councils and specific ministry programs and may involve federal, state and city-based sources and earmarked taxes. The private sector has developed some role in teaching, with many LMICs having private universities, but relatively little academic research is funded from private sources in these countries and few LMIC private universities have significant research capacity. On the other hand, overseas funding for research in LMICs may seem extremely large relative to national sources. The complex interplay of all these factors makes it difficult for a country to plan the growth and development of disciplines like chemistry and to enhance the roles that they play in addressing national priorities.

A further issue is that many scientists are resistant to the very notion of national planning, fearing that the identification of national priority areas for research will lead to the elimination of "blue skies research" and a corresponding loss of "academic freedom". An open public debate is needed which avoids taking an all-or-nothing extreme view about the direction of resources, but focuses on what are the appropriate proportions of national resources that should be apportioned to priority-focused versus undirected research and appropriate and evidence-informed mechanisms for selecting the priorities. Scientists in LMICs need to work more closely with their political leaders to show the importance of investment in long-term research in science. They are often very quick to blame their leaders for not allocating funds for research, especially when they wish to appeal for funding from external agencies.

1.7.2 National Policies for S&T

The evidence from emerging economies such as those in Brazil and China indicates that science and scientists fare well and make positive contributions to development when there are in place:

- National policies for science, technology & innovation, addressing
  - S&T education and training
  - financing national R&D
  - stimulation of innovation
  - exploitation of natural resources
  - environmental protection
  - regulation of medicines
  - regulation of intellectual property
- Key infrastructures for
  - education
  - R&D
  - innovation

Taiwan provides an example of a country which transformed its economy during the second half of the 20th century, with national planning and investment in chemistry capacity playing a key role (Box 1.6) [132].

1.7.3 Responsibilities

The government needs to take responsibility for instituting comprehensive policies for S&T, increasing levels of investment of public funds in R&D and fostering an environment that values knowledge and evidence and that promotes innovation. The scientists and academics have a corresponding set of responsibilities to be sensitive to national priority problems and to the need to communicate effectively with policy makers. The development of mutual trust and improved understanding between policy makers and researchers is now the subject of considerable attention – especially in areas such as health and the environment – and chemists must also engage in order to influence and benefit from the process.

It is also vital for international development partners to accept and orient their behavior towards the systems-based approach. Development assistance has undergone a revolution in recent years, with the preferred modality shifting away from project-based bilateral programs. Many such programs are now perceived to have had limited success and poor sustainability, being too donor-driven and failing to build country capacities and local support. In their place have come new multilateral arrangements based on sector-wide programs or general budget support, enabling the government to be in the driving seat in terms of national policies and fostering the building of government capacities for policy development, implementation and accountability. This new model has been formalized in the Paris Declaration on Aid Effectiveness and Accra Agenda for Action (Box 1.7) [133].

However, while the Paris/Accra principles are increasingly being applied to broad areas of development assistance to social and economic sectors and overall government finances, the entire field of research has been relatively neglected. Northern research institutions, research funding agencies and development assistance partners often still indulge in project-type approaches that engage individuals

Box 1.6 Taiwan's experience in national planning and investment in chemistry capacity

Taiwan's "economic miracle"

Taiwan's per capita GNP rose from US$919 in the 1950s, through US$1671 in the 1960s, US$3626 in the 1970s and US$6501 in the 1980s to US$7358 in 1990, as the agrarian economy was transformed into an export-oriented industrial one. In 1990, the total value of industrial production was US$165.3 billion, giving Taiwan foreign reserves of US$80 billion (first or second in the world).

By the early 1990s, the chemical industry was the largest industrial sector, contributing 24.2% of the total production value of US$165.3 billion, but only 8.5% directly to export sales of US$95.6 billion. This demonstrates the strategic importance of the chemical industry, as a supplier of materials and chemicals, in underpinning other export industries, including electrical/electronic goods and textiles.

The development of Taiwan's chemical industry can be divided into a number of phases:

1913–1943 Manufacturing of basic chemicals (e.g., fertilizer, chloralkali) (Japanese colony)

1944–1953 Production of substitutes for imported consumer goods (e.g., consumer commodities, agricultural products)

1954–1967 Development of light industries (emphasizing paper/food/textile products, etc)

1968–1975 Beginning of backward integration of petrochemical industry (boosting the export of textile products)

1976–1988 No. 3 and No. 4 naphtha crackers started with the fastest growth of the petrochemical industry

1989 Restructured strategy to shift away from commodities to higher value chemical products and advanced materials

Four main factors contributed to Taiwan's success in becoming one of the world's leading producers of a number of plastics and synthetic fibers by the 1990s:

1) Establishing an integrated chemical industry – for example, integrating backwards from a garment industry dependent on cheap labor and raw materials by successively developing capabilities for the synthesis of earlier intermediates (terylene: ethylene glycol, terephthalic acid: ethylene, xylene); similarly the shoe industry was strengthened by development of the plastics industry.

2) Development of a "debottle-necking" capacity – creating a cadre of skilled engineers and technicians able to de-bug and even improve on technology (e.g., for vinyl chloride production) originally imported from elsewhere.

3) Cooperation between up/mid/downstream operators for example, in the pricing of chemical raw materials, intermediates and products to ensure competitiveness.

4) Strong support by the government – including tax and investment incentives, well planned industrial zones, government-owned low-profit raw material and intermediate manufacturers and production/procurement agreement between these and petrochemical suppliers, custom/tariff protection and export incentives.

Information from [132]

Box 1.7 Paris Declaration on Aid Effectiveness and Accra Agenda for Action

The Paris Declaration, endorsed on 2 March 2005, is an international agreement to which over 100 Ministers, Heads of Agencies and other Senior Officials adhered and committed their countries and organizations to continue to increase efforts in harmonization, alignment and managing aid for results with a set of monitorable actions and indicators. Key principles are:

Ownership Partner countries set their own strategies for poverty reduction, improve their institutions and tackle corruption.

Alignment Donor countries align behind these objectives and use local systems.

Harmonization Donor countries coordinate, simplify procedures and share information to avoid duplication.

Results Partner countries and donors shift focus to development results and results get measured.

Mutual accountability Donors and partners are accountable for development results

The Accra Agenda for Action was drawn up in 2008 and builds on the commitments agreed in the Paris Declaration, focusing on:

Predictability Donors will provide 3–5 year forward information on their planned aid to partner countries.

Country systems Partner country systems will be used to deliver aid as the first option, rather than donor systems.

Conditionality Donors will switch from reliance on prescriptive conditions about how and when aid money is spent to conditions based on the partner country's own development objectives.

Untying Donors will relax restrictions that prevent partner countries from buying the goods and services they need from whomever and wherever they can get the best quality at the lowest price.

Based on [133].

or institutions in specific research or capacity-building programs. These not only ignore but often effectively undermine any existing national policies and programs, drawing scarce research resources into externally-driven activities. The Australian Centre for International Agricultural Research provides an example of an international research program that is shaped for congruence with the Paris/Accra principles [134]

Chemists in LMICs can help to counter these fragmentary approaches by engaging – among themselves, with science colleagues and with policy makers – in serious analysis and debate to promote the establishment of clear national policies on S&T and to support their governments to encourage HIC collaborators and development assistance partners to harmonize their approaches and work through and in alignment with the national policies.

1.7.4 Professional Associations and Cooperative Networks for Chemistry and Development

Professional associations such as national societies for chemistry or its subbranches played an important role in facilitating the development of the subject

and the profession in countries industrializing in the 19th and early 20th centuries. Through their impact on training, professional qualifications, the dissemination of information, encouraging good standards of practice, and the popularization of science, they contributed to strengthening the roles and contributions of chemistry in enhancing knowledge, wealth and health in these countries.

The world's largest professional society for chemistry is the American Chemical Society, founded in 1876 and now having over 160 000 members [135]. The Royal Society of Chemistry (RSC) [136] the largest organization in Europe for advancing the chemical sciences, was formed in 1980 by amalgamation of the Chemical Society (founded 1841); the Society for Analytical Chemistry (founded 1874, initially as the Society of Public Analysts); the Royal Institute of Chemistry (founded 1877, initially as the Institute of Chemistry of Great Britain) and the Faraday Society (founded 1903). Other early chemical societies include Australia (1917) [137], Austria (1897) [138], Brazil 1922 [139], Egypt (1928) [140], France (1901) [141], Germany (1867) [142], Japan (1878) [143], Netherlands (1903) [144], Norway (1893) [145], Portugal (1911) [146], Russia (1868) [147], South Africa (1912) [66], Spain (1903) [148], Sweden (1883) [149] and Switzerland (1901) [150].

The International Union of Pure and Applied Chemistry (IUPAC), founded in 1919, includes many national chemical societies among its members and provides global networking opportunities through its conferences and symposia [151]. Recently, international groupings of chemical societies have taken on regional networking and capacity building roles. For example:

- European Association for Chemical and Molecular Sciences – multiple society members from 37 European countries [152]
- Federation of African Societies of Chemistry (FASC) – established in 2006 with assistance from the Royal Society of Chemistry, it has 8 member societies, one of which is the West African Chemical Society (representing members from Benin, Burkina Faso, Côte d'Ivoire, Guinea, Mali, Niger, Senegal, Togo) [153].
- Federation of Asian Chemical Societies – 28 member chemical societies in the Asia-Pacific region [154].

In an example of inter-regional collaboration, FASC participated in launching the Pan African Chemistry Network (PACN). This is a program for Africa launched in November 2007 by the Royal Society of Chemistry with support from Syngenta. The PACN has established initial hubs in Nairobi and Addis Ababa and aims to help African countries to integrate into regional, national and international scientific networks [155].

One of the important ways that chemists can contribute to sustainable capacity development and utilization in LMICs is by developing and participating in South–South and North–South–South cooperation networks. During the course of the last century, advances in science generally have moved from being the work of highly gifted individuals (e.g., Galileo, Newton, Darwin, Einstein) to involving the work of localized groups of collaborators (e.g., Watson and Crick) and then to international networks of scientists. Such networks are not only a feature of "big" science such as particle physics or the human genome project, where dozens or

hundreds of collaborators may be involved at a large number of centers, but also extend to a range of projects which are substantial scientific challenges that require a critical mass of workers tackling different aspects of a problem and/or that cross disciplinary boundaries [156, 157].

In addition to these types of networks that are driven by the demands of tackling large, complex challenges, there are also networks whose purpose is to provide support and capacity building for scientists working in settings with limited resources.

Both types of networks are of relevance to many chemists working in LMICs, since they provide opportunities for being associated with large, cutting-edge global programs as well as enabling research to be conducted in areas of local interest or relevance and reducing researchers' isolation [158]. There are recent trends showing the increasing engagement of chemists in networks aiming to foster research, capacity building and development, as summarized below.

The International Science Programme (ISP) at Uppsala University supports networks on a long-term basis. It aims at assisting developing countries to strengthen their domestic research capacity within the chemical, physical and mathematical sciences. Support focuses on regional networks and on research groups that are primarily in least developed countries targeted by the Swedish government for long-term cooperation [159].

The International Foundation for Science (IFS) receives funding from governmental and nongovernmental sources, as well as national and international organizations and has an annual budget of approximately US$ 5 million. A primary form of support provided is in the form of an IFS Research Grant to young scientists at the beginning of their research careers, which amounts to US$ 12 000 and may be renewed twice. It is intended for the purchase of the basic tools needed to conduct a research project: equipment, expendable supplies, and literature. Since 1974 there have been 3500 IFS Grantees in Africa, Asia and the Pacific, and Latin America and the Caribbean. Of these 22% are women. IFS also acts as both enabler of existing and emerging networks and convener of new ones. Involvement is especially in the initial stages, with IFS providing seed money, co-operative and administrative assistance and funding to workshops, training courses, exchange visits and fellowship programs [160]. Examples of IFS-and ISP-supported networks active in chemistry include:

- AFASSA (Co-ordination of Networks for Research on Biological Resources in Africa, Asia and South America) [161]

  AFASSA was set up as a result of an international symposium on natural product research held in Montevideo, Uruguay, in 1999, with participating scientists from Africa, Asia and South America. Members to date are:
  - African Laboratory of Natural Products
  - Asian Network of Research on Anti-diabetic Plants
  - Latin American Network for Research on Bioactive Natural Compounds
  - Network for Analytical and Bio-assay Services in Africa
  - Natural Products Research Network for Eastern and Central Africa

  - Southern African Regional Co-operation in Biochemistry, Molecular Biology and Biotechnology

- ANCAP (African Network for the Chemical Analysis of Pesticides) [162]
  ANCAP was initiated in 2001 under IFS auspices. Member countries are to date Ethiopia, Kenya, Tanzania and Uganda.

- NABSA (Network for Analytical and Bio-assay Services in Africa) [163]
  See below. IFS and NABSA work on the issue of maintaining proper function of scientific equipment.

- NAPRECA (Natural Products Research Network for Eastern and Central Africa) [164]
  NAPRECA was initiated in 1984. Among the founding African scientists were several IFS grantees. In 1987, NAPRECA became affiliated to UNESCO as one of UNESCO's network programs. IFS continues to provide funding for specific projects. NAPRECA members are mainly chemists, but also biologists and pharmacologists, working on the chemistry, botany, biological activities and economic exploitation of natural products. The NAPRECA headquarters is located in Dar es Salaam, Tanzania, and there are local branches in nine countries (Botswana, DR Congo, Ethiopia, Kenya, Madagascar, Rwanda, Sudan, Uganda, Zimbabwe).

- NITUB (Network of Instrument Technical Personnel and User Scientists of Bangladesh) [165]
  NITUB was formed in 1994 with seed money from IFS. The main objective of NITUB is to improve the competence of scientists and technical personnel to operate, maintain, and repair scientific instruments. NITUB maintains an inventory of scientific instruments at Bangladeshi institutions, and aims to create a stock of spare parts. The General Secretary is located in the Department of Chemistry, University of Dhaka.

- NUSESA (Network of Users of Scientific Equipment in Southern and Eastern Africa) [166]
  NUSESA was initiated in 1989 with the aim to provide a forum for information and discussion on issues related to the purchase, use, and maintenance of scientific equipment in southern Africa. IFS passed on the coordination and administration of NUSESA activities when the NUSESA Secretariat was established in 1996 in Harare, Zimbabwe. Since NUSESA has secured funding from government agencies, IFS–NUSESA joint collaboration has evolved to policy-making and consultation. Local NUSESA chapters have been set up in 16 countries (Botswana, Eritrea, Ethiopia, Kenya, Lesotho, Madagascar, Malawi, Mauritius, Mozambique, Namibia, South Africa, Swaziland, Tanzania, Uganda, Zambia and Zimbabwe). The NUSESA Secretariat is located at University of Western Cape in South Africa.

- WANNPRES (Western Africa Network of Natural Products Research Scientists) [167]

WANNPRES was established in 2002 by COSTED (Committee on Science and Technology in Developing Countries) on the initiative of a group of scientists from universities and research institutes in West Africa. The objective with WANNPRES activities is to enhance research and capacity building in the conservation and effective use of natural resources in Africa. IFS involvement dates back to the founding assembly. The WANNPRES Secretariat is located at the Department of Chemistry at the University of Ghana.

In the area of providing support and capacity building for scientists working in settings with limited resources, IOCD [102] began a program in the 1980s to provide analytical services for chemists in LMICs. This was initially a North–South network, with chemists at City University, London (Stephen Matlin), the University of Missouri (Michael Tempesta) and the Universidad Nacional Autónoma de México (Carlos Rius) receiving samples from chemists in a range of countries in Africa, Asia and Latin America and providing, free of charge, infrared, ultraviolet, NMR and mass spectra. In some cases, at the invitation of the submitting group, assistance was provided with the interpretation of spectra and the elucidation of structures of synthetic and natural products.

In 1992, IOCD's Walter Benson participated in a meeting in Gaborone, Botswana which led to the launch of a new activity, the Network for Analytical and Bioassay Services in Africa (NABSA) and IOCD contributed launching funds. There has been strong support from the University of Botswana and additional funding for NABSA has come from a variety of international sources including USAID, UNESCO, International Programme in the Chemical Sciences (Uppsala) and TWAS.

Coordinated by Berhanu Abegaz at the University of Botswana [168], NABSA's objectives are:

- To promote the development of scientific activities in Africa by offering analytical, bioassay and literature support services to chemists.
- To cooperate with active scientists in a joint short-term intensive-research undertaking by inviting them to the reasonably well equipped laboratory in Botswana.
- To promote the professional development of young scientists by arranging sub-regional symposia.

NABSA collaborating centers contribute a range of spectroscopic facilities including 200–600 MHz NMR (Botswana, Ethiopia, Kenya, Lesotho, South Africa and Zimbabwe) (Figure 1.11). In the period 1998–2009, NABSA provided over 11 000 NMR spectra and over 2000 mass spectra to African scientists. Recipient countries of NABSA services have included Cameroon, Democratic Republic of Congo, Egypt, Ethiopia, Ghana, Kenya, Nigeria, Sierra Leone, South Africa, Sudan, Tanzania and Zimbabwe. More than 30 short-term visits to the University of Gabarone were arranged for chemists from different African countries [163].

From 2005, NABSA's focus shifted to research cooperation with research groups in selected countries and institutions, particularly in Cameroon, Ethiopia, Nigeria,

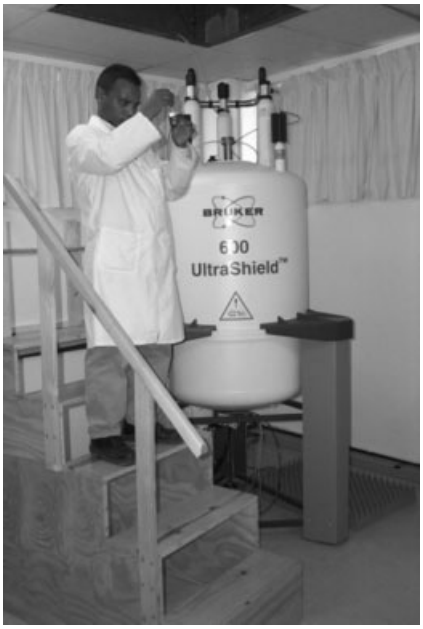


Figure 1.11 600 MHz NMR forming part of the NABSA analytical service at the University of Gabarone, Botswana. Photo from B.M. Abegaz.

South Africa, Tanzania and Zimbabwe, in order to help build and strengthen capacities and increase the overall impact of the collaboration. The NABSA centre itself has been involved in at least 63 publications in peer-reviewed international and regional journals Productive NABSA collaborations include a range of phytochemistry studies aimed at structure elucidation and the identification of bioactive natural products, such as anthraquinones [169]; Studies of Lathyrism, a disorder produced by toxic non-protein amino acids from the grass pea (*Lathyrus* species, *Leguminosae*) [170]; identification of dimeric sesquiterpenoid lactones from a south African plant (*Dicoma anomala*) possessing anti-plasmodial properties [171], leading to patenting; and identification of insect antifeedant bichalcones from *Rhus pyroides* and development of a general synthetic method for C–C linked bichalcones [172, 173].

NABSA has been an important source of energy and momentum for strengthening Africa's capacity for natural products research. In 2006 a new aspect was initiated with the establishment at the University of Botswana of the Centre for Scientific Research, Indigenous Knowledge and Innovation (CESRIKI), which has facilitated the Research Visitors' Programme of NABSA and also introduced a range of bioassay capabilities. In 2009, a consultative meeting in Gabarone led to an initiative for the establishment of a Pan-African Natural Products Library (p-ANPL) as a repository for compounds that have been isolated, identified and screened. The Board is drawn from several African countries.

Other networks operating in the African region include:

- SEANAC (Southern and Eastern African Network for Analytical Chemists). Established as a result of a SIDA-funded workshop in Gabarone in 2002, SEANAC's objectives are to (i) promote analytical chemistry in the region through collaboration, research, research training, teaching and information sharing; (ii) facilitate inventory, access, operation, maintenance and repairs of analytical equipment; and (iii) collaborate with organizations with similar aims [174].
- AAPAC (African Association of Pure and Applied Chemistry). AAPAC's objectives are to (i) provide a forum for the exchange among scientists and development agents of scientific information on the state of the chemical sciences in Africa; (ii) foster research in the chemical sciences; (iii) cooperate with other international bodies which pursue aims and objectives similar to those of AAPAC; (iv) promote mutually beneficial interdependent linkages between industry and other entrepreneurial bodies on the one hand and research institutes, including universities, on the other [175].

1.7.5 National Funding for Research

While funding from public and private international sources is very important, the availability of regular, national sources of public funding is essential if a critical mass of researchers is to be established and maintained [176].

In the 18th and 19th centuries, as the pace of technological progress increased before and during the industrial revolution, most scientific and technological research was carried out by individual inventors using their own funds [177]. Historically, the development of national channels for funding scientific research in Europe and North America is relatively recent and lagged considerably behind these first waves of technological progress. Apart from military research, they were driven by specific challenges such as public health problems and by broader concerns about international competitiveness. Some examples to illustrate this include:

- The Royal Society of England, founded in 1660 as a learned society, received a government grant in 1850 to assist scientists in their research and to buy equipment [178]. But it was in the early the 20th century that the challenge of tuberculosis led to the establishment in 1913 of the Medical Research Committee and the realization that Britain was falling behind its competitors led to the establishment in 1916 of the Department of Scientific and Industrial Research to fund applied research and technological innovation [179].
- Germany's Kaiser Wilhelm Gesellschaft zur Förderung von Wissenschaft und Forschung (KWG), now the Max-Planck-Gesellschaft, was created in 1911 [180]. The Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) traces its origins back to its forerunner, the Notgemeinschaft der Deutschen Wissenschaft, which was established in 1920 on the initiative of Fritz Haber [181].

- Following Belgium's early industrialization and development of chemical industries (Box 1.2), it established the Fonds National de la Recherche Scientifique (FNRS) in 1928 [182].
- The Centre National de la Recherche Scientifique (CNRS) was created in 1939 and is the largest governmental research organization in France and the largest fundamental science agency in Europe [183, 184].
- In the USA, the first National Institute of Health was established in 1930 for medical research, but the main science-funding agency, the National Science Foundation, was not created until 1950 [185].

1.7.6 Gender Issues

Across the world, there is a substantial under-representation of women in science. In 121 countries with available data, women represent on average 29% of researchers. In 37% of these countries, they represent less than one-third. Only about 15% of countries have achieved gender parity, and only a handful of others have more women researchers than men. The proportion of women researchers is low in Africa (33%) and particularly low in Asia (18%) (Figure 1.12) [55].

Equality for women in all fields of human activity, including in science, is first and foremost a fundamental human right. In addition, as noted by the World Economic Forum [186] "Countries that do not capitalize on the full potential of one half of their societies are misallocating their human resources and undermining their competitive potential".

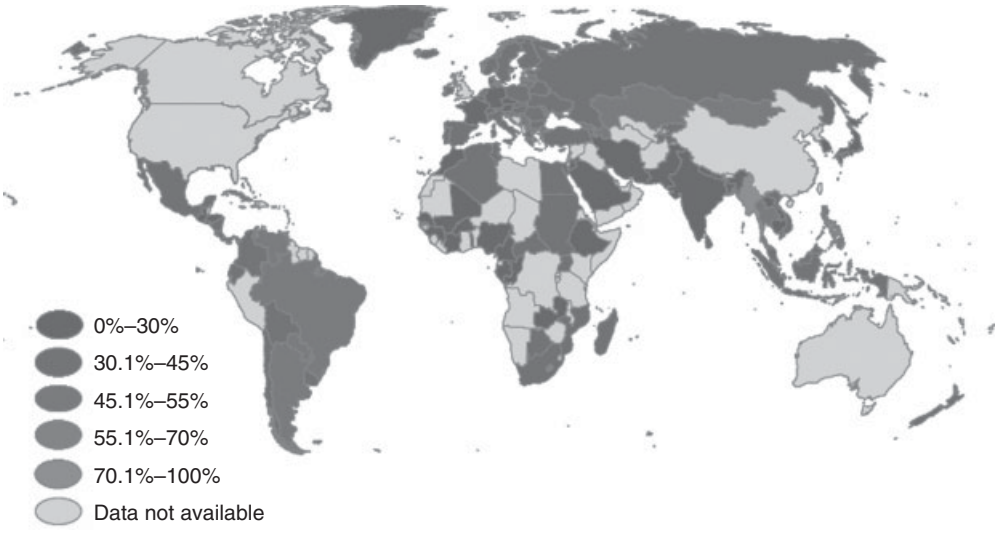


Figure 1.12 The gender gap in science. Women as a share of total researchers, 2007 or latest available year [55]. (Please find a color version of this figure in the color plates.)

To ensure that women achieve equality of opportunity in the field of science, countries need to institute a range of measures including ensuring equal access and treatment in school science programs and helping girls to build confidence in their science abilities; eliminating biases in university entrance procedures; and enacting anti-discrimination measures in employment [187].

1.7.7 Open Access

Access to published information is critical for scientists in order to enable them to assemble, understand and build on prior knowledge [188]. In chemistry and other sciences, there has been a rapid proliferation in the numbers and costs of journals over recent decades and libraries have found it difficult or impossible to keep pace. This problem has been especially acute for libraries in LMICs. As access to the internet has gradually extended to lower-income countries and some existing and newly established journals have introduced open access policies, there has been some improvement in access for those working in resource-poor settings. However, the situation is still far from satisfactory and there remains an unresolved tension between those who favor a policy of completely open, free-for-all access to scientific publications, those promoting special schemes which provide discounts and subsidies to scientists in certain LMIC-based institutions, and those wishing to preserve the commercial basis of publications. Learned societies in HICs are often finding this a challenging issue, since many produce leading journals in their subject areas and have become dependent on the income from journal subscriptions to maintain their financial viability [189–191].

Chemists need to work with their learned societies, professional bodies and funding agencies to seek innovative solutions that will maximize access to published material for those working in resource-poor settings.

1.7.8 Technology Transfer

The term "technology transfer" is used in two contexts, both of which are important to the subject of chemistry for development.

1) **Transfer between academia and industry:** In HICs, many universities have established "technology transfer offices" or "development offices" whose role is to assist academic researchers to find commercial applications for their discoveries. These offices may facilitate patenting, identifying commercial partners, start-up costs for small-medium size businesses, and so on. A number of technologically strong universities have established their own business parks as a further way of promoting commercialization.To date, relatively few LMIC academic institutions have adopted such models, so that their chemists and other scientists find it much more difficult to develop their ideas and may find it necessary to become involved in overseas partnerships instead.

LMIC governments and academic institutions can benefit from encouragement and assistance with establishing appropriate frameworks and building human resource capacities to be able to institute academic–industrial technology transfer arrangements as part of an overall enhancement of national innovation. There are instructive examples to be found in some successful initiatives in LMICs [192–194].

2) **Transfer of technology between countries:** The diffusion of technologies from one part of the world to another is not a new phenomenon. Prominent historic examples in the field of chemistry include a range of processes for chemical manufacture adopted from the Islamic world by European countries over a period of several centuries [195]. Interest now centers on the acquisition by LMICs of processes for manufacturing chemicals, pharmaceuticals and advanced materials. Tanzania provides an example (Box 1.8) of a country that has benefitted from giving greater attention to S&T and technology transfer [196, 197]).

Sources of technology for transfer may include the private sector [198, 199], international organizations [200] and public–private partnerships [201]. For LMICs,

Box 1.8 Science and technology (S&T) and the transfer of technology in Tanzania

S&T has made major contributions to Tanzania's economic development, including:

- Improvement of overall productivity by instituting structural changes in the models and methods of production which lead to greater efficiency and innovativeness in economic activities.
- Boosting exports by improving the quality of products, reducing the costs of production.
- Improvement of food security by supplying technical information to agri-producers and providing reliable markets for their products.
- Spreading income-earning opportunities by making technical information available to small entrepreneurs in rural and urban Tanzania.
- Upgrading the technical skills altitudes and productivity of the labor force through science and technology education and popularization.

Transfer of Technology

In the past, Tanzania had no deliberate strategies or plans for appropriate selection, acquisition and transfer of technology for effective integration of imported technologies with local capacity for R & D. However, currently, deliberate efforts have been put into place in order to make sure that the speed of technology transfer is effective and sustainable for example, establishment of the Tanzania Commission for Science and Technology in 1986 and the Centre for the Development and Transfer of Technology in 1994, to institute a workable mechanism for the coordination of capacity building in the selection assessment, negotiation, adoption, R&D, information exchange and extension services.

Policy/Regulatory Framework

The first National S&T policy was enacted in 1985 and revised in 1995, its major thrust being to establish a prioritized program for generating new knowledge and to determine strategies for

Continued

the application of S&T development in Tanzania. The broad objectives of the Tanzania S&T Policy are to:

- Promote science and technology as tools for economic development, the improvement of human, physical and social well being and for the protection of national sovereignty.
- Promote scientific and technological self-reliance in support of economic activities through the upgrading of R&D capabilities.
- Promote and encourage the public and private productive sectors in developing S&T.
- Promote active participation of women in S&T.
- Establish and/or strengthen national S&T institutions.

Legal Framework and Technology Policy Instruments

A scientific and technical advisory committee on S&T has been established in order to advise the President in addition to the Inter-Ministerial Technical Committee of the Cabinet.

A legal framework was laid down through the (Investment Promotion and Protection) Act of 1990, and the Tanzania Commission for Science and Technology Act No. 7 of 1986 which spells out the establishment of a National Centre for Development and Transfer of Technology. This Centre is charged with powers to establish rules and regulations for rationalizing the acquisition evaluation, choice coordination and development of technology.

In order to achieve the national goal of steady economic growth, maximum utilization of local resources and technology, expansion of technical education through development of local research units in enterprises, and long-term comprehensive technological policies integrated within the overall national development plans have been adopted by the Tanzania government.

Extracts from Tanzania *National Website [197]*

technology transfer may be the entry point for developing high value-added industries. In recent times, the growth of the chemical industries in Brazil, China and India have reflected a combination of technology transfer, adaptation and, in some cases, "reverse engineering" to devise new processes for making known high-value products.

The challenge for LMICs is to develop the combination of policy and economic conditions that favor inward investment, guarantee orderly labor relations and access to markets and that ensure the supply of well qualified scientists, managers and administrators.

1.8 Chemistry and Future Challenges to Health, Wealth and Wellbeing

1.8.1 "Glocal" – Thinking and Acting from Global to Local

Some have argued that the pursuit of S&T capability is not at all essential for lower income countries, but is an expensive luxury, while essential knowledge and tech-

nologies are best acquired by transfer from the richer nations that can afford to create them. But experience has shown that this view is not correct. Countries substantially accelerate their own development, enhance their competitiveness and strengthen their international negotiating positions by increasing their S&T capabilities. They need to be able to adapt and localize technologies even if these are created elsewhere; and they need the capacity to address their own problems, which are not always shared with higher-income countries (e.g., tropical diseases, local agricultural sustainability, specific water pollutants). Moreover, all countries need a basic capacity in fundamental areas like analytical chemistry in order to be able to monitor what is happening, identify problems and develop and apply solutions – for example, in relation to the quality of the environment or the quality and authenticity of pharmaceuticals in local supply.

These considerations are encouraging a growing realization that many problems have a dual character, involving a global dimension which requires joint international action on the one hand, but a local adaptation, application or focus on the other. Thus, as globalization progresses, the slogan "think globally, act locally" is evolving into a "glocal" or "glolocal" principle which requires everyone to "think and act globally and locally" and which has been adopted in the private sector [202, 203].

The following discussion of some major challenges provides a number of examples of the importance of this global–local duality of approach.

1.8.2 Agriculture, Food and Nutrition

The term "green revolution" describes a series of research, development, and technology transfer initiatives involving the development of high-yielding varieties of cereal grains, expansion of irrigation infrastructure, and distribution of hybridized seeds, synthetic fertilizers, and pesticides to farmers. They enabled many LMICs to increase their industrialized agriculture production and thereby to help meet the food and nutrition needs of growing populations (Box 1.9) [204]. Supported by the Rockefeller Foundation, the Nobel Laureate Norman Borlaug (who was also one of IOCD's early advisers) played a leading role in the green revolution, developing new, high-yield dwarf wheat that resisted a variety of plant pests and diseases and yielded two to three times more grain than traditional varieties.

Overall, raised productivity in three of the world's main staple food crops – rice, wheat, and corn – had substantial impact in a number of LMICs, including Mexico, Pakistan, India, and China. While this approach worked in Asia and other places where rice and wheat are the staple crops, it provided little benefit in Africa, where sorghum, millet, and cassava were consumed by the poor and where many countries also suffered from poor soil and uncertain rainfall, a shortage of trained agriculturalists and lack of technology. Borlaug was convinced that, while traditional plant breeding methods remained important, agricultural biotechnology and herbicide-resistant crops had a vital role to play in places like Africa [205, 206].

Box 1.9 The green revolution

"Record yields, harvests of unprecedented size and crops now in the ground demonstrate that throughout much of the developing world – and particularly in Asia – we are on the verge of an agricultural revolution.

- In May 1967 Pakistan harvested 600 000 acres to new high-yielding wheat seed. This spring (1968) the farmers of Pakistan will harvest the new wheats from an estimated 3.5 million acres. They will bring in a total wheat crop of 7.5 to 8 million tons – a new record. Pakistan has an excellent chance of achieving self-sufficiency in food grains in another year.
- In 1967 the new high-yielding wheats were harvested from 700 000 acres in India. This year they will be planted to 6 million acres. Another 10 million acres will be planted to high-yield varieties of rice, sorghum, and millet. India will harvest more than 95 million tons in food grains this year – again a record crop. She hopes to achieve self-sufficiency in food grains in another three or four years. She has the capability to do so.
- Turkey has demonstrated that she can raise yields by two and three times with the new wheats. Last year's Turkish wheat crop set a new record. In 1968 Turkey will plant the new seed to one-third of its coastal wheat growing area. Total production this year may be nearly one-third higher than in 1965.
- The Philippines have harvested a record rice crop with only 14% of their rice fields planted to new high-yielding seeds. This year more land will be planted to the new varieties. The Philippines are clearly about to achieve self-sufficiency in rice.

These and other developments in the field of agriculture contain the makings of a new revolution. It is not a violent Red Revolution like that of the Soviets, nor is it a White Revolution like that of the Shah of Iran. I call it the Green Revolution.

This new revolution can be as significant and as beneficial to mankind as the industrial revolution of a century and a half ago. To accelerate it, to spread it, and to make it permanent, we need to understand how it started and what forces are driving it forward. Good luck – good monsoons – helped bring in the recent record harvests. But hard work, good management, and sound agricultural policies in the developing countries and foreign aid were also very much involved."

Extract from speech by William Gaud, 6 March 1968 [204]

With the world's population expected to increase by 50% in the first half of the 21st century, there is now seen to be a need for another green revolution [49, 207] – one which provides adequate amounts of nutritious and affordable foodstuffs in high yield and through agricultural practices that do not require large increases in the global amount of land under cultivation and do not cause pollution of earth, water or air.

Chemistry has many roles to play in meeting these challenges, from soil chemistry to pollution monitoring, from creation of better methods of plant crop protection to helping develop new, more productive and more robust varieties.

1.8.3
Climate Change

There is now incontrovertible evidence that human activities during the last couple of centuries are responsible for a significant rise in average global temperature and a concomitant increase in extreme weather events such as floods and droughts and heat waves and severe winters. If continuing unchecked, this global warming and shifts in weather patterns threaten the lives and livelihoods of many millions of people, due to drought, desertification, floods, inundation of major tracts of land and even entire small island states, disruption of agriculture and the spread of water- and vector-borne diseases [208].

While there has been a temptation to see the impact of climate change as being something that will be felt some decades in the future, there is much evidence that the adverse consequences are already being seen, Three broad categories of health impacts are associated with climatic conditions (Figure 1.13) [209] and current excess mortality due to climate change may now be a few hundred thousand extra deaths per year [209, 210].

Furthermore, it is of special concern that the most severe impact of climate change is being felt by vulnerable populations who have contributed least to the problem. The risk of death or disability and economic loss due to the adverse impacts of climate change is increasing globally and is concentrated in poorer countries [48, 209].

Needed contributions from chemistry towards the international response to climate change involve a combination of measures to mitigate its extent and to adapt to the unavoidable consequences. LMICs will require substantial assistance, including technology transfer [211]. Chemistry has already made innumerable contributions to identifying climate-related problems (e.g., through environmental analysis of ozone depletion and greenhouse gas emission); understanding their underlying causes and contributing solutions (e.g., through studies of the roles of

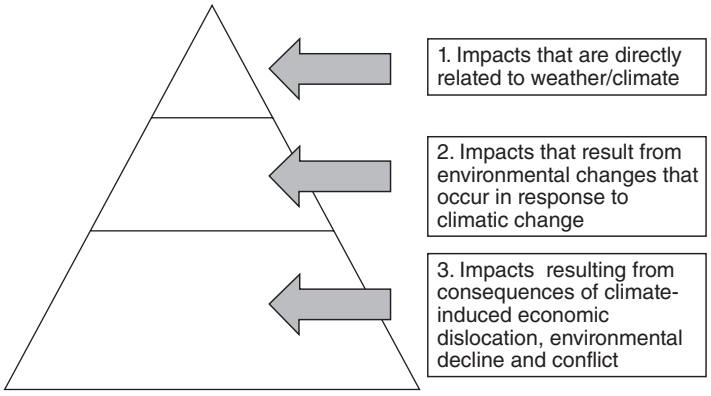


Figure 1.13 Health impacts associated with climatic conditions. Reproduced from [209].

gases like carbon dioxide and methane and the atmospheric photochemistry of fluorocarbon refrigerants and aerosol propellants).

Much more will be required of chemistry in the future, including better, cheaper and more robust field methods for environmental monitoring and impact assessment and developing new processes for energy generation, industrial production, materials recycling, and so on.

1.8.4 Energy

Intimately linked to the issue of climate change is the challenge of generating energy in an environmentally appropriate and sustainable manner.

World energy use (measured in kg of oil equivalent per capita) rose from 1338 kg per capita in 1971 to 1819 kg per capita in 2007, while world population rose from 3.8 billion to 6.6 billion during that period, representing an absolute increase in energy consumption of 236% [212]. With 80–90% of world energy production coming from the combustion of fossil fuels, there is an extremely urgent need to develop environmentally sustainable methods of energy generation, as well as methods for increasing the efficiency of energy use.

Among the many roles that chemistry is playing are:

1) devising ways of extracting and fixing carbon dioxide and other pollutants generated in burning coal and oil and reducing the emission of particulates that damage the respiratory system;
2) contributing to the improvement of efficiency of combustion engines, thereby reducing consumption;
3) developing new bio-fuels which reduce the net difference between current absorption and emission of greenhouse gases;
4) innovating new materials for the capture of sunlight and its transformation into energy;
5) creating new, environmentally clean processes for energy production, such as fuel cells.

1.8.5 Environment and Sustainable Development

Human activities such as the clearing of land for agriculture; diversion of water courses and extraction of water from underground water tables; extraction and transformation of raw materials to provide physical products and energy for power appliances; and the creation of domestic and industrial waste materials, all contribute to changes in the natural environment.

The pace of these changes has accelerated as the population of Earth has grown, especially in the last couple of centuries, but it is only within the last few decades that the extent of the problem has been recognized internationally [213] and a

concerted global effort has begun to address the issues which have come to be expressed in the objective of "sustainable development". This term encapsulates the need to ensure that human beings can live in an environment free from pollutants and health hazards and that the sum of human activities does not cause degradation of the physical and biological environments of the planet. In particular, the Rio Declaration emerging from the UN Conference on Environment and Development, held in Rio de Janeiro in 1992, provided the fundamental principles and a program of action ("Agenda 21") for achieving sustainable development, establishing linkages among economic and social development and environmental protection [57]. This was followed up a decade later by the World Summit on Sustainable Development held in Johannesburg in 2002 [58].

Sustainable chemistry is understood as the contribution of chemistry to the implementation of the Rio Declaration and Agenda 21 and of follow-on processes such as the Johannesburg Declaration. The focus of sustainable development concerns in areas such as water, energy, nutrition, human habitation and the quality of the environment mean that chemistry has many central roles to play – in monitoring the environment, in understanding, preventing and mitigating adverse chemical/biological impacts of human activities, and in devising new, cleaner and more environmentally sustainable process. Strengthening capacity for analytical chemistry in LMICs is indispensable to achieving these objectives [214].

1.8.6 Health

The nature of health challenges faced in every part of the world is changing, as a result of shifting patterns of disease, the globalization of health threats, changes in the environment and in human behavior. Some of the key health challenges and roles for chemistry in meeting them are highlighted below.

Non-communicable diseases (NCDs – e.g., cancer, diabetes, heart disease, stroke) are becoming the most prevalent causes of ill-health and death everywhere – but in many LMICs they are appearing alongside still-prevalent *infectious diseases* – including some that are global challenges (e.g., HIV/AIDS and tuberculosis) and some specific to tropical regions (e.g., malaria, African and South American trypanosomiasis, visceral and cutaneous lieshmaniasis, schistosomiasis). Those diseases that are prevalent globally, like NCDs and HIV/AIDS, have attracted considerable R&D investment and drugs to prevent or treat them have become available, but often these are too expensive for use in LMICs and/or require functioning health systems able to support patients with chronic conditions – which many LMICs lack. Many of the tropical infectious diseases have been neglected by the global pharmaceutical industry and new or improved drugs are still needed [215]. Chemists have a central role to play in the discovery of new drugs for these communicable and chronic diseases – drugs that are safe, effective, affordable and suitable for use in resource-poor settings where there may be a dearth of cold chains, laboratory diagnostic and clinical analysis facilities and specialist medical facilities and personnel.

The *quality of pharmaceuticals in circulation* in developing countries has become an issue of very serious concern in recent years. Drugs that are sub-standard, illegal imitations and non-effective fakes are widely available in many LMICs, as well as authentic drugs that are out of date or have deteriorated due to poor conditions of transport and storage [216]. Every country needs mechanisms to identify such pharmaceuticals – requiring the establishment and maintenance of well-equipped and staffed national analytical laboratories able to conduct reliable and speedy analyses to internationally-recognized standards. Recently, there has been demand from some LMICs to be able to localize the production of pharmaceuticals essential to their health needs, which increases the need for local capacities for drug regulation and quality control.

Pandemics such as those caused by viruses responsible for severe respiratory diseases (e.g., SARS, avian flu, swine flu) have emerged as a serious global threat in recent years, with the potential to cause ill-health, death and economic disruption on a very large scale [217]. The importance of the contributions of clinical chemistry to diagnosis and of medicinal chemistry to developing preventions and treatments cannot be underestimated.

Demographic changes are occurring on an unprecedented scale of speed and scope. The world's population grew from a level of about 1 billion in 1800 to 2 billion around 1920 and then leapt to 6 billion in 2000. It is predicted to reach around 9 billion by 2050. The birth rate in HICs declined to around replacement level during the 20th century, so that most of the 50% increase anticipated in the 21st century will be in LMICs. While average population ages are rapidly increasing in HICs, creating serious challenges in the worker/dependent ratios, many LMICs currently have very high proportions of young people [218]. These demographic shifts have major implications for patterns of consumption and demands for physical and energy resources – and also for health. For example, the largest cohort of adolescents that the world has ever seen requires greatly increased attention to sexual and reproductive health services, including safe, effective and acceptable means of family planning, while the growing numbers of aging people will present growing challenges in the management of chronic diseases, including disabling conditions such as arthritis. Chemistry can make a major contribution, not only to the development of new drugs, diagnostics and medical devices appropriate to these changing populations but also to the creation of new materials that enhance their quality of life.

In 2007, for the first time the world's population living in urban settings was as large as the rural population and this *urbanization* trend is continuing. As most of the world's increase in population is happening in LMICs, a substantial proportion of the new citizens of the planet in the next half century will be living in cities in less wealthy countries, where currently many people live in very unhealthy conditions in crowded urban slums and informal settlements with poor access to clean water and sanitation [219]. Chemistry and allied sciences such as chemical engineering and materials, food, energy and sewage treatment sciences have much to offer in helping to ensure the availability of salubrious living conditions including healthy dwellings, clean water, sanitation and safe foodstuffs.

1.8.7 Intellectual Property

About US$ 1.1 trillion was invested globally in R&D in 2007, up from about US$ 525 billion in 1996. This intensification of investment in the creation of new knowledge has, in turn, fueled demand for rights over knowledge – that is, for the protection of intellectual property (IP). In 2007, 1.85 million patent applications, 3.3 million trademark applications and 621 000 industrial design applications were filed around the world. This creation of new knowledge is increasingly taking place in emerging economies and involving researchers in LMICs. As one indicator of the globalization of research, 21.9% of scientific articles in 2007 were internationally coauthored, three times as many as in 1985. Further scope for participation of LMIC scientists in the creation of IP is being provided by the increasing trend of "open innovation" – the tendency for firms to look outside themselves to satisfy their innovation needs, whether through traditional means, such as licensing, sub-contracting, R&D contracts or joint ventures, or through newer means, such as the use of problem-solvers on the Internet or open source cooperation [39].

Two aspects of IP issues are of particular relevance in the context of chemistry for development:

1) **Protection of IP rights of LMIC citizens and institutions:** This has been of particular concern in relation to the discovery or invention of useful processes and products by researchers in LMICs, and the protection of indigenous knowledge in areas like traditional agriculture and medicine.

With regard to this aspect, it is particularly important for LMIC governments to ensure that strong IP policies and legislation are developed and implemented and that governments and institutions in LMICs strengthen their capacities for the management of IP at the national levels and for the negotiation of IP issues at the global level.

- The World Intellectual Property Organization (WIPO) is a specialized UN agency established in 1967, dedicated to developing a balanced and accessible international IP system which rewards creativity, stimulates innovation and contributes to economic development while safeguarding the public interest [220]. WIPO's new Medium Term Strategic Plan [39] includes multiple technical assistance and capacity building activities.
- Among the tools available to assist in capacity building, the handbook of best practices in intellectual property management in health and agricultural innovation is an outstandingly useful resource [221].

2) **Protection of IP rights of LMICs to flexibilities under the rules of the World Trade Organization (WTO):** WTO deals globally with the rules of trade between nations, providing standards, enforcement and dispute settlement under the Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) which came into effect on 1 January 1995 [222]. The TRIPS Agreement includes flexibilities in rules concerning the observance of patent rights

on medicines and allows an exception that Members may exclude from patentability diagnostic, therapeutic and surgical methods for the treatment of humans or animals. In some cases, this can allow member states to import or synthesize drugs considered essential to their national health needs. However, the rules also make provision for agreements among parties which introduce additional restrictions. These "TRIPS Plus" variations have been extremely controversial in cases where they have the effect of reducing the ability of LMICs to protect the public interest. One notorious example has been the dispute over the rights of LMICs to access drugs for the treatment of HIV/AIDS without having to pay the extremely high prices that were being charged by multinational pharmaceutical companies [223–225].

Following the work of a Commission on Intellectual Property Right, Innovation and Public Health [226], an inter-governmental negotiating process under the aegis of the World Health Organization led to agreement on the WHO Global Strategy and Plan of Action on Public Health, Innovation and Intellectual Property [227]. In this document, Member States endorsed by consensus a strategy designed to promote new thinking in innovation and access to medicines, which would encourage needs-driven research rather than purely market-driven research to target diseases which disproportionately affect people in developing countries. The challenge now is to achieve consistency in the ways that member states approach IP issues in their dealings with WIPO, WTO and WHO.

1.8.8
Natural Resources Exploitation

Exploitation of the Earth's physical and biological resources has always been a feature of human activities and the pace and extent of this exploitation have increased markedly in the last two centuries – both driven by and feeding technology advances, economic growth and population expansion.

Many LMICs which have been mainly the source of raw materials such as minerals and primary agriculture products now wish to reap the economic and developmental benefits of increasing production and adding value to the materials through processing. At the same time, there is pressure on these countries to conserve their natural resources, engage in sustainable development and not follow the historic pathways set by HICs which have led to pollution, exhaustion of resources and loss of biodiversity.

Countries everywhere are now being faced with the challenges of dwindling stocks of many key natural resources such as minerals, oil, and gas, as well as a global reduction of biological diversity due to factors such as deforestation, overfishing, hunting, and excessive use of monocultures in agriculture.

Some examples of key contributions of chemistry to these challenges include developing cleaner, more efficient, less energy-intensive and less polluting extraction and refining methods for minerals; methods for the recycling of inorganic and organic materials, and new substitute materials.

Chemical studies of natural products have made exceptionally valuable contributions to human health and wealth. As an example of "chemical prospecting", Eisner [228] cites the example of ivermectin, a fungal metabolite of highly complex structure which has been marketed for treatment of parasitic worm infections in animals and human beings. As well as generating US$ 1 billion in sales, its donation by Merck to the World Health Organization laid the basis for the treatment and prospective eradication of river blindness (onchocerciasis) in Africa.

The exploitation of biological resources has become an area of particular concern. Conservation of biodiversity is considered vital for long-term human survival because plants, animals and bacteria can be the source of new nutrients, genes conferring resistance to crop pests, and drugs for combating diseases. This implies that studies are undertaken globally to uncover these valuable assets, but exploitation needs to conserve their stocks as well as ensuring appropriate rewards for their owners. Countries which have some of the most valuable and diverse and least studied biological resources – often LMICs – have sometimes experienced "biopiracy" in which samples of plants or knowledge about their uses have been taken abroad and exploited without benefit to the country of origin or to the local inhabitants whose indigenous knowledge has been the key.

Valuable lessons have been learned from the experience of LMICs that have developed ways to meet these challenges. One very instructive example has been that of Costa Rica, a tiny Central American country which covers 0.04% of the world's total land area, yet is believed to harbor about 4–5% of the estimated terrestrial biodiversity of the Earth. In 1989, Costa Rica founded a national institution to gather knowledge on the country's biological diversity, its conservation and its sustainable use. The Instituto Nacional de Biodiversdad (INBio) was established as a non-profit, public interest NGO with a high degree of autonomy and with initial financial assistance from the Swedish Cooperation Agency (SIDA) and the MacArthur Foundation [229]. In 1991, INBio instituted an innovative agreement with a multinational pharmaceutical company, in which Merck was granted the right to evaluate the commercial prospects of up to 10 000 plant, insect, and microbial samples collected in Costa Rica. In return for these "bioprospecting" rights, Merck paid INBio US$ 1 million over two years, and provided equipment for processing samples and scientific training. Merck also agreed to pay a royalty – to be shared equally by INBio and the Costa Rican Ministry of Environment and Energy – on the profits of any future pharmaceutical product or agricultural compound that was isolated or developed from an INBio sample. Subsequently, INBio negotiated several further bioprospecting contracts involving other partners than Merck, including Eli Lilly, with the result that income from INBio's bioprospecting activities rose to about US$ 1 million per year [230].

The Costa Rica example demonstrates the possibility of conducting research to identify new medicinal products from natural sources in an LMIC, in a way that preserves property rights, generates a financial return and encourages capacity building. The contrasting experience of Eli Lilly's efforts to work in Cameroon highlights the importance of clear government policies and laws that establish the legal basis for conducting bioprospecting [231, 232].

Many of the lessons were taken forward in the International Cooperative Biodiversity Groups Program, initiated in 1992 to make multi-disciplinary, multi-institutional awards to foster work on the three interdependent issues of drug discovery, biodiversity conservation, and sustainable economic growth [233, 234]. On a more modest scale, as a result of Eisner's encouragement, in 1996 IOCD established a program to facilitate and catalyze ethical bioprospecting. This has included work in South Africa, Kenya and Uganda – in the latter case, most recently assisting policy makers in the development of draft legislation [235].

1.8.9 Water

Unsafe water, coupled with a lack of basic sanitation, kills at least 1.6 million children under the age of five every year. The target under MDG Goal 7 aims to halve, by 2015, the proportion of people without sustainable access to safe drinking water and basic sanitation. There has been some progress: between 1990 and 2006, over 1.5 billion people gained access to improved drinking water sources (usage of drinking water from improved sources rising to 87%, as compared to 77% in 1990) and over 1.1 billion gained access to improved sanitation (usage of improved sanitation facilities rising to 62%, as compared to 54% in 1990) (Figure 1.14). However, while the world is on track to meet the MDG drinking-water supply target by 2015 at the global level, many countries in sub-Saharan Africa and in Oceania are currently projected to miss MDG country targets, leaving significant portions of the population without access to improved drinking-water supplies. Moreover, the world is not on track to meet the MDG sanitation target by 2015. About 340 million Africans lack access to safe drinking water and almost 500 million lack access to adequate sanitation [236–239].

Meeting the challenge requires coordinated action by international agencies, governments and civil society partners, with a strong focus on sanitation [240] and major inputs from science and technology. Efforts among the various UN agencies with an interest in water and/or sanitation are coordinated by UN-Water [241].

The role of the chemical sciences is crucial, as highlighted in the report [242] of the Pan African Chemistry Network's "Sustainable Water Conference 2009". One of the key messages from this conference was that increasing Africa's capacity in analytical chemistry is imperative in order to support chemical monitoring and water management activities.

1.9 Conclusions

Chemistry has demonstrated its capacity to contribute substantially to increasing health, wellbeing and economic growth [243] and has great potential to assist in the development of LMICs. But what is needed for the future is not simply more of the same. The world has changed substantially in the last two centuries and

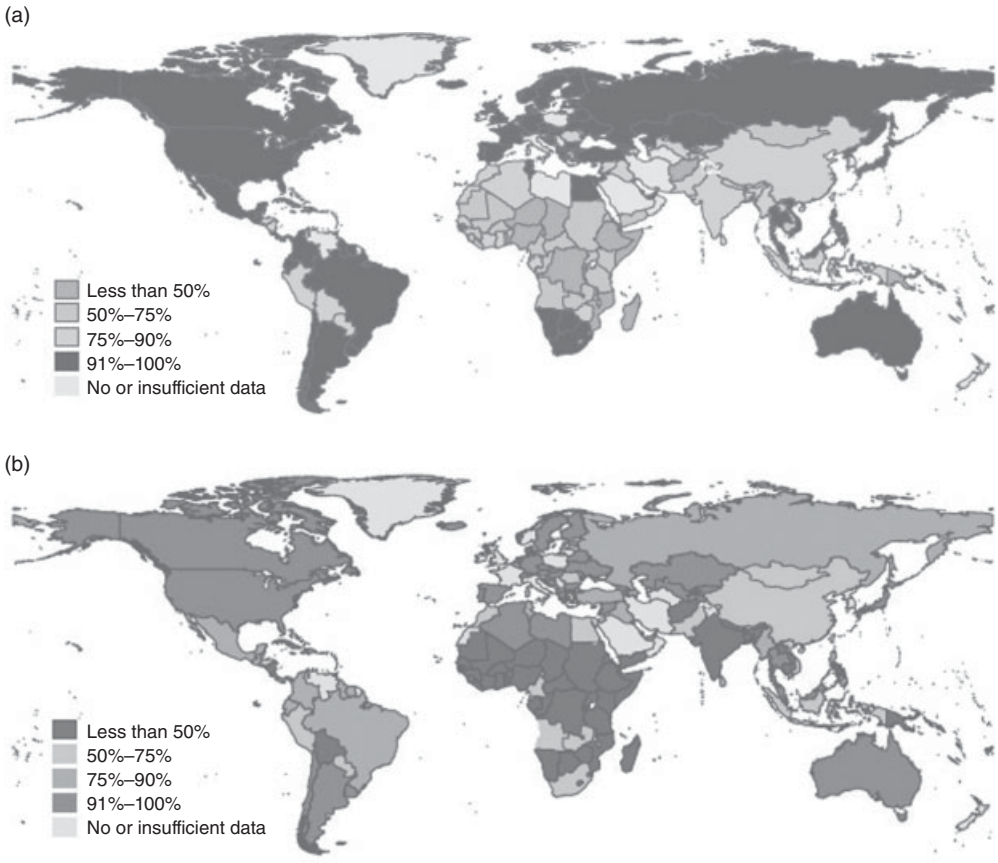


Figure 1.14 Global water and sanitation coverage [236]. (a) Improved drinking-water coverage, 2006. (b) Improved sanitation coverage, 2006. (Please find a color version of this figure in the color plates.)

many of the challenges it now faces are not only bigger but in some cases fundamentally different from those in the past. Globalization, urbanization, aging and population increases, threats of new diseases, pandemics, irreversible environmental damage and climate change, the revolutions in information technology, genomics and nanotechnology – all of these, individually and collectively, present new playing fields and new opportunities in an increasingly competitive and interconnected world.

Chemistry is a platform science. Its potential is, therefore, not only to help improve the human condition through its own direct contributions in fields such as the analysis and transformation of chemical entities, but also through the underpinning it provides to such diverse fields as medicine, genetics, biotechnology, materials and energy.

To take advantage of the new opportunities and to ensure that chemistry fulfils its potential, LMICs must invest in their own future by strengthening their science base and ensuring that the conditions exist for science, technology and innovation to flourish. This requires establishing sound policies, adequate funding mechanisms and environments where research is valued and its products utilized. It can be assisted by development partners, both public and private, who support the country's own policies and help build individual, institutional and country capacities to conduct, manage and exploit the fruits of science and technology [244].

Acknowledgments

The authors thank members of the Board of IOCD and in particular its President, Jean-Marie Lehn, and Executive Director, Alain Krief, for helpful discussions.

References

1 Gardner, C.A., Acharya, T., and Yach, D. (2007) Technological and social innovation: a unifying new paradigm for global health. *Health Affairs*, **26** (4), 1052–1061. http://content.healthaffairs.org/cgi/content/abstract/26/4/1052 (accessed 6 January 2011).

2 Buchmann, I. (2001) *Batteries in a Portable World*, 2nd edn, Cadex, Vancouver, http://www.buchmann.ca (accessed 6 January 2011).

3 Burns, R. (1951) Industrial and engineering chemistry – "electrochemical industry". *Ind. Eng. Chem.*, **43** (2), 301–304. http://pubs.acs.org/doi/abs/10.1021/ie50494a603 (accessed 6 January 2011).

4 Chemical Heritage Foundation (2011) Herbert Henry Dow. http://www.chemheritage.org/discover/chemistry-in-history/themes/electrochemistry/dow.aspx (accessed 6 January 2011).

5 Beck, T.R. (2008) Electrolytic production of aluminium. Electrochemistry Encyclopedia, http://electrochem.cwru.edu/encycl/art-a01-al-prod.htm (accessed 6 January 2011).

6 Yess, M. (2008) The electrochemical society: the first hundred years, 1902–2002. Electrochemistry Encyclopedia, http://electrochem.cwru.edu/encycl/art-e04-echem-soc.htm (accessed 6 January 2011).

7 Morris, P.J.T. and Travis, A.S. (1992) A history of the international dyestuff industry. *Am. Dyestuff Rep.*, **81** (11). Reproduced in: http://www.colorantshistory.org/HistoryInternationalDyeIndustry.html (accessed 6 January 2011).

8 Institute Charles Gerhardt (2010) *Charles Gerhardt 1816–1856*, Institute Charles Gerhardt, Montpellier, http://www.icgm.fr/spip.php?article306 (accessed 6 January 2011).

9 Wikipedia (2010) Louis Pasteur http://en.wikipedia.org/wiki/Louis_Pasteur (accessed 6 January 2011).

10 Gilman, A.G., Goodman, L.S., Rall, T.W., and Murad, F. (1985) *The Pharmacological Basis of Therapeutics*, 7th edn, Macmillan, New York.

11 Papac, R.J. (2001) Origins of cancer therapy. *Yale J. Biol. Med.*, **74**, 391–398. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2588755/pdf/yjbm00015-0028.pdf (accessed 6 January 2011).

12 Association of the British Pharmaceutical Industry (2010) Facts & Statistics from the Pharmaceutical Industry: Pharmaceuticals and the UK Economy. ABPI, http://www.abpi.org.uk (accessed 6 January 2011).

13 Griminger, P. (1972) Casimir Funk, a biographical sketch (1884–1967). *J. Nutr.*, **102**, 1105–1114.

http://jn.nutrition.org/cgi/reprint/102/9/1105.pdf (accessed 6 January 2011).
14 Rosenfeld, L. (1997) Vitamine–vitamin. The early years of discovery. *Clin. Chem.*, **43** (4), 680–685. http://www.clinchem.org/cgi/reprint/43/4/680 (accessed 6 January 2011).
15 Pauling, L., Itano, H., Singer, S.J., and Wells, I. (1949) *Science*, **110** (2865), 543–548. http://www.sciencemag.org/cgi/content/citation/110/2865/543 (accessed 6 January 2011).
16 Crick, F., Watson, J., and Wilkins, M. (2010) The Nobel Prize in Physiology or Medicine 1962. Nobelprize.org. http://nobelprize.org/nobel_prizes/medicine/laureates/1962/index.html (accessed 6 January 2011).
17 Sanger, F. (2010) The Nobel Prize in Chemistry 1958. Nobelprize.org. http://nobelprize.org/nobel_prizes/chemistry/laureates/1958/ (accessed 6 January 2011).
18 Perutz, M.F. and Kendrew, J.C. (2010) The Nobel Prize in Chemistry 1962. Nobelprize.org. http://nobelprize.org/nobel_prizes/chemistry/laureates/1962/kendrew-bio.html (accessed 6 January 2011).
19 Ferracane, J.L. (2001) *Materials in Dentistry: Principles and Applications*, 2nd edn, Lippincott Williams and Wilkins, New York.
20 Smil, V. (2001) *Enriching the Earth: Fritz Haber, Carl Bosch, and the Transformation of World Food Production*, MIT.
21 Hager, T. (2008) *The Alchemy of Air*, Harmony Books, New York.
22 Wikipedia (2010) Haber process http://en.wikipedia.org/wiki/Haber_process#cite_note-13.
23 Centres for Disease Control and Prevention (2011) *The History of Malaria, An Ancient Disease*, Centres for Disease Control, Atlanta, http://www.cdc.gov/malaria/history/index.htm (accessed 6 January 2011).
24 Müller, P. (2010) The Nobel Prize in Physiology or Medicine, 1948. Nobelprize.org. http://nobelprize.org/nobel_prizes/medicine/laureates/1948 (accessed 6 January 2011).
25 Hartley, D. and Kidd, H. (1983) *The Agrochemicals Handbook*, Royal Society of Chemistry, Nottingham, UK.
26 Senchenkova, E.M. (1976) Tsvet (or Tswett), Mikhail Semenovich (1872–1919), in *Dictionary of Scientific Biography*, vol. 13 (ed. C.C. Gillispie), American Council of Learned Societies, Charles Scribner & Sons, New York, pp. 486–488.
27 Wikipedia (2010) William Herschel http://en.wikipedia.org/wiki/William_Herschel (accessed 6 January 2011).
28 Ingle, J.D.J. and Crouch, S.R. (1988) *Spectrochemical Analysis*, Prentice Hall, Englewood Cliffs, NJ.
29 Röntgen, W.C. (2010) The Nobel Prize in Physics, 1901. Nobelprize.org. http://nobelprize.org/nobel_prizes/physics/laureates/1901/rontgen-bio.html (accessed 6 January 2011).
30 Thomson, J.J. (2010) The Nobel Prize in Physics, 1906. Nobelprize.org. http://nobelprize.org/nobel_prizes/physics/laureates/1906/thomson.html (accessed 6 January 2011).
31 Rabi, I.I. (2010) The Nobel Prize in Physics, 1944. Nobelprize.org. http://nobelprize.org/nobel_prizes/physics/laureates/1944 (accessed 6 January 2011).
32 The Charles Goodyear Story. (2010) Goodyear Company, http://www.goodyear.com/corporate/history/history_story.html (accessed 6 January 2011).
33 Kauffman, G.B. (2011) Rubber, *Chemistry Explained*. http://www.chemistryexplained.com/Ru-Sp/Rubber.html (accessed 6 January 2011).
34 Biography of Alexander Parkes (1813–1890). (2010) yourdictionary.com, http://www.yourdictionary.com/biography/alexander-parkes (accessed 6 January 2011).
35 Leo Hendrik Baekeland, 1863–1944. (2010) Chemical Achievers: The Human Face of the Chemical Sciences. The Chemical Heritage Foundation, http://www.chemheritage.org/classroom/chemach/plastics/baekeland.html (accessed 6 January 2011).
36 Wallace Carothers. (2010) National Historic Chemical Landmaks. American Chemical Society, http://acswebcontent.acs.org/landmarks/nylon/carothers.html (accessed 6 January 2011).

37 Hamilton, J. (2004) *A Life of Discovery: Michael Faraday, Giant of the Scientific Revolution*, Random House, New York.

38 Maddison, A. (ed.) (2001) HS–8: the world economy, 1–2001 AD, in *The World Economy: A Millennial Perspective*, OECD, Paris, pp. 241–263, http://www.ggdc.net/maddison/ (accessed 6 January 2011).

39 World Intellectual Property Organization (WIPO) (2010) Medium-Term Strategic Plan 2010–15: Consultation Paper. http://www.wipo.int/export/sites/www/about-wipo/en/pdf/mtsp.pdf (accessed 6 January 2011).

40 Dye, C. (2010) Is wealth good for your health? *Gresham Lecture Notes*, http://www.gresham.ac.uk/uploads/Dye%207%20Wealth-Health.ppt (accessed 6 January 2011).

41 Easterlin, R. (1999) How benevolent is the market? A look at the modem history of mortality. *Eur. Rev. Econ. Hist.*, **3**, 257–294.

42 Jamison, D., Sandbu, M.E., and Wang, J. (2004) Why has infant mortality decreased at such different rates in different countries? Disease Control Priorities Project, Working Paper No. 21.

43 Jamison, D., Breman, J.G., Measham, A.R., Alleyne, G., Claeson, M., Evans, D.B., Jha, P., Mills, A., and Musgrove, P. (eds) (2006) *Disease Control Priorities in Developing Countries (DCP2)*, 2nd edn, World Bank, Washington, DC, http://files.dcp2.org/pdf/DCP/DCPFM.pdf (accessed 6 January 2011).

44 United Nations (2000) Millennium Summit, New York, http://www.un.org/millennium/summit.htm (accessed 6 January 2011).

45 UNDP (2010) Millennium Development Goals. http://www.undp.org/mdg/basics.shtml (accessed 6 January 2011).

46 World Bank (2010) *What is poverty?* World Bank, Washington, DC, http://www.thequietworld.com/ahealthyworld/index.php?page=poverty (accessed 6 January 2011).

47 Juma, C. and Lee, Y.-C. (2005) *Task Force on Science, Technology and Innovation. Innovation: Applying Knowledge in Development. UN Millennium Project*, Earthscan, London, http://www.unmillenniumproject.org/documents/Science-complete.pdf (accessed 6 January 2011).

48 United Nations (2010) *Millennium Development Goals Report 2010*, United Nations, New York, http://bit.ly/bmwAhK (accessed 6 January 2011).

49 Nature Editorial (2010) How to feed a hungry world, *Nature*, **466**, 531–532. http://www.nature.com/nature/journal/v466/n7306/full/466531a.html (accessed 6 January 2011).

50 UNESCO Institute for Statistics (2005) *Children out of School: Measuring Exclusion from Primary Education*, UIS, Montreal, http://www.uis.unesco.org/template/pdf/educgeneral/OOSC_EN_WEB_FINAL.pdf (accessed 6 January 2011).

51 Universal Declaration of Human Rights (1948) adopted by UN General Assembly, http://www.un.org/en/documents/udhr/index.shtml (accessed 6 January 2011).

52 Convention on the Elimination of All Forms of Discrimination against Women (CEDAW) (1979), adopted by UN General Assembly, http://www.un.org/womenwatch/daw/cedaw/cedaw.htm (accessed 6 January 2011).

53 *Beijing Declaration and Platform for Action* (1995) Fourth World Conference on Women, Beijing, http://www.un.org/womenwatch/daw/beijing/platform (accessed 6 January 2011).

54 International Institute for Sustainable Development (1994) Programme of action. International Conference on Population and Development Cairo, 5–13 September, http://www.iisd.ca/cairo.html (accessed 6 January 2011).

55 UNESCO Institute for Statistics (2009) A Global Perspective on Research and Development. UIS Fact Sheet, No. 2. http://www.uis.unesco.org/template/pdf/S&T/Factsheet_No2_ST_2009_EN.pdf (accessed 6 January 2011).

56 Hogan, M.C. *et al.* (2010) *Building Momentum: Global Progress Toward Reducing Maternal and Child Mortality*, IHME, Seattle, http://www.healthmetricsandevaluation.org/resources/policyreports/2010/building_momentum_0610.html (accessed 6 January 2011).

57 Report of the United Nations Conference on Environment and Development, (1992) Rio de Janeiro, http://www.un.org/esa/sustdev (accessed 6 January 2011).

58 *Johannesburg Declaration*, (2002) World Summit on Sustainable Development, Johannesburg, http://www.un-documents.net/jburgdec.htm#fn1 (accessed 6 January 2011).

59 IPCC (2007) Fourth Assessment Report. Intergovernmental Panel on Climate Change, http://www.ipcc.ch (accessed 6 January 2011).

60 United Nations Human Settlements Programme (2009) *State of the World's Cities 2006/7*, Earthscan, London, http://www.unhabitat.org/pmss/listItemDetails.aspx?publicationID=2101 (accessed 6 January 2011).

61 ELDIS (2010) *Access to Medicines and International Issues*, Institute of Development Studies, Sussex, http://www.eldis.org/go/topics/resource-guides/health-systems/access-to-medicines-and-international-issues (accessed 6 January 2011).

62 Solvay Company: History. (2010) http://www.solvay.com (accessed 6 January 2011).

63 Essenscia (2010) Belgium: A World Champion for Chemicals and Plastics. Belgian Federation for Chemistry and Life Sciences Industries, http://www.essenscia.be/01/MyDocuments/WORLD_CHAMPION_BAN_030310.pdf (accessed 6 January 2011).

64 Institut Pasteur International Network. (2010) http://www.pasteur-international.org/ip/easysite/pasteur-international-en/institut-pasteur-international-network/the-network (accessed 6 January 2011).

65 Indian Institute of Science (2009) Celebrating 100 Years of Achievement, http://www.iisc.ernet.in/ (accessed 6 January 2011).

66 South African Chemical Institute (2010) http://www.saci.co.za/ (accessed 6 January 2011).

67 Oswaldo Cruz Foundation (Fiocruz), Brazil (2010) http://www.fiocruz.br/cgi/cgilua.exe/sys/start.htm?UserActiveTemplate=template%5Fingles&infoid=2297&sid=281 (accessed 6 January 2011).

68 Fiocruz: Farmanguinhos Medicines and Drugs Technology Institute (2010) http://www.fiocruz.br/cgi/cgilua.exe/sys/start.htm?infoid=820&sid=122&UserActiveTemplate=template_ingles (accessed 6 January 2011).

69 Rubber Research Institute of Sri Lanka (2008) http://www.rrisl.lk/sub_pags/aboutus_home.html (accessed 6 January 2011).

70 Rubber Research Institute of Nigeria (2010) http://www.icpsr.org.ma/?Page=showInstitute&InstituteID=RRIN123&CountryID=Nigeria (accessed 6 January 2011).

71 Rubber Research Institute of Malaya (2010) http://sejarahmalaysia.pnm.my/portalBI/detail.php?section=sm05&spesifik_id=47&ttl_id=15 (accessed 6 January 2011).

72 Rubber Research Institute of India (2005) http://www.irikerala.org/ (accessed 6 January 2011).

73 UN Agenda for Development, (2005) adopted by the General Assembly, Science and Technology: Paras 72–75. http://www.un.org/Docs/SG/ecodev.htm (accessed 6 January 2011).

74 UNESCO (1945) Constitution of the United Nations Educational, Scientific, and Cultural Organization (UNESCO), adopted 16 November 1945, http://www.icomos.org/unesco/unesco_constitution.html (accessed 6 January 2011).

75 Dixon, D. (2005) World Bank Puts Science Back on the Agenda. http://www.scidev.net/en/news/world-bank-puts-science-back-on-the-agenda.html (accessed 6 January 2011).

76 Swedish International Development Cooperation Agency (SIDA) (2010) Secretariat for Research Cooperation. http://www.sida.se/English/Partners/Universities-and-research/From-funding-research-to-fighting-poverty/About-FORSKSEK/ (accessed 6 January 2011).

77 United Nations Economic and Social Commission for Asia and the Pacific (2010) Statistical Yearbook for Asia and the Pacific 2009. Chapter 1: Research

and Development, 95–97. http://www.unescap.org/stat/data/syb2009/15-Research-and-development.asp (accessed 6 January 2011).
78 Leadbeater, C. and Wilsdon, J. (2007) *The Atlas of Ideas: How Asian Innovation Can Benefit Us All*, Demos, London, http://www.demos.co.uk/files/Overview_Final.pdf (accessed 6 January 2011).
79 African Union/New Partnership for Africa's Development (NEPAD) (2005) Africa's Science and Technology Consolidated Plan of Action. http://www.africa-union.org/root/AU/Conferences/2010/March/Amcost/docs/Africa's%20Consolidated%20Plan%20of%20Action.pdf (accessed 6 January 2011).
80 UNESCO Natural Sciences Sector (2010) *Science for Sustainable Development*, UNESCO, Paris, http://www.unesco.org/science/psd/publications/rep_usa_eur_05.shtml (accessed 6 January 2011).
81 United Nations Educational, Scientific and Cultural Organization (2010) *International Science Prizes*, UNESCO, Paris, http://www.unesco.org/science/intern_prizes.shtml (accessed 6 January 2011).
82 UNDP (2007) Globalization and the least developed countries: issues in technology. United Nations Ministerial Conference of the Least Developed Countries: Making Globalization Work for the LDCs. Istanbul, 9–11 July, http://www.un.int/turkey/1.pdf (accessed 6 January 2011).
83 Academy of Sciences for the Developing World (TWAS) (2010) http://twas.ictp.it/ (accessed 6 January 2011).
84 Organization of Women in Science for the Developing World (2011) formerly Third World Organization for Women in Science (TWOWS), http://twows.ictp.it/ (accessed 6 January 2011).
85 International Association of Science and Technology for Development (IASTED) (2011) http://www.iasted.org (accessed 6 January 2011).
86 *United Nations Millennium Declaration*. (2000) UN General Assembly, New York, http://www.un.org/millennium/declaration/ares552e.pdf (accessed 6 January 2011).
87 Commission on Macroeconomics and Health (2001) *Macroeconomics and Health: Investing in Health for Economic Development*, WHO, Geneva, http://whqlibdoc.who.int/publications/2001/924154550x.pdf (accessed 6 January 2011).
88 Juma, C. (ed.) (2005) *Going for Growth: Science, Technology and Innovation in Africa*, The Smith Institute, London.
89 Juma, C. (2005) We need to reinvent the African university. SciDevNet, 14 June, http://www.scidev.net/en/opinions/we-need-to-reinvent-the-african-university.html (accessed 6 January 2011).
90 Coober, D.I., Langer, S.S., and Pratt, J.M. (eds) (1992) *Chemistry and Developing Countries*, Commonwealth Science Council/Royal Society of Chemistry, London.
91 Chemical Research Applied to World Needs (CHEMRAWN) (2010) http://www.iupac.org/web/ins/021 (accessed 6 January 2011).
92 Malin, J.M. (2006) History and Effectiveness of CHEMRAWN Conferences, 1978–2006. IUPAC, http://old.iupac.org/standing/chemrawn/CR_Histrory_061027.pdf (accessed 6 January 2011).
93 Crabbé, P. and Cardyn, L. (1983) *The Time for Another World*, University Printing Services, Columbia, MO.
94 Crabbé, P. (1983) A new challenge for the university. *Interciencia*, **8**, 279.
95 Crabbé, P., Archer, S., Benagiano, G., Diczfalusy, E., Djerassi, C., Fried, J., and Higuchi, T. (1983) Long-acting contraceptive agents: design of the WHO Chemical Synthesis Programme. *Steroids*, **41**, 243–253. http://www.ncbi.nlm.nih.gov/pubmed/6658872 (accessed 6 January 2011).
96 Seaborg, G.T. (1984) An international effort in chemical science. *Science*, **223**, 9. http://www.sciencemag.org/cgi/pdf_extract/223/4631/9 (accessed 6 January 2011).
97 Matlin, S.A., Chan, L., Hadjigeogiou, P., Prazeres, M.A., Mehani, S., and Roshdi, S. (1983) Long-acting contraceptive agents: analysis and purification of steroid esters. *Steroids*, **41**, 361–367. http://www.ncbi.nlm.nih.gov/

pubmed/6419409 (accessed 6 January 2011).

98 Matlin, S.A., Chan, L., Prazeres, M.A., Mehani, S., and Cass, Q.B. (1987) Long-acting androgens: analytical and preparative HPLC of testosterone esters. *J. High Resolut. Chromatogr. Chromatogr. Commun.*, **10**, 186–190.

99 Waites, G.M.H. (2003) Development of methods of male contraception: impact of the World Health Organization Task Force. *Fertility and Sterility*, **80**, 1–15. http://www.fertstert.org/article/S0015-0282(03)00577-6/abstract (accessed 6 January 2011).

100 Crabbé, P., Diczfalusy, E., and Djerassi, C. (1980) Injectable contraceptive synthesis: an example of international cooperation. *Science*, **209**, 992–994. http://www.ncbi.nlm.nih.gov/pubmed/7403868 (accessed 6 January 2011).

101 History of the International Organization for Chemical Sciences in Development (2007) http://www.iocd.org/ourhistory.shtml (accessed 6 January 2011).

102 Inter-Academy Council Study Panel (2003) Inventing a better future. IAC Report, http://www.interacademycouncil.net/CMS/Reports/9866/9430.aspx?returnID=9866 (accessed 6 January 2011).

103 Matlin, S.A., Landriault, E., and Monot, J.-J. (2009) The 2009 Report Card on financing research and development for health, in *Monitoring Financial Flows for Health Research 2009: Behind the Global Numbers* (eds E. Landriault and S.A. Matlin), Global Forum for Health Research, Geneva, pp. 153–193, http://www.globalforumhealth.org/Media-Publications/Publications/Monitoring-Financial-Flows-for-Health-Research-2009-Behind-the-Global-Numbers (accessed 6 January 2011).

104 Eurostat (2010) R & D Expenditure. Eurostat Statistics: Main Tables. European Commission-Eurostat, http://epp.eurostat.ec.europa.eu/portal/page/portal/science_technology_innovation/data/main_tables (accessed 6 January 2011).

105 Nordling, L. (2010) Big spending on science promised for East Africa. SciDevNet, 11 June, http://www.scidev.net/en/news/big-spending-on-science-promised-for-east-africa.html (accessed 6 January 2011).

106 National Advisory Council on Innovation Indicators Reference Group (2008) South African Science and Technology Indicators 2008, South Africa, http://www.nacinnovation.biz/south-african-science-and-technology-indicators-2008/ (accessed 6 January 2011).

107 New Partnership for Africa's Development (NEPAD) (2010) African Science, Technology & Innovation Indicators Initiative (ASTII), http://www.nepadst.org/astii/index.shtml (accessed 6 January 2011).

108 Obama, B. (2009) Speech to US National Academy of Sciences, 27 April, http://www.whitehouse.gov/the_press_office/Remarks-by-the-President-at-the-National-Academy-of-Sciences-Annual-Meeting/ (accessed 6 January 2011).

109 Council for Scientific and Industrial Research (2005) *The Tshwane Consensus on Science and Development* CSIR, Pretoria. Media release, 19 October, http://ntww1.csir.co.za/plsql/ptl0002/PTL0002_PGE013_MEDIA_REL?MEDIA_RELEASE_NO=7323626 (accessed 6 January 2011).

110 Esteves, B. (2006) Brazil to boost health research capacity in Angola. SciDevNet, 7 August, http://www.scidev.net/News/index.cfm?fuseaction=readNews&itemid=3028&language=1 (accessed 6 January 2011).

111 Worldmapper maps (2006) http://www.worldmapper.org (accessed 6 January 2011).

112 UNESCO Institute for Statistics (2005) What do Bibliometric Indicators Tell Us about World Scientific Output? UIS Bulletin on Science and Technology Statistics. UIS, Issue No. 2. http://www.uis.unesco.org/template/pdf/S&T/BulletinNo2EN.pdf (accessed 6 January 2011).

113 Pendlebury, D.A. (2008) *White Paper: Using Bibliometrics in Evaluating Research*, Thomson Reuters, Philadelphia, http://wokinfo.com/media/pdf/UsingBibliometricsinEval_WP.pdf (accessed 6 January 2011).

114 Bibliometrics Special Interest Group, Universiti Technologi MARA, Shah

Alam (2004) *Science and Technology Knowledge Productivity in Malysia 2003*, Malaysian Science and Technology Information Centre, Putrajaya, http://myais.fsktm.um.edu.my/590/ (accessed 6 January 2011).

115 Dumont, J.C. and Lemaître, G. (2005) Counting Immigrants and Expatriates: A New Perspective. OECD, Social Employment and Migration Working papers. OECD, http://www.oecd.org/dataoecd/27/5/33868740.pdf (accessed 6 January 2011).

116 Hunt, J. and Gauthier-Loiselle, M. (2008) How Much Does Immigration Boost Innovation? Working Paper No. 14312. National Bureau of Economic Research, Cambridge, http://www.nber.org/papers/w14312.pdf (accessed 6 January 2011).

117 UNDP (2009) *Human Development Report 2009. Overcoming Barriers: Human Mobility and Development*, Palgrave Macmillan, New York, http://hdr.undp.org/en/media/HDR_2009_EN_Complete.pdf (accessed 6 January 2011).

118 Chu, R. and Pugatch, M. (2010) From test tube to patient – national innovation strategies for the biomedical field. Stockholm Network, http://www.stockholm-network.org/downloads/publications/From_Test_Tube_to_Patient_Final.pdf (accessed 6 January 2011).

119 Silberglitt, R., Antón, P.S., Howell, D.R., and Wong, A. (2006) The Global Technology Revolution 2020: In-Depth Analyses. RAND Corporation, http://www.rand.org/pubs/technical_reports/2006/RAND_TR303.pdf (accessed 6 January 2011).

120 Bound, K. (2008) *Brazil: The Natural Knowledge Economy*, Demos, London, http://www.demos.co.uk/publications/brazil (accessed 6 January 2011).

121 Páscoa, M.B.A. (2005) *In Search of An Innovative Environment – the New Brazilian Innovation Law*, World Intellectual Property Organization, Geneva, http://www.wipo.int/sme/en/documents/brazil_innovation.htm (accessed 6 January 2011).

122 Veneu, F. (2004) Brazil adopts innovation law. SciDevNet, 20 December, http://www.scidev.net/en/news/brazil-adopts-innovation-law.html (accessed 6 January 2011).

123 Massarani, L. (2006) Editorial. Brazil's innovation law: lessons for Latin America. SciDevNet, 3 August, http://www.scidev.net/en/editorials/brazils-innovation-law-lessons-for-latin-america.html (accessed 6 January 2011).

124 Petherick, A. (2010) High hopes for Brazilian science. *Nature*, **465**, 674–675. http://www.nature.com/news/2010/100609/full/465674a.html (accessed 6 January 2011).

125 Ryan, M.P. (2010) Patent Incentives, Technology Markets, and Public-Private Bio-Medical Innovation Networks in Brazil. Creative and Innovative Economy Center Research Paper, http://www.law.gwu.edu/Academics/research_centers/ciec/Documents/research%20papers/GW%20CIEC%20Brazil%20patents%20bio-medical%20technology%20networks.pdf (accessed 6 January 2011).

126 Peng, K. (2009) China issues 50-year science strategy. SciDevNet, 6 July, http://www.scidev.net/en/news/china-issues-50-year-science-strategy-.html (accessed 6 January 2011).

127 World Trade Organization (2010) Understanding the WTO: Membership, Alliances and Bureaucracy. WTO, http://www.wto.org/english/thewto_e/whatis_e/tif_e/org3_e.htm (accessed 6 January 2011).

128 Shetty, P. (2010) BioMed analysis: India's patent catch-22. SciDevNet, 20 May, http://www.scidev.net/en/opinions/biomed-analysis-india-s-patent-catch-22.html (accessed 6 January 2011).

129 Nuyens, Y. (2005) *No Development without Research: A Challenge for Research Capacity Strengthening*, Global Forum for Health Research, Geneva, http://www.globalforumhealth.org/Media-Publications/Publications/No-Development-Without-Research-A-challenge-for-research-capacity-strengthening (accessed 6 January 2011).

130 Abegaz, B.M. (1998) The Status of basic sciences in Africa, in *Basic Sciences and Development: Rethinking Donor Policy* (eds M.J. Garett and C.G. Granqvist), Ashgate, Aldershot, pp. 73–102.

131 Swedish International Development Cooperation Agency (2009) International Conference on the Strengthening of Research and Higher Education in Basic Sciences by Regional and Interregional Cooperation; Relevance for Developing Countries. Addis Ababa, 1–4 September 2009, http://www2.math.uu.se/~leifab/addis/programme_pdf.pdf (accessed 6 January 2011).

132 Wu, T.K. (1992) Planning a chemical industry – Taiwan ROC: a case study, in *Chemistry and Developing Countries* (eds D.I. Coober, S.S. Langer, and J.M. Pratt), Commonwealth Science Council/Royal Society of Chemistry, London, pp. 141–146.

133 Development Cooperation Directorate (OECD-DAC) (2010) *The Paris Declaration and Accra Agenda for Action*, OECD, Paris, http://www.oecd.org/document/18/0,3343,en_2649_3236398_35401554_1_1_1_1,00.html (accessed 6 January 2011), http://www.oecd.org/dataoecd/11/41/34428351.pdf. (accessed 6 January 2011).

134 Australian Centre for International Agricultural Research (2010) *ACIAR and the Paris Declaration on Aid Effectiveness*, Australian Centre for International Agricultural Research, Canberra, http://aciar.gov.au/node/2405 (accessed 6 January 2011).

135 American Chemical Society (2011) http://www.acs.org (accessed 6 January 2011).

136 Royal Society of Chemistry (2011) http://www.rsc.org (accessed 6 January 2011).

137 Royal Australian Chemical Institute Inc. (2011) http://www.raci.org.au/ (accessed 6 January 2011).

138 Gesellschaft Österreichischer Chemiker (2011) http://www.goech.at/info_english.shtml (accessed 6 January 2011).

139 Brazilian Chemical Society (2011) http://www.sbq.org.br/ingles/history.php (accessed 6 January 2011).

140 Egyptian Chemical Society (ECS) (2009) http://www.egy-chem-soc.org/ (accessed 6 January 2011).

141 Société Chimique de France (2011) http://www.societechimiquedefrance.fr/ (accessed 6 January 2011).

142 Gesellschaft Deutscher Chemiker (2011) http://www.gdch.de/gdch__e.htm (accessed 6 January 2011).

143 The Chemical Society of Japan (2011) http://www.chemistry.or.jp/ (accessed 6 January 2011).

144 Koninklijke Nederlandse Chemische Vereniging (2011) http://www.kncv.nl/ (accessed 6 January 2011).

145 Norwegian Chemical Society (NKS) (2011) http://www.kjemi.no/english/ (accessed 6 January 2011).

146 Sociedade Portuguesa de Quimica (2009) http://www.spq.pt/ (accessed 6 January 2011).

147 European Association for Chemical and Molecular Sciences (2011) Mendeleev Russian Chemical Society, http://www.euchems.org/MemberSocieties/Mendeleev/ (accessed 6 January 2011).

148 Real Sociedad Española de Química (2011) http://www.rseq.org/ (accessed 6 January 2011).

149 Svenska Kemistsamfundet (2011) http://www.chemsoc.se/ (accessed 6 January 2011).

150 Swiss Chemical Society (SCS), Schweizerische Chemische Gesellschaft (SCG), Société Suisse de Chimie (SSC) (2011) http://www.swiss-chem-soc.ch/org (accessed 6 January 2011).

151 International Union of Pure and Applied Chemistry (IUPAC) (2011) http://www.iupac.org/ (accessed 6 January 2011).

152 European Association for Chemical and Molecular Sciences (EuCheMS) (2010) http://www.euchems.org/ (accessed 6 January 2011).

153 Federation of African Societies of Chemistry (2011) http://www.faschem.org (accessed 6 January 2011).

154 Federation of Asian Chemical Societies (FACS) (2011) http://www.facs-as.org/index.php?page=facs (accessed 6 January 2011).

155 Pan African Chemistry Network (2011) http://www.rsc.org/Membership/Networking/InternationalActivities/

PanAfrica/index.asp (accessed 6 January 2011).

156 Barabási, A.-L. (2005) Network theory – the emergence of the creative enterprise. *Science*, **308**, 639–641. http://www.sciencemag.org/cgi/content/short/308/5722/639 (accessed 6 January 2011).

157 Esparza, J. and Yamada, T. (2007) The discovery value of "big science". *J. Exp. Med.*, **204** (4), 701–704. http://ukpmc.ac.uk/backend/ptpmcrender.cgi?accid=PMC2118535&blobtype=pdf (accessed 6 January 2011).

158 Nordling, L. (2010) Africa analysis: professional societies need networking. SciDevNet, 29 January, http://www.scidev.net/en/opinions/africa-analysis-professional-societies-need-networking-1.html (accessed 6 January 2011).

159 International Science Programme (2010) http://www.isp.uu.se/ (accessed 6 January 2011).

160 International Foundation for Science (2011) http://www.ifs.se/Partners/networks.asp (accessed 6 January 2011).

161 AFASSA (Co-ordination of Networks for research on Biological Resources in Africa, Asia and South America) (2009) http://www.afassa.org (accessed 6 January 2011).

162 ANCAP (African Network for the Chemical Analysis of Pesticides) (2010) http://www.ancap.org/ (accessed 6 January 2011).

163 NABSA (Network for Analytical and Bio-assay Services in Africa) (2010) http://www.nabsaonline.org/ (accessed 6 January 2011).

164 NAPRECA (Natural Products Research Network for Eastern and Central Africa) (2011) http://www.napreca.net/ (accessed 6 January 2011).

165 NITUB (Network of Instrument Technical Personnel and User Scientists of Bangladesh) (2005) http://www.nitub.org/ (accessed 6 January 2011).

166 NUSESA (Network of Users of Scientific Equipment in Southern and Eastern Africa) (2008) http://www.ansti.org/new/index.php?option=com_content&task=view&id=313&Itemid=40 (accessed 6 January 2011).

167 WANNPRES (Western Africa Network of Natural Products Research Scientists) (2010) http://www.ifs.se/Partners/networks.asp (accessed 6 January 2011).

168 Abegaz, B.M. (2001) Promoting research and research-training in Eastern and Southern Africa through science-focused networks, societies and academies: an overview of progress and potentials. Paper presented at the Ford Foundation Conference on "Innovations in African Higher Education" October 1–3, 2001, Nairobi, Kenya.

169 Bringmann, G., Mutanyatta-Comar, J., Knauer, M., and Abegaz, B.M. Knipholone and related 4-phenylanthraquinones: structurally, pharmacologically, and biosynthetically remarkable natural products. *Nat. Prod. Rep.*, 2008, **25**, 696–718. http://pubs.rsc.org/en/Content/ArticleLanding/2008/NP/b803784c (accessed 6 January 2011).

170 Abegaz, B.M., Tekle-Haimanot, R., Palmer, V.S., and Spencer, P.S. (eds) (1994) *Nutrition, Neurotoxins and Lathyrism: The ODAP Challenge*, Third World Medical Research Foundation, New York.

171 Abegaz, B.M. Personal communication. Data from paper presented at Conference on Regional and Interregional Cooperation to Strengthen Basic Sciences in Developing Countries. Addis Ababa, 1–4 September 2009. http://www.math.uu.se/~leifab/addis/programme_pdf.pdf (accessed 6 January 2011); http://www.math.uu.se/~leifab/addis/abstracts/Abegaz%20IPICS%20NABSA.pdf (accessed 6 January 2011).

172 Masesane, I.B., Yeboah, S.O., Liebscher, J., Muegge, C., and Abegaz, B.M. (1999) A bichalcone from the twigs of *Rhus pyroides*. *Phytochemistry*, **53**, 1005–1008.

173 Mdee, L.K., Yeboah, S.O., and Abegaz, B.M. (2003) Rhuschalcones II-VI, five new bichalcones from the root bark of *Rhus pyroides*. *J. Nat. Prod.*, **66** (5), 599–604.

174 Southern and Eastern African Network for Analytical Chemists (2007) http://www.seanac.org (accessed 6 January 2011).

175 African Association of Pure and Applied Chemistry (2001) http://old.iupac.org/links/ao/aapac.html (accessed 6 January 2011).

176 Becker, E., Ober, C., and Henry, B. (2009) Chemistry research funding. *Chem. Int.*, **31**, 23–24. http://www.iupac.org/publications/ci/2009/3104/july09.pdf (accessed 6 January 2011).

177 Wikipedia (2010) Research funding http://en.wikipedia.org/wiki/Research_funding (accessed 6 January 2011).

178 History of the Royal Society (2011) http://royalsociety.org/History-of-the-Royal-Society/ (accessed 6 January 2011).

179 Royal Society of Chemistry (2006) *Funding Science and Technology: Who Pays? Who Benefits? Report of a Seminar Organised by the Royal Society of Chemistry at the House of Commons, London*, Royal Society of Chemistry, London. http://www.rsc.org/images/funding%20science%20report_tcm18-91112.pdf (accessed 6 January 2011).

180 Die Kaiser Wilhelm Gesellschaf (2010) http://www.dhm.de/lemo/html/kaiserreich/wissenschaft/kwg/index.html (accessed 6 January 2011).

181 Deutsche Forschungsgemeinschaft (2010) http://www.dfg.de (accessed 6 January 2011).

182 Fonds National de la Recherche Scientifique (2010) Notre histoire et nos statuts. FNRS. http://www2.frs-fnrs.be/index.php?option=com_content&view=article&id=76&Itemid=4&lang=fr (accessed 6 January 2011).

183 Centre National de la Recherche Scientifique (2011) http://www.cnrs.fr (accessed 6 January 2011).

184 Aux origines du CNRS (1989) *Colloque sur l'Histoire du CNRS des 23 et 24 octobre 1989*, CNRS, Paris, http://www.histcnrs.fr/origines_cnrs_1.html (accessed 6 January 2011).

185 Standler, R.B. (2009) Funding of Research in Basic Science in the USA. 1–13. http://www.rbs0.com/funding.pdf (accessed 6 January 2011).

186 Lopez-Claros, A. and Zahidi, S. (2005) Women's Empowerment: Measuring the Global Gender Gap. World Economic Forum, http://www.weforum.org/pdf/Global_Competitiveness_Reports/Reports/gender_gap.pdf (accessed 6 January 2011).

187 Jenkins, E.W. (1997) Gender and science & technology education. *Educ. Newsl.*, **23** (1). UNESCO, Paris, http://www.unesco.org/education/educprog/ste/newsletter/eng_n1/gender.html (accessed 6 January 2011).

188 Inter-Academy Council Study Panel (2003) Inventing a Better Future. Ch. 3: Expanding human resources. IAC Report, 43–60. http://www.interacademycouncil.net/Object.File/Master/6/742/Chapter%203.pdf (accessed 6 January 2011).

189 Directory of Open Access Journals. (2010) Lund University Libraries, http://www.doaj.org/doaj?func=loadTempl&templ=about (accessed 6 January 2011).

190 Royal Society of Chemistry (2006) *Chemistry in the Developing World*, RSC, London, http://www.rsc.org/ScienceAndTechnology/Policy/Bulletins/Issue3/ChemDeveloping World.asp (accessed 6 January 2011).

191 Trager, R. (2007) Chemistry's Open Access Dilemma. Chemistry World, http://www.rsc.org/chemistryworld/Issues/2007/December/ChemistrysOpenAccessDilemma.asp (accessed 6 January 2011).

192 Bueno, R. (2009) The inova success story – technology transfer in Brazil. WIPO Magazine, November. http://www.wipo.int/wipo_magazine/en/2009/06/article_0009.html (accessed 6 January 2011).

193 Teng, H. (2010) University-industry technology transfer: framework and constraints. *J. Sust. Dev.*, **3**, 296–299. http://www.ccsenet.org/journal/index.php/jsd/article/viewFile/6345/5102 (accessed 6 January 2011).

194 Ganguli, P. (2005) Industry-Academic Interaction in Technology Transfer and IPR. The Indian Scene – An Overview. WIPO, http://www.wipo.int/export/sites/www/uipc/en/partnership/pdf/ui_partnership_in.pdf (accessed 6 January 2011).

195 al-Hassan, A.Y. (2005) *Transfer of Islamic Technology to the West. Part III: Technology Transfer in the Chemical Industries – Transmission of Practical Chemistry*, IRCICA, Istanbul, http://www.history-science-technology.com/Articles/articles%2072.htm#_edn1 (accessed 6 January 2011).

196 Okoso-Amaa, K. and Mapima, C. (1995) Technology transfer and acquisition of managerial capability in Tanzania, in *Technology Policy and Practice in Africa* (eds O.M. Ogbu, B.O. Oyeyinka, and H.M. Mlawa), IDRC, Ottawa, Chapter 7. http://www.idrc.ca/en/ev-9301-201-1-DO_TOPIC.html (accessed 6 January 2011).

197 Government of Tanzania (2010) Science and technology. Tanzania National Website, http://www.tanzania.go.tz/science_technology.html (accessed 6 January 2011).

198 International Federation of Pharmaceutical Manufacturers & Associations (2010) *Technology Transfer and Anti-Retroviral Licensing in Developing Countries*, IFPMA, Geneva, https://www.ifpma.org/index.php?id=3878 (accessed 6 January 2011).

199 Glaxo SmithKline (2007) Technology transfer, capacity building and the developing world Glaxo SmithKline, http://www.gsk.com/policies/GSK-on-technology-transfer-capacity-building.pdf (accessed 6 January 2011).

200 United Nations Industrial Development Organization (UNIDO) (2011) http://www.unido.org (accessed 6 January 2011).

201 Diamant, R., Davison, H., and Pugatch, M.P. (2007) Promoting technology transfer in developing countries: lessons from public-private partnerships in the field of pharmaceuticals. Stockholm Network, http://www.stockholm-network.org/downloads/publications/Promoting_Technology_Transfer_1.pdf (accessed 6 January 2011).

202 Wikipedia (2010) Glocalisation – definition http://en.wikipedia.org/wiki/Glocalisation (accessed 6 January 2011).

203 See, for example, *AMF Americas* Network website: http://www.amfamericasnetwork.com/index.html (accessed 6 January 2011).

204 Gaud, W.S. (1968) The Green Revolution: Accomplishments and Apprehensions. Speech to the Society for International Development, 8 March, http://www.agbioworld.org/biotech-info/topics/borlaug/borlaug-green.html (accessed 6 January 2011).

205 Norman Borlaug: The Nobel Peace Prize 1970. (2010) Nobelprize.org, http://nobelprize.org/nobel_prizes/peace/laureates/1970/ (accessed 6 January 2011).

206 Borlaug, N. (2002) Biotechnology and the green revolution. An ActionBioscience.org original interview, http://www.actionbioscience.org/biotech/borlaug.html (accessed 6 January 2011).

207 Strengthening Food Security: Alliance for a Green Revolution in Africa (AGRA) (2010) Rockefeller Foundation, http://www.rockefellerfoundation.org/what-we-do/current-work/strengthening-food-security-alliance/ (accessed 6 January 2011).

208 Pachauri, R.K. and Reisinger, A. (eds) (2007) *Climate Change 2007: Synthesis Report. Contribution of Working Groups I, II and III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*, IPCC, Geneva, http://www.ipcc.ch/publications_and_data/ar4/syr/en/contents.html (accessed 6 January 2011).

209 Shahab, S., Ghaffar, A., Stearns, B.P., and Woodward, A. (2008) *Strengthening the Base: Preparing Health Research for Climate Change*, Global Forum for Health Research, Geneva, http://www.globalforumhealth.org/Media-Publications/Publications/Strengthening-the-base-preparing-health-research-for-climate-change (accessed 6 January 2011).

210 Campbell-Lendrum, D. and Woodruff, R. (2006) Comparative risk assessment of the burden of disease from climate change. *Environ. Health Perspect.*, **114** (12), 1935–1941. http://www.bvsde.paho.org/bvsacd/cd67/comparative.pdf (accessed 6 January 2011).

211 Third World Network (2008) Developing countries call for new technology transfer mechanism. TWN Bonn News Update 4, 6 June. http://www.twnside.org.sg/title2/climate/news/TWNbonnupdate4.doc (accessed 6 January 2011).

212 Data from World Bank Development Indicators (2010). http://data.worldbank.org/indicator (accessed 6 January 2011).

213 *Report of the United Nations Conference on the Human Environment, Stockholm, 5–16 June* (1972). United Nations Publication, Sales No. E.73.II.A.14 and corrigendum.

214 Solomon, T., Åkerblom, M., and Thulstrup, E.W. (2003) Chemistry in the developing world. *Anal. Chem.*, **75** (5), 107A–113A. http://pubs.acs.org/doi/pdfplus/10.1021/ac0312458 (accessed 6 January 2011).

215 Matlin, S.A. (2006) The changing scene in *Monitoring Financial Flows for Health Research 2006: The Changing Landscape of Health Research for Development* (eds A. de Francisco and S.A. Matlin), Global Forum for Health Research, Geneva, pp. 3–32, http://www.globalforumhealth.org/Media-Publications/Publications/Monitoring-Financial-Flows-for-Health-Research-2006-The-changing-landscape-of-health-research-for-development (accessed 6 January 2011).

216 International Pharmaceutical Federation (2003) Statement of Policy on Counterfeit Medicines. IFPMA, http://www.fip.org/www/uploads/database_file.php?id=164&table_id= (accessed 6 January 2011).

217 Wikipedia (2010) Pandemic http://en.wikipedia.org/wiki/Pandemic (accessed 6 January 2011).

218 United Nations Population Funds (2009) *State of World Population 2009*, UNFPA, New York, http://www.unfpa.org/public/publications/search_pubs/swpreports (accessed 6 January 2011).

219 United Nations Human Settlements Program (2010) *State of the World's Cities 2010/2011 – Cities for All: Bridging the Urban Divide*, UN-Habitat, Nairobi, http://www.unhabitat.org/pmss/listItemDetails.aspx?publicationID=2917 (accessed 6 January 2011).

220 World Intellectual Property Organization (2011) http://www.wipo.int (accessed 6 January 2011).

221 Krattiger, A. *et al.* (ed.) (2009) *Intellectual Property Management in Health and Agricultural Innovation: Handbook of Best Practices. Executive Guide*, 2nd edn, MIHR, PIPRA, Oswaldo Cruz Foundation, and bioDevelopments-International Institute, http://www.iphandbook.org/handbook/about/attachment_files/ipHandbook%20Executive%20Guide%20Final.pdf (accessed 6 January 2011).

222 World Trade Organization (WTO) Overview: The TRIPS Agreement, http://www.wto.org/english/tratop_e/trips_e/intel2_e.htm (accessed 6 January 2011).

223 Musungu, S.F. and Dutfield, G. (2003) *Multilateral Agreements and A TRIPS-Plus World: The World Intellectual Property Organisation (WIPO)*, Quaker United Nations Office (QUNO), Geneva, http://www.quno.org/geneva/pdf/economic/Issues/Multilateral-Agreements-in-TRIPS-plus-English.pdf (accessed 6 January 2011).

224 Berger, J. and Prabhala, A. (2005) *Assessing the Impact of TRIPS-Plus Patent Rules in the Proposed US-SACU Free Trade Agreement*, WHO, Geneva, http://www.who.int/hiv/amds/capacity/tza2_oxfamreport_pricing_financing.pdf (accessed 6 January 2011).

225 Collins-Chase, C.T. (2008) The case against TRIPS-Plus protection in developing countries facing aids epidemics. *University of Pennsylvania Law School J*, **29**, 763–802. http://www.law.upenn.edu/journals/jil/articles/volume29/issue3/CollinsChase29U.Pa.J.Int'lL.763(2008).pdf (accessed 6 January 2011).

226 Commission on Intellectual Property Rights, Innovation and Public Health (2006) *Report of the Commission on Intellectual Property Rights, Innovation and Public Health*, World Health Organization, Geneva, http://www.who.int/intellectualproperty/en/ (accessed 6 January 2011).

227 World Health Organization (2008) *WHO Global Strategy and Plan of Action on Public Health, Innovation and Intellectual Property*, Health Organization, Geneva, http://www.who.int/phi/en/ (accessed 6 January 2011).

228 Eisner, T. (1994) Chemical prospecting: a global imperative. *Proc. Am. Philos. Soc.*, **138**, 385–395. http://www.jstor.org/pss/986744 (accessed 6 January 2011).

229 National Biodiversity Institute (INBio) of Costa Rica (2010) http://www.inbio.ac.cr/en/inbio/inb_queinbio.htm (accessed 6 January 2011).

230 Eberlee, J. (2000) Assessing the benefits of bioprospecting in Latin America. IDRC Reports Online, 21 January, http://www.idrc.ca/en/ev-5571-201-1-DO_TOPIC.html (accessed 6 January 2011).

231 Jenner, A. (2010) Experiences with bio-prospecting and access & benefit sharing. Presentation at the Geneva Pharma Forum, 6 May, http://www.ifpma.org/fileadmin/webnews/2010/pdfs/20100506_IFPMA_GPF_ABS_05052010_AJEN.pdf (accessed 6 January 2011).

232 Weiss, C. and Eisner, T. (1998) Partnerships for value-added through bioprospecting. *Technology in Society*, **20**, 481–498. http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V80-3VGSGFS-R&_user=10&_coverDate=11%2F30%2F1998&_rdoc=1&_fmt=high&_orig=search&_sort=d&_docanchor=&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=4036d4f2568177cd6839609f1435ccb1 (accessed 6 January 2011).

233 International Cooperative Biodiversity Groups Program (2011) http://www.icbg.org/ (accessed 6 January 2011).

234 Rosenthal, J. (1997) Integrating drug discovery, biodiversity conservation, and economic development: early lessons from the International Cooperative Biodiversity Groups, in *Biodiversity and Human Health* (eds F. Grifo and J. Rosenthal), Island Press, Washington DC, Ch. 13, http://www.icbg.org/pub/documents/lessons.pdf (accessed 6 January 2011).

235 IOCD: Working Group on Biotic Exploration (2009) http://www.iocd.org/bioticexpfund.shtml (accessed 6 January 2011).

236 World Health Organization (2008) *UN-Water Global Annual Assessment of Sanitation and Drinking-Water (GLAAS)*, WHO, Geneva, http://www.who.int/water_sanitation_health/glaas_2008_pilot_finalreport.pdf (accessed 6 January 2011).

237 United Nations (2008) *The Millennium Development Goals Report*, UN, New York, http://mdgs.un.org/unsd/mdg/Resources/Static/Products/Progress2008/MDG_Report_2008_En.pdf#page=42 (accessed 6 January 2011).

238 World Water Assessment Programme (2009) *Water in A Changing World. Third UN World Water Development Report*, UNESCO, Paris and Earthscan, London, http://www.unesco.org/water/wwap/wwdr/wwdr3/ (accessed 6 January 2011).

239 World Health Organization (2006) *Meeting the MDG Drinking-Water and Sanitation Target: the Urban and Rural Challenge of the Decade*, WHO, Geneva and UNICEF, Paris, http://www.who.int/water_sanitation_health/monitoring/jmpfinal.pdf (accessed 6 January 2011).

240 Botting, M.J., Porbeni, E.O., Joffres, M.R., Johnston, B.C., Black, R.E., and Mills, E.J. (2010) Water and sanitation infrastructure for health: the impact of foreign aid. *Global Health*, **6** (12), 1–8. http://www.globalizationandhealth.com/content/pdf/1744-8603-6-12.pdf (accessed 6 January 2011).

241 UN-WATER (2008) *Work Programme, 2008-9*, UN, New York, http://www.unwater.org/downloads/unw_workplan_2008.pdf (accessed 6 January 2011).

242 Pan African Chemistry Network (2010) *Africa's Water Quality – A Chemical Science Perspective*, PACN/Royal Society of Chemistry, London, http://www.rsc.org/Membership/Networking/InternationalActivities/PanAfrica/waterreport.asp (accessed 6 January 2011).

243 Alliance for Chemical Sciences and Technology in Europe (2010) Chemistry: Europe & the Future. AllChemE, http://www.cefic.org/allcheme/ (accessed 6 January 2011).

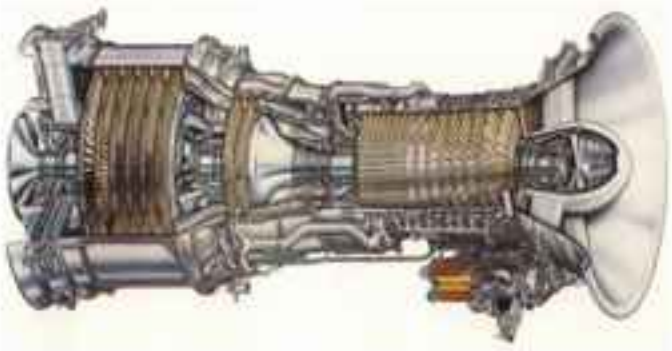
244 Conway, G. and Waage, J. (2010) *Science and Innovation for Development*, UK Collaborative on Development Sciences, London, http://www.ukcds.org.uk/publication-Science_and_Innovation_for_Develop%3E%20ment-172.html (accessed 6 January 2011).

REVISED NINTH EDITION

A Textbook of

FLUID MECHANICS AND HYDRAULIC MACHINES

S.I. Units



Dr. R.K. Bansal

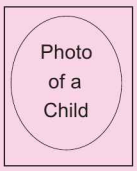
A TEXTBOOK OF FLUID MECHANICS AND HYDRAULIC MACHINES

WIN A CASH AWARD OF Rs. 500.00

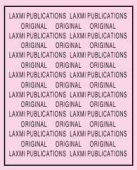
It has come to our notice that some booksellers are fraudulently **selling fake/duplicate copies** of some of our fast selling titles.

In our sincere efforts to provide you with our genuine books and to protect you against these counterfeit books, Laxmi Publications (LP) has put a Hologram on the cover of **some of its fast selling** titles. The Hologram displays a unique 3D multi-level, multi-colour effect from different angles. It has the following three levels of flat graphics merged together. The background artwork seems to be 'under' or 'behind' the Hologram and gives the illusion of depth unlike the fake Hologram on the fake/duplicate books.

Channel 1



Channel 2


Background

Overlay

Presently, only some titles have got the Holograms. In this case, **A Text Book of Fluid Mechanics and Hydraulic Machines** (2010 edition, priced at Rs. 450.00) has got the Hologram.

If you or any of your friends finds anywhere in India/abroad any book of this Edition without the LP Hologram, he/she is requested to write to us at **M/s LAXMI PUBLICATIONS PVT. LTD., 113, Golden House, Daryaganj, New Delhi-110002,** giving the name and address of the bookseller from where he/she purchased this book, together with the photocopy of the cover and the 2nd page on which the price of the book and name of the printer are printed. He/She will be sent a **cash award** of **Rs. 500.00.**

How to decide if the book is genuine or fake ?

1. The above information may or may not be printed.
2. The counterfeit edition of the book may have no LP Hologram or if it has, it will be without the illusionary depth as described above.

What is the harm in purchasing duplicate books ?

— Poor quality of paper and printing which affect your eyes.

— No royalty to authors who are scholars and have put their hard labour in writing the book, thus depriving them of their intellectual rights.

Warning : Selling or buying pirated books is an offence. Legal action shall be taken against the bookseller(s) and student(s) or whoever found guilty of such an offence in any way.

A TEXTBOOK OF

FLUID MECHANICS

AND

HYDRAULIC MACHINES

(In S.I. Units)

[For Degree, U.P.S.C. (Engg. Services), A.M.I.E. (India)]

By
Dr. R.K. BANSAL

B. Sc. Engg. (Mech.), M.Tech. Hons. (I.I.T., Delhi)
Ph. D., M.I.E. (India)
Formerly
Professor, Department of Mechanical Engineering,
Dean (U.G. Studies), Delhi College of Engineering, Delhi

LAXMI PUBLICATIONS (P) LTD

BANGALORE • CHENNAI • COCHIN • GUWAHATI • HYDERABAD
JALANDHAR • KOLKATA • LUCKNOW • MUMBAI • RANCHI
NEW DELHI

Published by :
LAXMI PUBLICATIONS (P) LTD
113, Golden House, Daryaganj,
New Delhi-110002

Phone : 011-43 53 25 00
Fax : 011-43 53 25 28

www.laxmipublications.com
info@laxmipublications.com

Author : **Dr. R.K. Bansal**
Compiled by : **Smt. Nirmal Bansal**

Price : **Rs. 495.00** *Only.*

First Edition : Sept. 1983
Ninth Edition : 2005
Reprint : 2006, 2007, 2008, 2009
Revised Ninth Edition : 2010

OFFICES

✆ **Bangalore**	080-26 61 15 61	✆ **Chennai**	044-24 34 47 26
✆ **Cochin**	0484-237 70 04, 405 13 03	✆ **Guwahati**	0361-254 36 69, 251 38 81
✆ **Hyderabad**	040-24 65 23 33	✆ **Jalandhar**	0181-222 12 72
✆ **Kolkata**	033-22 27 43 84	✆ **Lucknow**	0522-220 95 78
✆ **Mumbai**	022-24 91 54 15, 24 92 78 69	✆ **Ranchi**	0651-221 47 64

EFM-0559-495-FLUID MECHANICS & HM-BAN **C—**

Typesetted at : Shubham Composer, New Delhi. *Printed at :* Repro India Ltd., Mumbai.

Dedicated
to
The loving memory
of
my daughter, Babli

PREFACE TO THE NINTH EDITION

The popularity of the eighth edition and reprints of the book **A Textbook of Fluid Mechanics and Hydraulic Machines** amongst the students and the teachers of the various Universities of the country, has prompted the bringing out of the ninth edition of the book so soon. The ninth edition has been thoroughly revised and brought up-to-date. A large number of problems from different B.E. degree examinations of Indian Universities and other examining bodies such as Institution of Engineers and U.P.S.C. upto Summer 2002 examinations have been selected and have been solved at proper places on this edition. Most of these problems have been worked out in S.I. units. All of the text along with existing problems have been converted into S.I. Units.

In the ninth edition, a new chapter entitled Ideal Flow (or Potential Flow) has been added. Potential flow has been included in most of Indian Universities. This chapter has been written in a simple and easy-to-follow language so that even an average student can grasp the subject matter by self-study. Also a few new topics such as "Liquids in Relative Equilibrium" and "Pipe Network" have been added in this edition. The topic of Pipe Network has been included in the chapter of Flow Through Pipes. The pipe network is mostly used in city water supply system, Laboratory supply system or house hold supply of water and gas.

The objective type multiple-choice questions are often asked in the various competitive examinations. Hence a large number of objective type questions with answers have been added in the end of the book.

With these additions, it is hoped that the book will be quite useful for the students of different branches of Engineering at various Engineering Institutions.

I express my sincere thanks to my colleagues, friends, students and the teachers of different Indian Universities for their valuable suggestions and recommending the book of their students.

Suggestions for the improvement of this book are most welcome and would be incorporated in the next edition with a view to make the book more useful.

– Author

PREFACE TO THE FIRST EDITION

I am glad to present the book entitled, **A Textbook of Fluid Mechanics and Hydraulic Machines** to the engineering students of mechanical, civil, electrical, aeronautical and chemical and also to the students preparing for the new scheme of Section B of A.M.I.E. Examination of Institution of Engineers (India). The course contents have been planned in such a way that the general requirements of all engineering students are fulfilled.

During my long experience of teaching this subject to undergraduate and post-graduate engineering students for the past 16 years, I have observed that the students face difficulty in understanding clearly the basic principles, fundamental concepts and theory without adequate solved problems along with the text. To meet this very basic requirement to the students, a large number of the questions taken from the examinations of the various Universities of India and from other professional and competitive examinations (such as Institution of Engineers and U.P.S.C. Engineering Service Examination) have been solved along with the text in M.K.S. and S.I. units.

The book is written in a simple and easy-to-follow language, so that even an average students can grasp the subject by self-study. At the end of each chapter highlights, theoretical questions and many unsolved numerical problems with answer are given for the students to solve them.

I am thankful to my colleagues, friends and students who encouraged me to write this book. I am grateful to Institution of Engineers (India), various Universities of India and those authorities whose work have been consulted and gave me a great help in preparing the book.

I express my appreciation and gratefulness to my publisher. Shri R.K. Gupta (as Mechanical Engineer) for his most co-operative, painstaking attitude and untiring efforts for bringing out the book in a short period.

Mrs. Nirmal Bansal deserves special credit as she not only provided an ideal atmosphere at home for book writing but also gave inspiration and valuable suggestions.

Though every care has been taken in checking the manuscripts and proof reading, yet claiming perfection is very difficult. I shall be very grateful to the readers and users of this book for pointing any mistakes that might have crept in. Suggestions for improvement are most welcome and would be incorporated in the next edition with a view to make the book more useful.

– Author

CONTENTS

(B) IDEAL FLOW (POTENTIAL FLOW)

1 CHAPTER

PROPERTIES OF FLUIDS

▶ 1.1 INTRODUCTION

Fluid mechanics is that branch of science which deals with the behaviour of the fluids (liquids or gases) at rest as well as in motion. Thus this branch of science deals with the static, kinematics and dynamic aspects of fluids. The study of fluids at rest is called fluid statics. The study of fluids in motion, where pressure forces are not considered, is called fluid kinematics and if the pressure forces are also considered for the fluids in motion, that branch of science is called fluid dynamics.

▶ 1.2 PROPERTIES OF FLUIDS

1.2.1 Density or Mass Density. Density or mass density of a fluid is defined as the ratio of the mass of a fluid to its volume. Thus mass per unit volume of a fluid is called density. It is denoted by the symbol ρ (rho). The unit of mass density in SI unit is kg per cubic metre, *i.e.*, kg/m^3. The density of liquids may be considered as constant while that of gases changes with the variation of pressure and temperature.

Mathematically, mass density is written as

$$\rho = \frac{\text{Mass of fluid}}{\text{Volume of fluid}}.$$

The value of density of water is 1 gm/cm^3 or 1000 kg/m^3.

1.2.2 Specific Weight or Weight Density. Specific weight or weight density of a fluid is the ratio between the weight of a fluid to its volume. Thus weight per unit volume of a fluid is called weight density and it is denoted by the symbol w.

Thus mathematically,
$$w = \frac{\text{Weight of fluid}}{\text{Volume of fluid}} = \frac{(\text{Mass of fluid}) \times \text{Acceleration due to gravity}}{\text{Volume of fluid}}$$

$$= \frac{\text{Mass of fluid} \times g}{\text{Volume of fluid}}$$

$$= \rho \times g \qquad \left\{\because \frac{\text{Mass of fluid}}{\text{Volume of fluid}} = \rho\right\}$$

$$\therefore \quad w = \rho g \qquad \ldots(1.1)$$

The value of specific weight or weight density (w) for water is 9.81×1000 Newton/m^3 in SI units.

1.2.3 Specific Volume. Specific volume of a fluid is defined as the volume of a fluid occupied by a unit mass or volume per unit mass of a fluid is called specific volume. Mathematically, it is expressed as

$$\text{Specific volume} = \frac{\text{Volume of fluid}}{\text{Mass of fluid}} = \frac{1}{\dfrac{\text{Mass of fluid}}{\text{Volume of fluid}}} = \frac{1}{\rho}$$

Thus specific volume is the reciprocal of mass density. It is expressed as m^3/kg. It is commonly applied to gases.

1.2.4 Specific Gravity. Specific gravity is defined as the ratio of the weight density (or density) of a fluid to the weight density (or density) of a standard fluid. For liquids, the standard fluid is taken water and for gases, the standard fluid is taken air. Specific gravity is also called relative density. It is dimensionless quantity and is denoted by the symbol S.

$$\text{Mathematically, } S(\text{for liquids}) = \frac{\text{Weight density (density) of liquid}}{\text{Weight density (density) of water}}$$

$$S(\text{for gases}) = \frac{\text{Weight density (density) of gas}}{\text{Weight density (density) of air}}$$

Thus weight density of a liquid $= S \times$ Weight density of water

$= S \times 1000 \times 9.81$ N/m^3

The density of a liquid $= S \times$ Density of water

$= S \times 1000$ kg/m^3. ...(1.1*A*)

If the specific gravity of a fluid is known, then the density of the fluid will be equal to specific gravity of fluid multiplied by the density of water. For example, the specific gravity of mercury is 13.6, hence density of mercury = $13.6 \times 1000 = 13600$ kg/m^3.

Problem 1.1 *Calculate the specific weight, density and specific gravity of one litre of a liquid which weighs 7 N.*

Solution. Given :

$$\text{Volume} = 1 \text{ litre} = \frac{1}{1000} \text{ m}^3 \quad \left(\because \ 1 \text{ litre} = \frac{1}{1000} \text{ m}^3 \text{ or } 1 \text{ litre} = 1000 \text{ cm}^3\right)$$

$$\text{Weight} = 7 \text{ N}$$

(*i*) Specific weight (w) $= \dfrac{\text{Weight}}{\text{Volume}} = \dfrac{7 \text{ N}}{\left(\dfrac{1}{1000}\right) \text{m}^3} =$ **7000 N/m^3. Ans.**

(*ii*) Density (ρ) $= \dfrac{w}{g} = \dfrac{7000}{9.81}$ kg/m^3 = **713.5 kg/m^3. Ans.**

(*iii*) Specific gravity $= \dfrac{\text{Density of liquid}}{\text{Density of water}} = \dfrac{713.5}{1000}$ $\{\because$ Density of water = 1000 kg/m$^3\}$

= **0.7135. Ans.**

Problem 1.2 *Calculate the density, specific weight and weight of one litre of petrol of specific gravity = 0.7*

Solution. Given : Volume = 1 litre = $1 \times 1000\ cm^3 = \frac{1000}{10^6}\ m^3 = 0.001\ m^3$

Sp. gravity $S = 0.7$

(*i*) *Density* (ρ)

Using equation (1.1*A*),

Density (ρ) $= S \times 1000\ kg/m^3 = 0.7 \times 1000 =$ **700 kg/m^3. Ans.**

(*ii*) *Specific weight* (w)

Using equation (1.1), $w = \rho \times g = 700 \times 9.81\ N/m^3 =$ **6867 N/m^3. Ans.**

(*iii*) *Weight* (W)

We know that specific weight $= \frac{\text{Weight}}{\text{Volume}}$

or $w = \frac{W}{0.001}$ or $6867 = \frac{W}{0.001}$

$\therefore$ $W = 6867 \times 0.001 =$ **6.867 N. Ans.**

▶ 1.3 VISCOSITY

Viscosity is defined as the property of a fluid which offers resistance to the movement of one layer of fluid over another adjacent layer of the fluid. When two layers of a fluid, a distance '*dy*' apart, move one over the other at different velocities, say u and $u + du$ as shown in Fig. 1.1, the viscosity together with relative velocity causes a shear stress acting between the fluid layers.

The top layer causes a shear stress on the adjacent lower layer while the lower layer causes a shear stress on the adjacent top layer. This shear stress is proportional to the rate of change of velocity with respect to y. It is denoted by symbol τ (Tau).

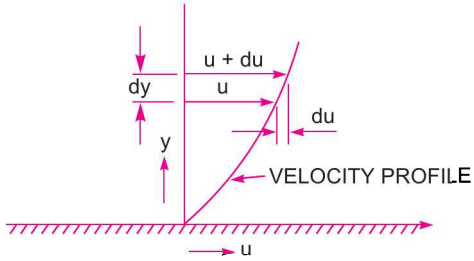


Fig. 1.1 *Velocity variation near a solid boundary.*

Mathematically, $\tau \propto \frac{du}{dy}$

or $\tau = \mu \frac{du}{dy}$ ...(1.2)

where μ (called *mu*) is the constant of proportionality and is known as the co-efficient of dynamic viscosity or only viscosity. $\frac{du}{dy}$ represents the rate of shear strain or rate of shear deformation or velocity gradient.

From equation (1.2), we have $\mu = \frac{\tau}{\left(\frac{du}{dy}\right)}$ ...(1.3)

Thus viscosity is also defined as the shear stress required to produce unit rate of shear strain.

1.3.1 Units of Viscosity. The units of viscosity is obtained by putting the dimensions of the quantities in equation (1.3)

$$\mu = \frac{\text{Shear stress}}{\frac{\text{Change of velocity}}{\text{Change of distance}}} = \frac{\text{Force/Area}}{\left(\frac{\text{Length}}{\text{Time}}\right) \times \frac{1}{\text{Length}}}$$

$$= \frac{\text{Force/(Length)}^2}{\frac{1}{\text{Time}}} = \frac{\text{Force} \times \text{Time}}{(\text{Length})^2}$$

In MKS system, force is represented by kgf and length by metre (m), in CGS system, force is represented by dyne and length by cm and in SI system force is represented by Newton (N) and length by metre (m).

$\therefore$ MKS unit of viscosity $= \frac{\text{kgf-sec}}{\text{m}^2}$

CGS unit of viscosity $= \frac{\text{dyne-sec}}{\text{cm}^2}$

In the above expression N/m^2 is also known as Pascal which is represented by Pa. Hence N/m^2 = Pa = Pascal

$\therefore$ SI unit of viscosity = Ns/m^2 = Pa s.

SI unit of viscosity $= \frac{\text{Newton-sec}}{\text{m}^2} = \frac{\text{Ns}}{\text{m}^2}$

The unit of viscosity in CGS is also called Poise which is equal to $\frac{\text{dyne-sec}}{\text{cm}^2}$.

The numerical conversion of the unit of viscosity from MKS unit to CGS unit is given below :

$$\frac{\text{one kgf-sec}}{\text{m}^2} = \frac{9.81\ \text{N-sec}}{\text{m}^2} \qquad \{\because\ 1\ \text{kgf} = 9.81\ \text{Newton}\}$$

But one Newton = one kg (mass) $\times$ one $\left(\frac{\text{m}}{\text{sec}^2}\right)$ (acceleration)

$$= \frac{(1000\ \text{gm}) \times (100\ \text{cm})}{\text{sec}^2} = 1000 \times 100\ \frac{\text{gm-cm}}{\text{sec}^2}$$

$$= 1000 \times 100\ \text{dyne} \qquad \left\{\because\ \text{dyne} = \text{gm} \times \frac{\text{cm}}{\text{sec}^2}\right\}$$

$\therefore$ $$\frac{\text{one kgf-sec}}{\text{m}^2} = 9.81 \times 100000\ \frac{\text{dyne-sec}}{\text{cm}^2} = 9.81 \times 100000\ \frac{\text{dyne-sec}}{100 \times 100 \times \text{cm}^2}$$

$$= 98.1\ \frac{\text{dyne-sec}}{\text{cm}^2} = 98.1\ \text{poise} \qquad \left\{\because\ \frac{\text{dyne-sec}}{\text{cm}^2} = \text{Poise}\right\}$$

Thus for solving numerical problems, if viscosity is given in poise, it must be divided by 98.1 to get its equivalent numerical value in MKS.

But $$\frac{\text{one kgf-sec}}{\text{m}^2} = \frac{9.81\ \text{Ns}}{\text{m}^2} = 98.1\ \text{poise}$$

$\therefore$ $$\frac{\text{one Ns}}{\text{m}^2} = \frac{98.1}{9.81}\ \text{poise} = 10\ \text{poise} \quad \text{or} \quad \text{One poise} = \frac{1}{10}\ \frac{\text{Ns}}{\text{m}^2}.$$

Alternate Method. One poise $= \dfrac{\text{dyne} \times \text{s}}{\text{cm}^2} = \left(\dfrac{1\text{ gm} \times 1\text{ cm}}{\text{s}^2}\right) \times \dfrac{\text{s}}{\text{cm}^2}$

But dyne $= 1\text{ gm} \times \dfrac{1\text{ cm}}{\text{s}^2}$

$\therefore$ One poise $= \dfrac{1\text{ gm}}{\text{s cm}} = \dfrac{\frac{1}{1000}\text{ kg}}{\text{s}\frac{1}{100}\text{ m}}$

$= \dfrac{1}{1000} \times 100\ \dfrac{\text{kg}}{\text{sm}} = \dfrac{1}{10}\dfrac{\text{kg}}{\text{sm}}$ or $1\dfrac{\text{kg}}{\text{sm}} = 10$ poise.

Note. (*i*) In SI units second is represented by 's' and not by 'sec'.

(*ii*) If viscosity is given in poise, it must be divided by 10 to get its equivalent numerical value in SI units. Sometimes a unit of viscosity as centipoise is used where

$$1 \text{ centipoise} = \frac{1}{100} \text{ poise} \quad \text{or} \quad 1 \text{ cP} = \frac{1}{100} \text{ P} \qquad [\text{cP} = \text{Centipoise, P} = \text{Poise}]$$

The viscosity of water at 20°C is 0.01 poise or 1.0 centipoise.

1.3.2 Kinematic Viscosity. It is defined as the ratio between the dynamic viscosity and density of fluid. It is denoted by the Greek symbol (ν) called 'nu'. Thus, mathematically,

$$\nu = \frac{\text{Viscosity}}{\text{Density}} = \frac{\mu}{\rho} \qquad ...(1.4)$$

The units of kinematic viscosity is obtained as

$$\nu = \frac{\text{Units of } \mu}{\text{Units of } \rho} = \frac{\text{Force} \times \text{Time}}{(\text{Length})^2 \times \dfrac{\text{Mass}}{(\text{Length})^3}} = \frac{\text{Force} \times \text{Time}}{\dfrac{\text{Mass}}{\text{Length}}}$$

$$= \frac{\text{Mass} \times \dfrac{\text{Length}}{(\text{Time})^2} \times \text{Time}}{\left(\dfrac{\text{Mass}}{\text{Length}}\right)} \qquad \left\{\because \text{Force} = \text{Mass} \times \text{Acc.} = \text{Mass} \times \frac{\text{Length}}{\text{Time}^2}\right\}$$

$$= \frac{(\text{Length})^2}{\text{Time}}.$$

In MKS and SI, the unit of kinematic viscosity is metre^2/sec or m^2/sec while in CGS units it is written as cm^2/s. In CGS units, kinematic viscosity is also known as stoke.

Thus, one stoke $= \text{cm}^2/\text{s} = \left(\dfrac{1}{100}\right)^2 \text{m}^2/\text{s} = 10^{-4}\text{ m}^2/\text{s}$

Centistoke means $= \dfrac{1}{100}$ stoke.

1.3.3 Newton's Law of Viscosity. It states that the shear stress (τ) on a fluid element layer is directly proportional to the rate of shear strain. The constant of proportionality is called the co-efficient of viscosity. Mathematically, it is expressed as given by equation (1.2) or as

$$\tau = \mu \frac{du}{dy}.$$

Fluids which obey the above relation are known as **Newtonian** fluids and the fluids which do not obey the above relation are called **Non-Newtonian** fluids.

1.3.4 Variation of Viscosity with Temperature. Temperature affects the viscosity. The viscosity of liquids decreases with the increase of temperature while the viscosity of gases increases with the increase of temperature. This is due to reason that the viscous forces in a fluid are due to cohesive forces and molecular momentum transfer. In liquids, the cohesive forces predominates the molecular momentum transfer, due to closely packed molecules and with the increase in temperature, the cohesive forces decreases with the result of decreasing viscosity. But in case of gases the cohesive forces are small and molecular momentum transfer predominates. With the increase in temperature, molecular momentum transfer increases and hence viscosity increases. The relation between viscosity and temperature for liquids and gases are:

(*i*) **For liquids,** $\mu = \mu_0 \left(\frac{1}{1 + \alpha t + \beta t^2} \right)$ ...(1.4*A*)

where μ = Viscosity of liquid at t°C, in poise

μ_0 = Viscosity of liquid at 0°C, in poise

α, β = Constants for the liquid

For water, $\mu_0 = 1.79 \times 10^{-3}$ poise, $\alpha = 0.03368$ and $\beta = 0.000221$.

Equation (1.4*A*) shows that with the increase of temperature, the viscosity decreases.

(*ii*) **For a gas,** $\mu = \mu_0 + \alpha t - \beta t^2$ ...(1.4*B*)

where for air $\mu_0 = 0.000017$, $\alpha = 0.000000056$, $\beta = 0.1189 \times 10^{-9}$.

Equation (1.4*B*) shows that with the increase of temperature, the viscosity increases.

1.3.5 Types of Fluids. The fluids may be classified into the following five types :

1. Ideal fluid,
2. Real fluid,
3. Newtonian fluid,
4. Non-Newtonian fluid, and
5. Ideal plastic fluid.

1. Ideal Fluid. A fluid, which is incompressible and is having no viscosity, is known as an ideal fluid. Ideal fluid is only an imaginary fluid as all the fluids, which exist, have some viscosity.

2. Real Fluid. A fluid, which possesses viscosity, is known as real fluid. All the fluids, in actual practice, are real fluids.

3. Newtonian Fluid. A real fluid, in which the shear stress is directly proportional to the rate of shear strain (or velocity gradient), is known as a Newtonian fluid.

4. Non-Newtonian Fluid. A real fluid, in which the shear stress is not proportional to the rate of shear strain (or velocity gradient), known as a Non-Newtonian fluid.

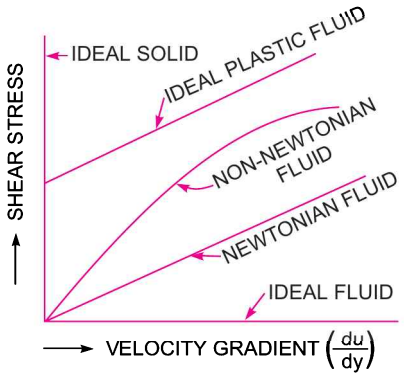


Fig. 1.2 *Types of fluids.*

5. Ideal Plastic Fluid. A fluid, in which shear stress is more than the yield value and shear stress is proportional to the rate of shear strain (or velocity gradient), is known as ideal plastic fluid.

Problem 1.3 *If the velocity distribution over a plate is given by* $u = \frac{2}{3} y - y^2$ *in which* u *is the velocity in metre per second at a distance* y *metre above the plate, determine the shear stress at* $y = 0$ *and* $y = 0.15$ *m. Take dynamic viscosity of fluid as 8.63 poises.*

Solution. Given : $u = \frac{2}{3}y - y^2 \quad \therefore \quad \frac{du}{dy} = \frac{2}{3} - 2y$

$$\left(\frac{du}{dy}\right)_{\text{at } y=0} \text{ or } \left(\frac{du}{dy}\right)_{y=0} = \frac{2}{3} - 2(0) = \frac{2}{3} = 0.667$$

Also $$\left(\frac{du}{dy}\right)_{\text{at } y=0.15} \text{ or } \left(\frac{du}{dy}\right)_{y=0.15} = \frac{2}{3} - 2 \times .15 = .667 - .30 = 0.367$$

Value of $\mu = 8.63$ poise $= \frac{8.63}{10}$ SI units $= 0.863$ N s/m^2

Now shear stress is given by equation (1.2) as $\tau = \mu \frac{du}{dy}$.

(*i*) Shear stress at $y = 0$ is given by

$$\tau_0 = \mu \left(\frac{du}{dy}\right)_{y=0} = 0.863 \times 0.667 = \mathbf{0.5756\ N/m^2.\ Ans.}$$

(*ii*) Shear stress at $y = 0.15$ m is given by

$$(\tau)_{y=0.15} = \mu \left(\frac{du}{dy}\right)_{y=0.15} = 0.863 \times 0.367 = \mathbf{0.3167\ N/m^2.\ Ans.}$$

Problem 1.4 *A plate 0.025 mm distant from a fixed plate, moves at 60 cm/s and requires a force of 2 N per unit area i.e., 2 N/m^2 to maintain this speed. Determine the fluid viscosity between the plates.*

Solution. Given :

Distance between plates, $dy = .025$ mm $= .025 \times 10^{-3}$ m

Velocity of upper plate, $u = 60$ cm/s $= 0.6$ m/s

Force on upper plate, $F = 2.0 \frac{\text{N}}{\text{m}^2}$.

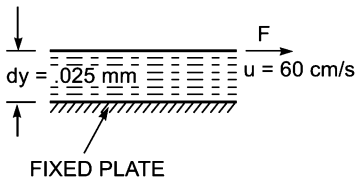


Fig. 1.3

This is the value of shear stress *i.e.*, τ

Let the fluid viscosity between the plates is μ.

Using the equation (1.2), we have $\tau = \mu \frac{du}{dy}$.

where du = Change of velocity $= u - 0 = u = 0.60$ m/s

dy = Change of distance $= .025 \times 10^{-3}$ m

τ = Force per unit area $= 2.0 \frac{\text{N}}{\text{m}^2}$

$$\therefore \quad 2.0 = \mu \frac{0.60}{.025 \times 10^{-3}} \quad \therefore \quad \mu = \frac{2.0 \times .025 \times 10^{-3}}{0.60} = 8.33 \times 10^{-5} \frac{\text{Ns}}{\text{m}^2}$$

$$= 8.33 \times 10^{-5} \times 10 \text{ poise} = \mathbf{8.33 \times 10^{-4}\ poise.\ Ans.}$$

Problem 1.5 *A flat plate of area 1.5 × 10^6 mm^2 is pulled with a speed of 0.4 m/s relative to another plate located at a distance of 0.15 mm from it. Find the force and power required to maintain this speed, if the fluid separating them is having viscosity as 1 poise.*

Solution. Given :

Area of the plate, $A = 1.5 \times 10^6 \text{ mm}^2 = 1.5 \text{ m}^2$

Speed of plate relative to another plate, $du = 0.4$ m/s

Distance between the plates, $dy = 0.15 \text{ mm} = 0.15 \times 10^{-3}$ m

Viscosity $\mu = 1 \text{ poise} = \frac{1}{10} \frac{\text{Ns}}{\text{m}^2}$.

Using equation (1.2) we have $\tau = \mu \frac{du}{dy} = \frac{1}{10} \times \frac{0.4}{.15 \times 10^{-3}} = 266.66 \frac{\text{N}}{\text{m}^2}$

(*i*) $\therefore$ Shear force, $F = \tau \times \text{area} = 266.66 \times 1.5 =$ **400 N. Ans.**

(*ii*) Power* required to move the plate at the speed 0.4 m/sec

$= F \times u = 400 \times 0.4 =$ **160 W. Ans.**

Problem 1.6 *Determine the intensity of shear of an oil having viscosity = 1 poise. The oil is used for lubricating the clearance between a shaft of diameter 10 cm and its journal bearing. The clearance is 1.5 mm and the shaft rotates at 150 r.p.m.*

Solution. Given : $\mu = 1 \text{ poise} = \frac{1}{10} \frac{\text{N s}}{\text{m}^2}$

Dia. of shaft, $D = 10 \text{ cm} = 0.1 \text{ m}$

Distance between shaft and journal bearing,

$dy = 1.5 \text{ mm} = 1.5 \times 10^{-3}$ m

Speed of shaft, $N = 150$ r.p.m.

Tangential speed of shaft is given by

$$u = \frac{\pi DN}{60} = \frac{\pi \times 0.1 \times 150}{60} = 0.785 \text{ m/s}$$

Using equation (1.2), $\tau = \mu \frac{du}{dy}$,

where du = change of velocity between shaft and bearing $= u - 0 = u$

$$= \frac{1}{10} \times \frac{0.785}{1.5 \times 10^{-3}} = \textbf{52.33 N/m}^2\textbf{. Ans.}$$

Problem 1.7 *Calculate the dynamic viscosity of an oil, which is used for lubrication between a square plate of size 0.8 m × 0.8 m and an inclined plane with angle of inclination 30° as shown in Fig. 1.4. The weight of the square plate is 300 N and it slides down the inclined plane with a uniform velocity of 0.3 m/s. The thickness of oil film is 1.5 mm.*

Solution. Given :

Area of plate, $A = 0.8 \times 0.8 = 0.64 \text{ m}^2$

Angle of plane, $\theta = 30°$

Weight of plate, $W = 300$ N

Velocity of plate, $u = 0.3$ m/s

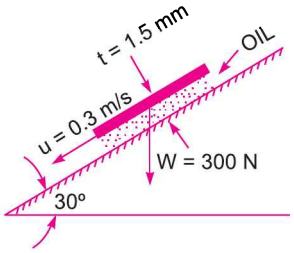


Fig. 1.4

* Power = $F \times u$ N m/s = $F \times u$ W ($\because$ Nm/s = Watt)

Thickness of oil film, $t = dy = 1.5 \text{ mm} = 1.5 \times 10^{-3} \text{ m}$

Let the viscosity of fluid between plate and inclined plane is μ.

Component of weight W, along the plane = $W \cos 60° = 300 \cos 60° = 150$ N

Thus the shear force, F, on the bottom surface of the plate = 150 N

and shear stress, $$\tau = \frac{F}{\text{Area}} = \frac{150}{0.64} \text{ N/m}^2$$

Now using equation (1.2), we have

$$\tau = \mu \frac{du}{dy}$$

where du = change of velocity = $u - 0 = u = 0.3$ m/s

$$dy = t = 1.5 \times 10^{-3} \text{ m}$$

$$\therefore \quad \frac{150}{0.64} = \mu \frac{0.3}{1.5 \times 10^{-3}}$$

$$\therefore \quad \mu = \frac{150 \times 1.5 \times 10^{-3}}{0.64 \times 0.3} = 1.17 \text{ N s/m}^2 = 1.17 \times 10 = \mathbf{11.7 \text{ poise. Ans.}}$$

Problem 1.8 *Two horizontal plates are placed 1.25 cm apart, the space between them being filled with oil of viscosity 14 poises. Calculate the shear stress in oil if upper plate is moved with a velocity of 2.5 m/s.*

Solution. Given :

Distance between plates, $dy = 1.25 \text{ cm} = 0.0125 \text{ m}$

Viscosity, $\mu = 14 \text{ poise} = \frac{14}{10} \text{ N s/m}^2$

Velocity of upper plate, $u = 2.5$ m/sec.

Shear stress is given by equation (1.2) as, $\tau = \mu \frac{du}{dy}$

where du = Change of velocity between plates = $u - 0 = u = 2.5$ m/sec.

$$dy = 0.0125 \text{ m}.$$

$$\therefore \quad \tau = \frac{14}{10} \times \frac{2.5}{.0125} = \mathbf{280 \text{ N/m}^2. \text{ Ans.}}$$

Problem 1.9 *The space between two square flat parallel plates is filled with oil. Each side of the plate is 60 cm. The thickness of the oil film is 12.5 mm. The upper plate, which moves at 2.5 metre per sec requires a force of 98.1 N to maintain the speed. Determine :*

(i) the dynamic viscosity of the oil in poise, and

(ii) the kinematic viscosity of the oil in stokes if the specific gravity of the oil is 0.95.

Solution. Given :

Each side of a square plate = 60 cm = 0.60 m

$\therefore$ Area, $A = 0.6 \times 0.6 = 0.36 \text{ m}^2$

Thickness of oil film, $dy = 12.5 \text{ mm} = 12.5 \times 10^{-3} \text{ m}$

Velocity of upper plate, $u = 2.5$ m/sec

$\therefore$ Change of velocity between plates, $du = 2.5$ m/sec

Force required on upper plate, $F = 98.1$ N

$\therefore$ Shear stress, $\tau = \dfrac{\text{Force}}{\text{Area}} = \dfrac{F}{A} = \dfrac{98.1\,\text{N}}{0.36\,\text{m}^2}$

(*i*) Let μ = Dynamic viscosity of oil

Using equation (1.2), $\tau = \mu \dfrac{du}{dy}$ or $\dfrac{98.1}{0.36} = \mu \times \dfrac{2.5}{12.5 \times 10^{-3}}$

$\therefore$ $\mu = \dfrac{98.1}{0.36} \times \dfrac{12.5 \times 10^{-3}}{2.5} = 1.3635 \dfrac{\text{Ns}}{\text{m}^2}$ $\left(\because \dfrac{1\,\text{Ns}}{\text{m}^2} = 10 \text{ poise}\right)$

$= 1.3635 \times 10 =$ **13.635 poise. Ans.**

(*ii*) Sp. gr. of oil, $S = 0.95$

Let ν = kinematic viscosity of oil

Using equation (1.1*A*),

Mass density of oil, $\rho = S \times 1000 = 0.95 \times 1000 = 950$ kg/m^3

Using the relation, $\nu = \dfrac{\mu}{\rho}$, we get $\nu = \dfrac{1.3635\left(\dfrac{\text{Ns}}{\text{m}^2}\right)}{950} = .001435$ m^2/sec $= .001435 \times 10^4$ cm^2/s

$=$ **14.35 stokes. Ans.** ($\because$ cm^2/s = stoke)

Problem 1.10 *Find the kinematic viscosity of an oil having density 981 kg/m^3. The shear stress at a point in oil is 0.2452 N/m^2 and velocity gradient at that point is 0.2 per second.*

Solution. Given :

Mass density, $\rho = 981$ kg/m^3

Shear stress, $\tau = 0.2452$ N/m^2

Velocity gradient, $\dfrac{du}{dy} = 0.2$ s

Using the equation (1.2), $\tau = \mu \dfrac{du}{dy}$ or $0.2452 = \mu \times 0.2$

$\therefore$ $\mu = \dfrac{0.2452}{0.200} = 1.226$ Ns/m^2

Kinematic viscosity ν is given by

$\therefore$ $\nu = \dfrac{\mu}{\rho} = \dfrac{1.226}{981} = .125 \times 10^{-2}$ m^2/sec

$= 0.125 \times 10^{-2} \times 10^4$ cm^2/s $= 0.125 \times 10^2$ cm^2/s

$= 12.5$ cm^2/s = **12.5 stoke. Ans.** ($\because$ cm^2/s = stoke)

Problem 1.11 *Determine the specific gravity of a fluid having viscosity 0.05 poise and kinematic viscosity 0.035 stokes.*

Solution. Given :

Viscosity, $\mu = 0.05$ poise $= \dfrac{0.05}{10}$ N s/m^2

Kinematic viscosity, $\nu = 0.035$ stokes

$= 0.035\ cm^2/s$ $\{\because$ Stoke $= cm^2/s\}$

$= 0.035 \times 10^{-4}\ m^2/s$

Using the relation $\nu = \frac{\mu}{\rho}$, we get $0.035 \times 10^{-4} = \frac{0.05}{10} \times \frac{1}{\rho}$

$\therefore \quad \rho = \frac{0.05}{10} \times \frac{1}{0.035 \times 10^{-4}} = 1428.5\ kg/m^3$

$\therefore$ Sp. gr. of liquid $= \frac{\text{Density of liquid}}{\text{Density of water}} = \frac{1428.5}{1000} = 1.4285 \simeq$ **1.43. Ans.**

Problem 1.12 *Determine the viscosity of a liquid having kinematic viscosity 6 stokes and specific gravity 1.9.*

Solution. Given :

Kinematic viscosity $\nu = 6$ stokes $= 6\ cm^2/s = 6 \times 10^{-4}\ m^2/s$

Sp. gr. of liquid $= 1.9$

Let the viscosity of liquid $= \mu$

Now sp. gr. of a liquid $= \frac{\text{Density of the liquid}}{\text{Density of water}}$

or $1.9 = \frac{\text{Density of liquid}}{1000}$

$\therefore$ Density of liquid $= 1000 \times 1.9 = 1900\ \frac{kg}{m^3}$

$\therefore$ Using the relation $\nu = \frac{\mu}{\rho}$, we get

$6 \times 10^{-4} = \frac{\mu}{1900}$

or $\mu = 6 \times 10^{-4} \times 1900 = 1.14\ Ns/m^2$

$= 1.14 \times 10 =$ **11.40 poise. Ans.**

Problem 1.13 *The velocity distribution for flow over a flat plate is given by $u = \frac{3}{4}\ y - y^2$ in which u is the velocity in metre per second at a distance y metre above the plate. Determine the shear stress at $y = 0.15$ m. Take dynamic viscosity of fluid as 8.6 poise.*

Solution. Given : $u = \frac{3}{4}\ y - y^2$

$\therefore \quad \frac{du}{dy} = \frac{3}{4} - 2y$

At $y = 0.15$, $\frac{du}{dy} = \frac{3}{4} - 2 \times 0.15 = 0.75 - 0.30 = 0.45$

Viscosity, $\mu = 8.5$ poise $= \frac{8.5}{10}\ \frac{N\,s}{m^2}$ $\left(\because 10 \text{ poise} = 1\ \frac{Ns}{m^2}\right)$

Using equation (1.2), $\tau = \mu \frac{du}{dy} = \frac{8.5}{10} \times 0.45 \frac{\text{N}}{\text{m}^2} = \mathbf{0.3825 \frac{N}{m^2}}$. **Ans.**

Problem 1.14 *The dynamic viscosity of an oil, used for lubrication between a shaft and sleeve is 6 poise. The shaft is of diameter 0.4 m and rotates at 190 r.p.m. Calculate the power lost in the bearing for a sleeve length of 90 mm. The thickness of the oil film is 1.5 mm.*

Solution. Given :

Viscosity $\mu = 6$ poise $= \frac{6}{10} \frac{\text{N s}}{\text{m}^2} = 0.6 \frac{\text{N s}}{\text{m}^2}$

Dia. of shaft, $D = 0.4$ m

Speed of shaft, $N = 190$ r.p.m

Sleeve length, $L = 90 \text{ mm} = 90 \times 10^{-3}$ m

Thickness of oil film, $t = 1.5 \text{ mm} = 1.5 \times 10^{-3}$ m

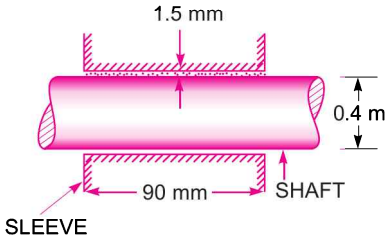


Fig. 1.5

Tangential velocity of shaft, $u = \frac{\pi DN}{60} = \frac{\pi \times 0.4 \times 190}{60} = 3.98$ m/s

Using the relation $\tau = \mu \frac{du}{dy}$

where du = Change of velocity $= u - 0 = u = 3.98$ m/s

dy = Change of distance $= t = 1.5 \times 10^{-3}$ m

$$\tau = 10 \times \frac{3.98}{1.5 \times 10^{-3}} = 1592 \text{ N/m}^2$$

This is shear stress on shaft

$\therefore$ Shear force on the shaft, F = Shear stress × Area

$= 1592 \times \pi D \times L = 1592 \times \pi \times .4 \times 90 \times 10^{-3} = 180.05$ N

Torque on the shaft, $T = \text{Force} \times \frac{D}{2} = 180.05 \times \frac{0.4}{2} = 36.01$ Nm

$\therefore$ *Power lost $= \frac{2\pi NT}{60} = \frac{2\pi \times 190 \times 36.01}{60} = \mathbf{716.48}$ **W. Ans.**

Problem 1.15 *If the velocity profile of a fluid over a plate is parabolic with the vertex 20 cm from the plate, where the velocity is 120 cm/sec. Calculate the velocity gradients and shear stresses at a distance of 0, 10 and 20 cm from the plate, if the viscosity of the fluid is 8.5 poise.*

Solution. Given :

Distance of vertex from plate = 20 cm

Velocity at vertex, $u = 120$ cm/sec

Viscosity, $\mu = 8.5 \text{ poise} = \frac{8.5}{10} \frac{\text{N s}}{\text{m}^2} = 0.85.$

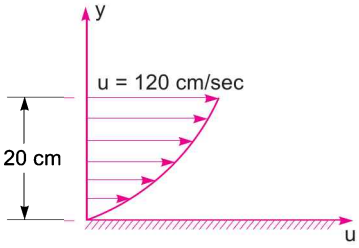


Fig. 1.6

* Power in S.I. unit = T * ω = T × $\frac{2\pi N}{60}$ Watt = $\frac{2\pi NT}{60}$ Watt

The velocity profile is given parabolic and equation of velocity profile is

$$u = ay^2 + by + c \qquad ...(i)$$

where a, b and c are constants. Their values are determined from boundary conditions as :

(a) at $y = 0$, $u = 0$

(b) at $y = 20$ cm, $u = 120$ cm/sec

(c) at $y = 20$ cm, $\frac{du}{dy} = 0$.

Substituting boundary condition (a) in equation (i), we get

$$c = 0.$$

Boundary condition (b) on substitution in (i) gives

$$120 = a(20)^2 + b(20) = 400a + 20b \qquad ...(ii)$$

Boundary condition (c) on substitution in equation (i) gives

$$\frac{du}{dy} = 2ay + b \qquad ...(iii)$$

or $$0 = 2 \times a \times 20 + b = 40a + b$$

Solving equations (ii) and (iii) for a and b

From equation (iii), $b = -40a$

Substituting this value in equation (ii), we get

$$120 = 400a + 20 \times (-40a) = 400a - 800a = -400a$$

$$\therefore \quad a = \frac{120}{-400} = -\frac{3}{10} = -0.3$$

$$\therefore \quad b = -40 \times (-0.3) = 12.0$$

Substituting the values of a, b and c in equation (i),

$$u = -0.3y^2 + 12y.$$

Velocity Gradient

$$\frac{du}{dy} = -0.3 \times 2y + 12 = -0.6y + 12$$

at $y = 0$, Velocity gradient, $\left(\frac{du}{dy}\right)_{y=0} = -0.6 \times 0 + 12 =$ **12/s. Ans.**

at $y = 10$ cm, $\left(\frac{du}{dy}\right)_{y=10} = -0.6 \times 10 + 12 = -6 + 12 =$ **6/s. Ans.**

at $y = 20$ cm, $\left(\frac{du}{dy}\right)_{y=20} = -0.6 \times 20 + 12 = -12 + 12 =$ **0. Ans.**

Shear Stresses

Shear stress is given by, $\tau = \mu \frac{du}{dy}$

(*i*) Shear stress at $y = 0$, $\tau = \mu \left(\frac{du}{dy}\right)_{y=0} = 0.85 \times 12.0 = 10.2$ N/m^2.

(*ii*) Shear stress at $y = 10$, $\tau = \mu \left(\frac{du}{dy}\right)_{y=10} = 0.85 \times 6.0 = 5.1$ N/m^2.

(*iii*) Shear stress at $y = 20$, $\tau = \mu \left(\frac{du}{dy}\right)_{y=20} = 0.85 \times 0 =$ **0. Ans.**

Problem 1.16 *A Newtonian fluid is filled in the clearance between a shaft and a concentric sleeve. The sleeve attains a speed of 50 cm/s, when a force of 40 N is applied to the sleeve parallel to the shaft. Determine the speed if a force of 200 N is applied.*

Solution. Given : Speed of sleeve, $u_1 = 50$ cm/s

when force, $F_1 = 40$ N.

Let speed of sleeve is u_2 when force, $F_2 = 200$ N.

Using relation $\tau = \mu \frac{du}{dy}$

where τ = Shear stress = $\frac{\text{Force}}{\text{Area}} = \frac{F}{A}$

du = Change of velocity = $u - 0 = u$

dy = Clearance = y

$\therefore \quad \frac{F}{A} = \mu \frac{u}{y}$

$\therefore \quad F = \frac{A \mu u}{y} \propto u$ {$\because$ A, μ and y are constant}

$\therefore \quad \frac{F_1}{u_1} = \frac{F_2}{u_2}$

Substituting values, we get $\frac{40}{50} = \frac{200}{u_2}$

$\therefore \quad u_2 = \frac{50 \times 200}{40} = 50 \times 5 =$ **250 cm/s. Ans.**

Problem 1.17 *A 15 cm diameter vertical cylinder rotates concentrically inside another cylinder of diameter 15.10 cm. Both cylinders are 25 cm high. The space between the cylinders is filled with a liquid whose viscosity is unknown. If a torque of 12.0 Nm is required to rotate the inner cylinder at 100 r.p.m., determine the viscosity of the fluid.*

Solution. Given :

Diameter of cylinder = 15 cm = 0.15 m

Dia. of outer cylinder = 15.10 cm = 0.151 m

Length of cylinders, $L = 25$ cm $= 0.25$ m

Torque, $T = 12.0$ Nm

Speed, $N = 100$ r.p.m.

Let the viscosity $= \mu$

Tangential velocity of cylinder, $u = \dfrac{\pi DN}{60} = \dfrac{\pi \times 0.15 \times 100}{60} = 0.7854$ m/s

Surface area of cylinder, $A = \pi D \times L = \pi \times 0.15 \times 0.25 = .1178$ m^2

Now using relation $\tau = \mu \dfrac{du}{dy}$

where $du = u - 0 = u = .7854$ m/s

$$dy = \frac{0.151 - 0.150}{2} \text{ m} = .0005 \text{ m}$$

$$\tau = \frac{\mu \times .7854}{.0005}$$

$\therefore$ Shear force, $F =$ Shear stress $\times$ Area $= \dfrac{\mu \times .7854}{.0005} \times .1178$

$\therefore$ Torque, $T = F \times \dfrac{D}{2}$

$$12.0 = \frac{\mu \times .7854}{.0005} \times .1178 \times \frac{.15}{2}$$

$\therefore$ $$\mu = \frac{12.0 \times .0005 \times 2}{.7854 \times .1178 \times .15} = 0.864 \text{ N s/m}^2$$

$= 0.864 \times 10 =$ **8.64 poise. Ans.**

Problem 1.18 *Two large plane surfaces are 2.4 cm apart. The space between the surfaces is filled with glycerine. What force is required to drag a very thin plate of surface area 0.5 square metre between the two large plane surfaces at a speed of 0.6 m/s, if :*

(i) the thin plate is in the middle of the two plane surfaces, and

(ii) the thin plate is at a distance of 0.8 cm from one of the plane surfaces ? Take the dynamic viscosity of glycerine = 8.10×10^{-1} N s/m^2.

Solution. Given :

Distance between two large surfaces = 2.4 cm

Area of thin plate, $A = 0.5$ m^2

Velocity of thin plate, $u = 0.6$ m/s

Viscosity of glycerine, $\mu = 8.10 \times 10^{-1}$ N s/m^2

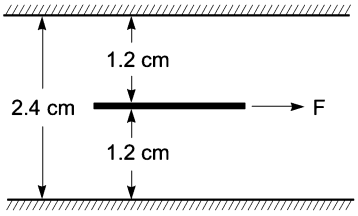


Fig. 1.7 (*a*)

Case I. When the thin plate is in the middle of the two plane surfaces [Refer to Fig. 1.7 (*a*)]

Let F_1 = Shear force on the upper side of the thin plate

F_2 = Shear force on the lower side of the thin plate

F = Total force required to drag the plate

Then $F = F_1 + F_2$

The shear stress (τ_1) on the upper side of the thin plate is given by equation,

$$\tau_1 = \mu\left(\frac{du}{dy}\right)_1$$

where du = Relative velocity between thin plate and upper large plane surface
= 0.6 m/sec
dy = Distance between thin plate and upper large plane surface
= 1.2 cm = 0.012 m (plate is a thin one and hence thickness of plate is neglected)

$$\therefore \qquad \tau_1 = 8.10 \times 10^{-1} \times \left(\frac{0.6}{.012}\right) = 40.5 \text{ N/m}^2$$

Now shear force, F_1 = Shear stress × Area
$= \tau_1 \times A = 40.5 \times 0.5 = 20.25$ N

Similarly shear stress (τ_2) on the lower side of the thin plate is given by

$$\tau_2 = \mu\left(\frac{du}{dy}\right)_2 = 8.10 \times 10^{-1} \times \left(\frac{0.6}{0.012}\right) = 40.5 \text{ N/m}^2$$

$\therefore$ Shear force, $F_2 = \tau_2 \times A = 40.5 \times 0.5 = 20.25$ N

$\therefore$ Total force, $F = F_1 + F_2 = 20.25 + 20.25 =$ **40.5 N. Ans.**

Case II. When the thin plate is at a distance of 0.8 cm from one of the plane surfaces [Refer to Fig. 1.7 (*b*)].

Let the thin plate is at a distance 0.8 cm from the lower plane surface.

Then distance of the plate from the upper plane surface
= 2.4 – 0.8 = 1.6 cm = .016 m
(Neglecting thickness of the plate)

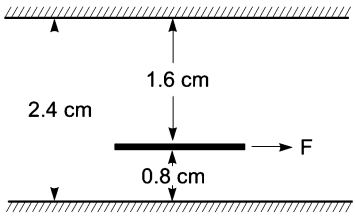


Fig. 1.7 (*b*)

The shear force on the upper side of the thin plate,

F_1 = Shear stress × Area = $\tau_1 \times A$

$$= \mu\left(\frac{du}{dy}\right)_1 \times A = 8.10 \times 10^{-1} \times \left(\frac{0.6}{0.016}\right) \times 0.5 = 15.18 \text{ N}$$

The shear force on the lower side of the thin plate,

$$F_2 = \tau_2 \times A = \mu\left(\frac{du}{dy}\right)_2 \times A$$

$$= 8.10 \times 10^{-1} \times \left(\frac{0.6}{0.8/100}\right) \times 0.5 = 30.36 \text{ N}$$

$\therefore$ Total force required = $F_1 + F_2 = 15.18 + 30.36 =$ **45.54 N. Ans.**

Problem 1.19 *A vertical gap 2.2 cm wide of infinite extent contains a fluid of viscosity 2.0 N s/m^2 and specific gravity 0.9. A metallic plate 1.2 m × 1.2 m × 0.2 cm is to be lifted up with a constant velocity of 0.15 m/sec, through the gap. If the plate is in the middle of the gap, find the force required. The weight of the plate is 40 N.*

Solution. Given :

Width of gap = 2.2 cm, viscosity, $\mu = 2.0$ N s/m^2

Sq. gr. of fluid = 0.9

$\therefore$ Weight density of fluid

$= 0.9 \times 1000 = 900 \text{ kgf/m}^3 = 900 \times 9.81 \text{ N/m}^3$

($\because$ 1 kgf = 9.81 N)

Volume of plate $= 1.2 \text{ m} \times 1.2 \text{ m} \times 0.2 \text{ cm}$

$= 1.2 \times 1.2 \times .002 \text{ m}^3 = .00288 \text{ m}^3$

Thickness of plate = 0.2 cm

Velocity of plate = 0.15 m/sec

Weight of plate = 40 N.

When plate is in the middle of the gap, the distance of the plate from vertical surface of the gap

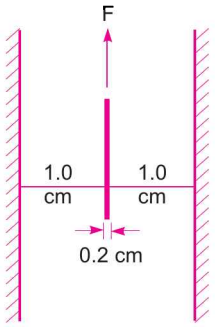


Fig. 1.8

$$= \left(\frac{\text{Width of gap} - \text{Thickness of plate}}{2}\right)$$

$$= \frac{(2.2 - 0.2)}{2} = 1 \text{ cm} = .01 \text{ m}.$$

Now the shear force on the left side of the metallic plate,

F_1 = Shear stress × Area

$$= \mu \left(\frac{du}{dy}\right)_1 \times \text{Area} = 2.0 \times \left(\frac{0.15}{.01}\right) \times 1.2 \times 1.2 \text{ N}$$

($\because$ Area $= 1.2 \times 1.2 \text{ m}^2$)

$= 43.2$ N.

Similarly, the shear force on the right side of the metallic plate,

$$F_2 = \text{Shear stress} \times \text{Area} = 2.0 \times \left(\frac{0.15}{.01}\right) \times 1.2 \times 1.2 = 43.2 \text{ N}$$

$\therefore$ Total shear force $= F_1 + F_2 = 43.2 + 43.2 = 86.4$ N.

In this case the weight of plate (which is acting vertically downward) and upward thrust is also to be taken into account.

The upward thrust = Weight of fluid displaced

= (Weight density of fluid) × Volume of fluid displaced

$= 9.81 \times 900 \times .00288$ N

($\because$ Volume of fluid displaced = Volume of plate = .00288)

$= 25.43$ N.

The net force acting in the downward direction due to weight of the plate and upward thrust

= Weight of plate – Upward thrust = 40 – 25.43 = 14.57 N

$\therefore$ Total force required to lift the plate up

= Total shear force + 14.57 = 86.4 + 14.57 = **100.97 N. Ans.**

▶ 1.4 THERMODYNAMIC PROPERTIES

Fluids consist of liquids or gases. But gases are compressible fluids and hence thermodynamic properties play an important role. With the change of pressure and temperature, the gases undergo

large variation in density. The relationship between pressure (absolute), specific volume and temperature (absolute) of a gas is given by the equation of state as

$$p \forall = RT \text{ or } \frac{p}{\rho} = RT \qquad ...(1.5)$$

where p = Absolute pressure of a gas in N/m^2

$\forall$ = Specific volume = $\frac{1}{\rho}$

R = Gas constant

T = Absolute temperature in °K

ρ = Density of a gas.

1.4.1 Dimension of R. The gas constant, R, depends upon the particular gas. The dimension of R is obtained from equation (1.5) as

$$R = \frac{p}{\rho T}$$

(*i*) In MKS units $$R = \frac{kgf/m^2}{\left(\frac{kg}{m^3}\right)°K} = \frac{kgf\text{-}m}{kg\,°K}$$

(*ii*) In SI units, p is expressed in Newton/m^2 or N/m^2.

$\therefore$ $$R = \frac{N/m^2}{\frac{kg}{m^3} \times K} = \frac{Nm}{kg\text{-}K} = \frac{Joule}{kg\text{-}K} \quad [Joule = Nm]$$

$$= \frac{J}{kg\text{-}K}$$

For air, $$R \text{ in MKS} = 29.3 \frac{kgf\text{-}m}{kg\,°K}$$

$$R \text{ in SI} = 29.3 \times 9.81 \frac{Nm}{kg°K} = 287 \frac{J}{kg\text{-}K}$$

1.4.2 Isothermal Process. If the change in density occurs at constant temperature, then the process is called isothermal and relationship between pressure (p) and density (ρ) is given by

$$\frac{p}{\rho} = \text{Constant} \qquad ...(1.6)$$

1.4.3 Adiabatic Process. If the change in density occurs with no heat exchange to and from the gas, the process is called adiabatic. And if no heat is generated within the gas due to friction, the relationship between pressure and density is given by

$$\frac{p}{\rho^k} = \text{Constant} \qquad ...(1.7)$$

where k = Ratio of specific heat of a gas at constant pressure and constant volume.

= 1.4 for air.

1.4.4 Universal Gas Constant

Let m = Mass of a gas in kg

$\forall$ = Volume of gas of mass m

p = Absolute pressure

T = Absolute temperature

Then, we have $p\forall = mRT$ ...(1.8)

where R = Gas constant.

Equation (1.8) can be made universal, *i.e.*, applicable to all gases if it is expressed in **mole-basis**.

Let n = Number of moles in volume of a gas

$\forall$ = Volume of the gas

$$M = \frac{\text{Mass of the gas molecules}}{\text{Mass of a hydrogen atom}}$$

m = Mass of a gas in kg

Then, we have $n \times M = m$.

Substituting the value of m in equation (1.8), we get

$$p\forall = n \times M \times RT \quad ...(1.9)$$

The product $M \times R$ is called universal gas constant and is equal to 848 $\frac{\text{kgf-m}}{\text{kg-mole °K}}$ in MKS units and 8314 J/kg-mole K in SI units.

One kilogram mole is defined as the product of one kilogram mass of the gas and its molecular weight.

Problem 1.20 *A gas weighs 16 N/m^3 at 25°C and at an absolute pressure of 0.25 N/mm^2. Determine the gas constant and density of the gas.*

Solution. Given :

Weight density, $w = 16$ N/m^2

Temperature, $t = 25$°C

$\therefore$ $T = 273 + t = 273 + 25 = 288$°K

$p = 0.25$ N/mm^2 (abs.) $= 0.25 \times 10^6$ N/m^2 $= 25 \times 10^4$ N/m^2

(*i*) Using relation $w = \rho g$, density is obtained as

$$\rho = \frac{w}{g} = \frac{16}{9.81} = \mathbf{1.63\ kg/m^3.\ Ans.}$$

(*ii*) Using equation (1.5), $\frac{p}{\rho} = RT$

$\therefore$ $$R = \frac{p}{\rho T} = \frac{25 \times 10^4}{1.63 \times 288} = \mathbf{532.55\ \frac{Nm}{kg\ K}.\ Ans.}$$

Problem 1.21 *A cylinder of 0.6 m^3 in volume contains air at 50°C and 0.3 N/mm^2 absolute pressure. The air is compressed to 0.3 m^3. Find (i) pressure inside the cylinder assuming isothermal process and (ii) pressure and temperature assuming adiabatic process. Take k = 1.4.*

Solution. Given :

Initial volume, $\forall_1 = 0.6$ m^3

Temperature $t_1 = 50°C$

$\therefore$ $T_1 = 273 + 50 = 323°K$

Pressure $p_1 = 0.3\ N/mm^2 = 0.3 \times 10^6\ N/m^2 = 30 \times 10^4\ N/m^2$

Final volume $\forall_2 = 0.3\ m^3$

$k = 1.4$

(*i*) Isothermal process :

Using equation (1.6), $\frac{p}{\rho}$ = Constant or $p\forall$ = Constant.

$\therefore$ $p_1\forall_1 = p_2\forall_2$

$\therefore$ $p_2 = \frac{p_1\forall_1}{\forall_2} = \frac{30 \times 10^4 \times 0.6}{0.3} = 0.6 \times 10^6\ N/m^2 = \mathbf{0.6\ N/mm^2.\ Ans.}$

(*ii*) Adiabatic process :

Using equation (1.7), $\frac{p}{\rho^k}$ = Constant or $p\ \forall^k$ = Constant

$\therefore$ $p_1\forall_1^k = p_2\forall_2^k.$

$\therefore$ $p_2 = p_1\frac{\forall_1^k}{\forall_2^k} = 30 \times 10^4 \times \left(\frac{0.6}{0.3}\right)^{1.4} = 30 \times 10^4 \times 2^{1.4}$

$= 0.791 \times 10^6\ N/m^2 = \mathbf{0.791\ N/mm^2.\ Ans.}$

For temperature, using equation (1.5), we get

$p\forall = RT$ and also $p\ \forall^k$ = Constant

$\therefore$ $p = \frac{RT}{\forall}$ and $\frac{RT}{\forall} \times \forall^k$ = Constant

or $RT\forall^{k-1}$ = Constant

or $T\ \forall^{k-1}$ = Constant $\{\because\ R$ is also constant$\}$

$\therefore$ $T_1\ \forall_1^{k-1} = T_2\ \forall_2^{k-1}$

$\therefore$ $T_2 = T_1\left(\frac{\forall_1}{\forall_2}\right)^{k-1} = 323\left(\frac{0.6}{0.3}\right)^{1.4-1.0} = 323 \times 2^{0.4} = 426.2°K$

$\therefore$ $t_2 = 426.2 - 273 = \mathbf{153.2°C.\ Ans.}$

Problem 1.22 *Calculate the pressure exerted by 5 kg of nitrogen gas at a temperature of 10°C if the volume is 0.4 m^3. Molecular weight of nitrogen is 28. Assume, ideal gas laws are applicable.*

Solution. Given :

Mass of nitrogen = 5 kg

Temperature, $t = 10°C$

$\therefore$ $T = 273 + 10 = 283°K$

Volume of nitrogen, $\forall = 0.4\ m^3$

Molecular weight = 28

Using equation (1.9), we have $p\forall = n \times M \times RT$

where $M \times R$ = Universal gas constant = $8314 \frac{\text{Nm}}{\text{kg-mole °K}}$

and one kg-mole = (kg-mass) × Molecular weight = (kg-mass) × 28

$\therefore$ R for nitrogen = $\frac{8314}{28} = 296.9 \frac{\text{Nm}}{\text{kg °K}}$

The gas laws for nitrogen is $p\forall = mRT$, where R = Characteristic gas constant

or $p \times 0.4 = 5 \times 296.9 \times 283$

$\therefore$ $p = \frac{5 \times 296.9 \times 283}{0.4} = 1050283.7 \text{ N/m}^2 = \mathbf{1.05 \text{ N/mm}^2}$**. Ans.**

▶ 1.5 COMPRESSIBILITY AND BULK MODULUS

Compressibility is the reciprocal of the bulk modulus of elasticity, K which is defined as the ratio of compressive stress to volumetric strain.

Consider a cylinder fitted with a piston as shown in Fig. 1.9.

Let $\forall$ = Volume of a gas enclosed in the cylinder

p = Pressure of gas when volume is $\forall$

Let the pressure is increased to $p + dp$, the volume of gas decreases from $\forall$ to $\forall - d\forall$.

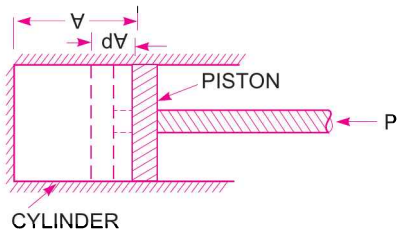


Fig. 1.9

Then increase in pressure = dp kgf/m^2

Decrease in volume = $d\forall$

$\therefore$ Volumetric strain = $-\frac{d\forall}{\forall}$

– ve sign means the volume decreases with increase of pressure.

$\therefore$ Bulk modulus $K = \frac{\text{Increase of pressure}}{\text{Volumetric strain}}$

$$= \frac{dp}{\frac{-d\forall}{\forall}} = \frac{-dp}{d\forall}\forall \qquad \text{...(1.10)}$$

Compressibility = $\frac{1}{K}$ ...(1.11)

Relationship between Bulk Modulus (K) and Pressure (p) for a Gas

The relationship between bulk modulus of elasticity (K) and pressure for a gas for two different processes of compression are as :

(i) **For Isothermal Process.** Equation (1.6) gives the relationship between pressure (p) and density (ρ) of a gas as

$$\frac{p}{\rho} = \text{Constant}$$

or $p\forall = \text{Constant}$ $\left\{\because \forall = \frac{1}{\rho}\right\}$

Differentiating this equation, we get (p and $\forall$ both are variables)

$$pd\forall + \forall dp = 0 \quad \text{or} \quad pd\forall = -\forall dp \quad \text{or} \quad p = \frac{-\forall dp}{d\forall}$$

Substituting this value in equation (1.10), we get

$$K = p \qquad ...(1.12)$$

(*ii*) **For Adiabatic Process.** Using equation (1.7) for adiabatic process

$$\frac{p}{\rho^k} = \text{Constant or } p\,\forall^k = \text{Constant}$$

Differentiating, we get $pd(\forall^k) + \forall^k(dp) = 0$

or $p \times k \times \forall^{k-1}\, d\forall + \forall^k\, dp = 0$

or $pkd\forall + \forall dp = 0$ [Cancelling $\forall^{k-1}$ to both sides]

or $$pkd\forall = -\forall dp \quad \text{or} \quad pk = -\frac{\forall dp}{d\forall}$$

Hence from equation (1.10), we have

$$K = pk \qquad ...(1.13)$$

where K = Bulk modulus and k = Ratio of specific heats.

Problem 1.23 *Determine the bulk modulus of elasticity of a liquid, if the pressure of the liquid is increased from 70 N/cm^2 to 130 N/cm^2. The volume of the liquid decreases by 0.15 per cent.*

Solution. Given :

Initial pressure $= 70$ N/cm^2

Final pressure $= 130$ N/cm^2

$\therefore$ dp = Increase in pressure $= 130 - 70 = 60$ N/cm^2

Decrease in volume $= 0.15\%$

$$\therefore \quad -\frac{d\forall}{\forall} = +\frac{0.15}{100}$$

Bulk modulus, K is given by equation (1.10) as

$$K = \frac{dp}{-\frac{d\forall}{\forall}} = \frac{60\text{ N/cm}^2}{\frac{.15}{100}} = \frac{60 \times 100}{.15} = \mathbf{4 \times 10^4\ N/cm^2.\ Ans.}$$

Problem 1.24 *What is the bulk modulus of elasticity of a liquid which is compressed in a cylinder from a volume of 0.0125 m^3 at 80 N/cm^2 pressure to a volume of 0.0124 m^3 at 150 N/cm^2 pressure ?*

Solution. Given :

Initial volume, $\forall = 0.0125$ m^3

Final volume $= 0.0124$ m^3

$\therefore$ Decrease in volume, $d\forall = .0125 - .0124 = .0001$ m^3

$$\therefore \qquad -\frac{d\forall}{\forall} = \frac{.0001}{.0125}$$

Initial pressure $= 80 \text{ N/cm}^2$

Final pressure $= 150 \text{ N/cm}^2$

$\therefore$ Increase in pressure, $dp = (150 - 80) = 70 \text{ N/cm}^2$

Bulk modulus is given by equation (1.10) as

$$K = \frac{dp}{-\frac{d\forall}{\forall}} = \frac{70}{\frac{.0001}{.0125}} = 70 \times 125 \text{ N/cm}^2$$

$$= \mathbf{8.75 \times 10^3 \text{ N/cm}^2. \text{ Ans.}}$$

▶ 1.6 SURFACE TENSION AND CAPILLARITY

Surface tension is defined as the tensile force acting on the surface of a liquid in contact with a gas or on the surface between two immiscible liquids such that the contact surface behaves like a membrane under tension. The magnitude of this force per unit length of the free surface will have the same value as the surface energy per unit area. It is denoted by Greek letter σ (called sigma). In MKS units, it is expressed as kgf/m while in SI units as N/m.

The phenomenon of surface tension is explained by Fig. 1.10. Consider three molecules *A*, *B*, *C* of a liquid in a mass of liquid. The molecule *A* is attracted in all directions equally by the surrounding molecules of the liquid. Thus the resultant force acting on the molecule *A* is zero. But the molecule *B*, which is situated near the free surface, is acted upon by upward and downward forces which are unbalanced. Thus a net resultant force on molecule *B* is acting in the downward direction. The molecule *C*, situated on the free surface of liquid, does experience a resultant downward force. All the molecules on the free surface experience a downward force. Thus the free surface of the liquid acts like a very thin film under tension of the surface of the liquid act as though it is an elastic membrane under tension.

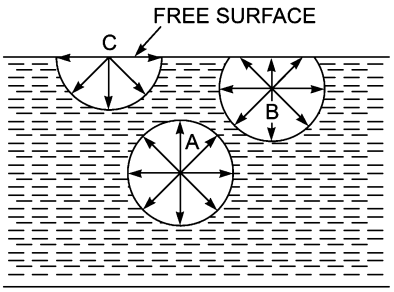


Fig. 1.10 *Surface tension.*

1.6.1 Surface Tension on Liquid Droplet. Consider a small spherical droplet of a liquid of radius '*r*'. On the entire surface of the droplet, the tensile force due to surface tension will be acting.

Let σ = Surface tension of the liquid

p = Pressure intensity inside the droplet (in excess of the outside pressure intensity)

d = Dia. of droplet.

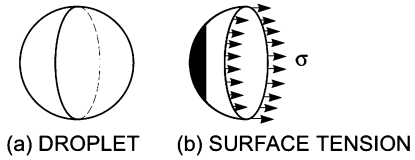
Let the droplet is cut into two halves. The forces acting on one half (say left half) will be

(*i*) tensile force due to surface tension acting around the circumference of the cut portion as shown in Fig. 1.11 (*b*) and this is equal to

$$= \sigma \times \text{Circumference}$$

$$= \sigma \times \pi d$$

(*ii*) pressure force on the area $\frac{\pi}{4}d^2 = p \times \frac{\pi}{4}d^2$ as shown in Fig. 1.11 (*c*). These two forces will be equal and opposite under equilibrium conditions, *i.e.*,



$$p \times \frac{\pi}{4} d^2 = \sigma \times \pi d$$

or
$$p = \frac{\sigma \times \pi d}{\frac{\pi}{4} \times d^2} = \frac{4\sigma}{d} \quad ...(1.14)$$

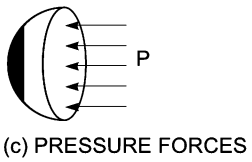


Fig. 1.11 *Forces on droplet.*

Equation (1.14) shows that with the decrease of diameter of the droplet, pressure intensity inside the droplet increases.

1.6.2 Surface Tension on a Hollow Bubble. A hollow bubble like a soap bubble in air has two surfaces in contact with air, one inside and other outside. Thus two surfaces are subjected to surface tension. In such case, we have

$$p \times \frac{\pi}{4} d^2 = 2 \times (\sigma \times \pi d)$$

$$\therefore \quad p = \frac{2\sigma\pi d}{\frac{\pi}{4}d^2} = \frac{8\sigma}{d} \quad ...(1.15)$$

1.6.3 Surface Tension on a Liquid Jet. Consider a liquid jet of diameter '*d*' and length 'L' as shown in Fig. 1.12.

Let p = Pressure intensity inside the liquid jet above the outside pressure

σ = Surface tension of the liquid.

Consider the equilibrium of the semi jet, we have

Force due to pressure = $p \times$ area of semi jet

$= p \times L \times d$

Force due to surface tension $= \sigma \times 2L$.

Equating the forces, we have

$$p \times L \times d = \sigma \times 2L$$

$$\therefore \quad p = \frac{\sigma \times 2L}{L \times d} \quad ...(1.16)$$

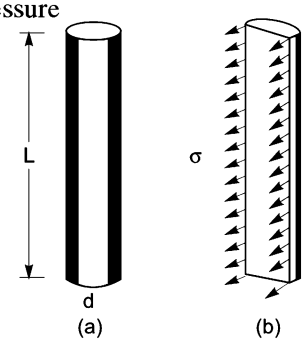


Fig. 1.12 *Forces on liquid jet.*

Problem 1.25 *The surface tension of water in contact with air at 20°C is 0.0725 N/m. The pressure inside a droplet of water is to be 0.02 N/cm^2 greater than the outside pressure. Calculate the diameter of the droplet of water.*

Solution. Given :

Surface tension, $\sigma = 0.0725$ N/m

Pressure intensity, p in excess of outside pressure is

$$p = 0.02 \text{ N/cm}^2 = 0.02 \times 10^4 \frac{\text{N}}{\text{m}^2}$$

Let d = dia. of the droplet

Using equation (1.14), we get $p = \frac{4\sigma}{d}$ or $0.02 \times 10^4 = \frac{4 \times 0.0725}{d}$

$\therefore \quad d = \frac{4 \times 0.0725}{0.02 \times (10)^4} = .00145 \text{ m} = .00145 \times 1000 = \mathbf{1.45 \text{ mm. Ans.}}$

Problem 1.26 *Find the surface tension in a soap bubble of 40 mm diameter when the inside pressure is 2.5 N/m^2 above atmospheric pressure.*

Solution. Given :

Dia. of bubble, $d = 40 \text{ mm} = 40 \times 10^{-3} \text{ m}$

Pressure in excess of outside, $p = 2.5 \text{ N/m}^2$

For a soap bubble, using equation (1.15), we get

$$p = \frac{8\sigma}{d} \quad \text{or} \quad 2.5 = \frac{8 \times \sigma}{40 \times 10^{-3}}$$

$$\sigma = \frac{2.5 \times 40 \times 10^{-3}}{8} \text{ N/m} = \mathbf{0.0125 \text{ N/m. Ans.}}$$

Problem 1.27 *The pressure outside the droplet of water of diameter 0.04 mm is 10.32 N/cm^2 (atmospheric pressure). Calculate the pressure within the droplet if surface tension is given as 0.0725 N/m of water.*

Solution. Given :

Dia. of droplet, $d = 0.04 \text{ mm} = .04 \times 10^{-3} \text{ m}$

Pressure outside the droplet $= 10.32 \text{ N/cm}^2 = 10.32 \times 10^4 \text{ N/m}^2$

Surface tension, $\sigma = 0.0725 \text{ N/m}$

The pressure inside the droplet, in excess of outside pressure is given by equation (1.14)

or $$p = \frac{4\sigma}{d} = \frac{4 \times 0.0725}{.04 \times 10^{-3}} = 7250 \text{ N/m}^2 = \frac{7250 \text{ N}}{10^4 \text{ cm}^2} = 0.725 \text{ N/cm}^2$$

$\therefore$ Pressure inside the droplet $= p$ + Pressure outside the droplet

$= 0.725 + 10.32 = \mathbf{11.045 \text{ N/cm}^2. \text{ Ans.}}$

1.6.4 Capillarity. Capillarity is defined as a phenomenon of rise or fall of a liquid surface in a small tube relative to the adjacent general level of liquid when the tube is held vertically in the liquid. The rise of liquid surface is known as capillary rise while the fall of the liquid surface is known as capillary depression. It is expressed in terms of cm or mm of liquid. Its value depends upon the specific weight of the liquid, diameter of the tube and surface tension of the liquid.

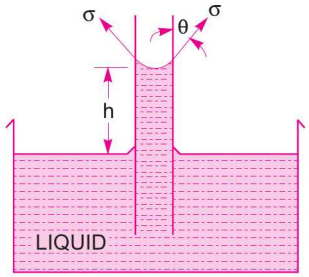


Fig. 1.13 *Capillary rise.*

Expression for Capillary Rise. Consider a glass tube of small diameter '*d*' opened at both ends and is inserted in a liquid, say water. The liquid will rise in the tube above the level of the liquid.

Let h = height of the liquid in the tube. Under a state of equilibrium, the weight of liquid of height h is balanced by the force at the surface of the liquid in the tube. But the force at the surface of the liquid in the tube is due to surface tension.

Let σ = Surface tension of liquid

θ = Angle of contact between liquid and glass tube.

The weight of liquid of height h in the tube = (Area of tube $\times$ h) $\times$ ρ $\times$ g

$$= \frac{\pi}{4} d^2 \times h \times \rho \times g \qquad ...(1.17)$$

where ρ = Density of liquid

Vertical component of the surface tensile force

$$= (\sigma \times \text{Circumference}) \times \cos\theta$$

$$= \sigma \times \pi d \times \cos\theta \qquad ...(1.18)$$

For equilibrium, equating (1.17) and (1.18), we get

$$\frac{\pi}{4} d^2 \times h \times \rho \times g = \sigma \times \pi d \times \cos\theta$$

or
$$h = \frac{\sigma \times \pi d \times \cos\theta}{\frac{\pi}{4} d^2 \times \rho \times g} = \frac{4\sigma \cos\theta}{\rho \times g \times d} \qquad ...(1.19)$$

The value of θ between water and clean glass tube is approximately equal to zero and hence cos θ is equal to unity. Then rise of water is given by

$$h = \frac{4\sigma}{\rho \times g \times d} \qquad ... (1.20)$$

Expression for Capillary Fall. If the glass tube is dipped in mercury, the level of mercury in the tube will be lower than the general level of the outside liquid as shown in Fig. 1.14.

Let h = Height of depression in tube.

Then in equilibrium, two forces are acting on the mercury inside the tube. First one is due to surface tension acting in the downward direction and is equal to $\sigma \times \pi d \times \cos\theta$.

Second force is due to hydrostatic force acting upward and is equal to intensity of pressure at a depth 'h' × Area

$$= p \times \frac{\pi}{4} d^2 = \rho g \times h \times \frac{\pi}{4} d^2 \{\because p = \rho g h\}$$

Equating the two, we get

$$\sigma \times \pi d \times \cos\theta = \rho g h \times \frac{\pi}{4} d^2$$

$$\therefore \quad h = \frac{4\sigma \cos\theta}{\rho g d} \qquad ...(1.21)$$

Value of θ for mercury and glass tube is 128°.

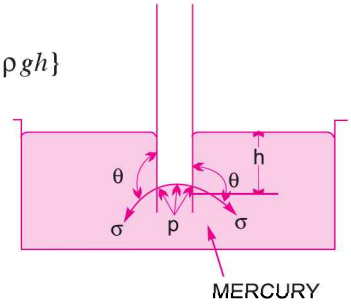


Fig. 1.14

Problem 1.28 *Calculate the capillary rise in a glass tube of 2.5 mm diameter when immersed vertically in (a) water and (b) mercury. Take surface tensions σ = 0.0725 N/m for water and σ = 0.52 N/m for mercury in contact with air. The specific gravity for mercury is given as 13.6 and angle of contact = 130°.*

Solution. Given :

Dia. of tube, $d = 2.5 \text{ mm} = 2.5 \times 10^{-3}$ m

Surface tension, σ for water = 0.0725 N/m

σ for mercury = 0.52 N/m

Sp. gr. of mercury = 13.6

$\therefore$ Density $= 13.6 \times 1000$ kg/m^3.

(*a*) **Capillary rise for water ($\theta = 0^o$)**

Using equation (1.20), we get $h = \dfrac{4\sigma}{\rho \times g \times d} = \dfrac{4 \times 0.0725}{1000 \times 9.81 \times 2.5 \times 10^{-3}}$

$= .0118$ m = **1.18 cm. Ans.**

(*b*) **For mercury**

Angle of contact between mercury and glass tube, $\theta = 130°$

Using equation (1.21), we get $h = \dfrac{4\sigma \cos\theta}{\rho \times g \times d} = \dfrac{4 \times 0.52 \times \cos 130°}{13.6 \times 1000 \times 9.81 \times 2.5 \times 10^{-3}}$

$= -.004$ m = **– 0.4 cm. Ans.**

The negative sign indicates the capillary depression.

Problem 1.29 *Calculate the capillary effect in millimetres in a glass tube of 4 mm diameter, when immersed in (i) water, and (ii) mercury. The temperature of the liquid is 20°C and the values of the surface tension of water and mercury at 20°C in contact with air are 0.073575 N/m and 0.51 N/m respectively. The angle of contact for water is zero and that for mercury is 130°. Take density of water at 20°C as equal to 998 kg/m^3.*

Solution. Given :

Dia. of tube, $d = 4$ mm $= 4 \times 10^{-3}$ m

The capillary effect (*i.e.*, capillary rise or depression) is given by equation (1.20) as

$$h = \frac{4\sigma \cos\theta}{\rho \times g \times d}$$

where σ = surface tension in N/m

θ = angle of contact, and ρ = density

(*i*) **Capillary effect for water**

$\sigma = 0.073575$ N/m, $\theta = 0°$

$\rho = 998$ kg/m^3 at 20°C

$\therefore$ $h = \dfrac{4 \times 0.073575 \times \cos 0°}{998 \times 9.81 \times 4 \times 10^{-3}} = 7.51 \times 10^{-3}$ m = **7.51 mm. Ans.**

(*ii*) **Capillary effect for mercury**

$\sigma = 0.51$ N/m, $\theta = 130°$ and

ρ = sp. gr. $\times$ 1000 = 13.6 $\times$ 1000 = 13600 kg/m^2

$\therefore$ $h = \dfrac{4 \times 0.51 \times \cos 130°}{13600 \times 9.81 \times 4 \times 10^{-3}} = -2.46 \times 10^{-3}$ m = **– 2.46 mm. Ans.**

The negative sign indicates the capillary depression.

Problem 1.30 *The capillary rise in the glass tube is not to exceed 0.2 mm of water. Determine its minimum size, given that surface tension for water in contact with air = 0.0725 N/m.*

Solution. Given :

Capillary rise, $h = 0.2$ mm $= 0.2 \times 10^3$ m

Surface tension, $\sigma = 0.0725$ N/m

Let dia. of tube $= d$

The angle θ for water $= 0^o$

Density (ρ) for water $= 1000 \text{ kg/m}^3$

Using equation (1.20), we get

$$h = \frac{4\sigma}{\rho \times g \times d} \text{ or } 0.2 \times 10^{-3} = \frac{4 \times 0.0725}{1000 \times 9.81 \times d}$$

$$\therefore \quad d = \frac{4 \times 0.0725}{1000 \times 9.81 \times .2 \times 10^{-3}} = 0.148 \text{ m} = \textbf{14.8 cm. Ans.}$$

Thus minimum diameter of the tube should be 14.8 cm.

Problem 1.31 *Find out the minimum size of glass tube that can be used to measure water level if the capillary rise in the tube is to be restricted to 2 mm. Consider surface tension of water in contact with air as 0.073575 N/m.*

Solution. Given :

Capillary rise, $h = 2.0 \text{ mm} = 2.0 \times 10^{-3} \text{ m}$

Surface tension, $\sigma = 0.073575 \text{ N/m}$

Let dia. of tube $= d$

The angle θ for water $= 0^o$

The density for water, $\rho = 1000 \text{ kg/m}^3$

Using equation (1.20), we get

$$h = \frac{4\sigma}{\rho \times g \times d} \text{ or } 2.0 \times 10^{-3} = \frac{4 \times 0.073575}{1000 \times 9.81 \times d}$$

$$\therefore \quad d = \frac{4 \times 0.073575}{1000 \times 9.81 \times 2 \times 10^{-3}} = 0.015 \text{ m} = \textbf{1.5 cm. Ans.}$$

Thus minimum diameter of the tube should be 1.5 cm.

Problem 1.32 *An oil of viscosity 5 poise is used for lubrication between a shaft and sleeve. The diameter of the shaft is 0.5 m and it rotates at 200 r.p.m. Calculate the power lost in oil for a sleeve length of 100 mm. The thickness of oil film is 1.0 mm.*

Solution. Given :

Viscosity, $\mu = 5 \text{ poise}$

$$= \frac{5}{10} = 0.5 \text{ N s/m}^2$$

Dia. of shaft, $D = 0.5 \text{ m}$

Speed of shaft, $N = 200 \text{ r.p.m.}$

Sleeve length, $L = 100 \text{ mm} = 100 \times 10^{-3} \text{ m} = 0.1 \text{ m}$

Thickness of oil film, $t = 1.0 \text{ mm} = 1 \times 10^{-3} \text{ m}$

Tangential velocity of shaft, $u = \frac{\pi DN}{60} = \frac{\pi \times 0.5 \times 200}{60} = 5.235 \text{ m/s}$

Using the relation, $\tau = \mu \frac{du}{dy}$

where, du = Change of velocity = $u - 0 = u = 5.235$ m/s

dy = Change of distance = $t = 1 \times 10^{-3}$ m

$$\therefore \qquad \tau = \frac{0.5 \times 5.235}{1 \times 10^{-3}} = 2617.5 \text{ N/m}^2$$

This is the shear stress on the shaft

$\therefore$ Shear force on the shaft, F = Shear stress × Area = $2617.5 \times \pi D \times L$ ($\because$ Area = $\pi D \times L$)

$= 2617.5 \times \pi \times 0.5 \times 0.1 = 410.95$ N

Torque on the shaft, T = Force × $\frac{D}{2} = 410.95 \times \frac{0.5}{2} = 102.74$ Nm

$$\therefore \qquad \text{Power* lost} = T \times \omega \text{ Watts} = T \times \frac{2\pi N}{60} \text{ W}$$

$$= 102.74 \times \frac{2\pi \times 200}{60} = 2150 \text{ W} = \mathbf{2.15 \text{ kW. Ans.}}$$

▶ 1.7 VAPOUR PRESSURE AND CAVITATION

A change from the liquid state to the gaseous state is known as vaporization. The vaporization (which depends upon the prevailing pressure and temperature condition) occurs because of continuous escaping of the molecules through the free liquid surface.

Consider a liquid (say water) which is confined in a closed vessel. Let the temperature of liquid is 20°C and pressure is atmospheric. This liquid will vaporise at 100°C. When vaporization takes place, the molecules escapes from the free surface of the liquid. These vapour molecules get accumulated in the space between the free liquid surface and top of the vessel. These accumulated vapours exert a pressure on the liquid surface. This pressure is known as **vapour pressure** of the liquid or this is the pressure at which the liquid is converted into vapours.

Again consider the same liquid at 20°C at atmospheric pressure in the closed vessel. If the pressure above the liquid surface is reduced by some means, the boiling temperature will also reduce. If the pressure is reduced to such an extent that it becomes equal to or less than the vapour pressure, the boiling of the liquid will start, though the temperature of the liquid is 20°C. Thus a liquid may boil even at ordinary temperature, if the pressure above the liquid surface is reduced so as to be equal or less than the vapour pressure of the liquid at that temperature.

Now consider a flowing liquid in a system. If the pressure at any point in this flowing liquid becomes equal to or less than the vapour pressure, the vaporization of the liquid starts. The bubbles of these vapours are carried by the flowing liquid into the region of high pressure where they collapse, giving rise to high impact pressure. The pressure developed by the collapsing bubbles is so high that the material from the adjoining boundaries gets eroded and cavities are formed on them. This phenomenon is known as **cavitation**.

Hence the cavitation is the phenomenon of formation of vapour bubbles of a flowing liquid in a region where the pressure of the liquid falls below the vapour pressure and sudden collapsing of these vapour bubbles in a region of higher pressure. When the vapour bubbles collapse, a very high pressure is created. The metallic surfaces, above which the liquid is flowing, is subjected to these high pressures, which cause pitting action on the surface. Thus cavities are formed on the metallic surface and hence the name is cavitation.

---

* Power in case of S.I. Unit = $T \times \omega$ or $\frac{2\pi NT}{60}$ Watts or $\frac{2\pi NT}{60{,}000}$ kW. The angular velocity $\omega = \frac{2\pi N}{60}$.

HIGHLIGHTS

1. The weight density or specific weight of a fluid is equal to weight per unit volume. It is also equal to, $w = \rho \times g$.
2. Specific volume is the reciprocal of mass density.
3. The shear stress is proportional to the velocity gradient $\frac{du}{dy}$. Mathematically, $\tau = \mu \frac{du}{dy}$.
4. Kinematic viscosity ν is given by $\nu = \frac{\mu}{\rho}$.
5. Poise and stokes are the units of viscosity and kinematic viscosity respectively.
6. To convert the unit of viscosity from poise to MKS units, poise should be divided by 98.1 and to convert poise into SI units, the poise should be divided by 10. SI unit of viscosity is Ns/m^2 or Pa s, where N/m^2 = Pa = Pascal.
7. For a perfect gas, the equation of state is $\frac{p}{\rho} = RT$

   where R = gas constant and for air = $29.3 \frac{\text{kgf-m}}{\text{kg}^\circ\text{K}}$ = 287 J/kg °K.
8. For isothermal process, $\frac{p}{\rho}$ = Constant whereas for adiabatic process, $\frac{p}{\rho^k}$ = constant.
9. Bulk modulus of elasticity is given as $K = \frac{-dp}{\left(\frac{d\forall}{\forall}\right)}$.
10. Compressibility is the reciprocal of bulk modulus of elasticity or = $\frac{1}{K}$.
11. Surface tension is expressed in N/m or dyne/cm. The relation between surface tension (σ) and difference of pressure (p) between the inside and outside of a liquid drop is given as $p = \frac{4\sigma}{d}$

    For a soap bubble, $p = \frac{8\sigma}{d}$.

    For a liquid jet, $p = \frac{2\sigma}{d}$.
12. Capillary rise or fall of a liquid is given by $h = \frac{4\sigma \cos\theta}{wd}$.

    The value of θ for water is taken equal to zero and for mercury equal to 128°.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define the following fluid properties :
   Density, weight density, specific volume and specific gravity of a fluid.
2. Differentiate between : (*i*) Liquids and gases, (*ii*) Real fluids and ideal fluids, (*iii*) Specific weight and specific volume of a fluid.
3. What is the difference between dynamic viscosity and kinematic viscosity ? State their units of measurements.

4. Explain the terms : (*i*) Dynamic viscosity, and (*ii*) Kinematic viscosity. Give their dimensions.
5. State the Newton's law of viscosity and give examples of its application.
6. Enunciate Newton's law of viscosity. Explain the importance of viscosity in fluid motion. What is the effect of temperature on viscosity of water and that of air?
7. Define Newtonian and Non-Newtonian fluids.
8. What do you understand by terms : (*i*) Isothermal process, (*ii*) Adiabatic process, and (*iii*) Universal-gas constant.
9. Define compressibility. Prove that compressibility for a perfect gas undergoing isothermal compression is $\frac{1}{p}$ while for a perfect gas undergoing isentropic compression is $\frac{1}{wp}$.
10. Define surface tension. Prove that the relationship between surface tension and pressure inside a droplet of liquid in excess of outside pressure is given by $p = \frac{4\sigma}{d}$.
11. Explain the phenomenon of capillarity. Obtain an expression for capillary rise of a liquid.
12. (*a*) Distinguish between ideal fluids and real fluids. Explain the importance of compressibility in fluid flow.
    (*b*) Define the terms : density, specific volume, specific gravity, vacuum pressure, compressible and incompressible fluids. (*R.G.P. Vishwavidyalaya, Bhopal S 2002*)
13. Define and explain Newton's law of viscosity.
14. Convert 1 kg/s-m dynamic viscosity in poise.
15. Why does the viscosity of a gas increases with the increase in temperature while that of a liquid decreases with increase in temperature ?
16. (*a*) How does viscosity of a fluid vary with temperature ?
    (*b*) Cite examples where surface tension effects play a prominent role. (*J.N.T.U., Hyderabad S 2002*)
17. (*i*) Develop the expression for the relation between gauge pressure P inside a droplet of liquid and the surface tension.
    (*ii*) Explain the following :
    Newtonian and Non-Newtonian fluids, vapour pressure, and compressibility.
    (*R.G.P.V., Bhopal S 2001*)

(B) NUMERICAL PROBLEMS

1. One litre of crude oil weighs 9.6 N. Calculate its specific weight, density and specific gravity.
   [**Ans.** 9600 N/m^3, 978.6 kg/m^3, 0.978]
2. The velocity distribution for flow over a flat plate is given by $u = \frac{3}{2}\ y - y^{3/2}$, where u is the point velocity in metre per second at a distance y metre above the plate. Determine the shear stress at $y = 9$ cm. Assume dynamic viscosity as 8 poise. (*Nagpur University*) [**Ans.** 0.839 N/m^2]
3. A plate 0.025 mm distant from a fixed plate, moves at 50 cm/s and requires a force of 1.471 N/m^2 to maintain this speed. Determine the fluid viscosity between the plates in the poise. [**Ans.** 7.357×10^{-4}]
4. Determine the intensity of shear of an oil having viscosity = 1.2 poise and is used for lubrication in the clearance between a 10 cm diameter shaft and its journal bearing. The clearance is 1.0 mm and shaft rotates at 200 r.p.m. [**Ans.** 125.56 N/m^2]
5. Two plates are placed at a distance of 0.15 mm apart. The lower plate is fixed while the upper plate having surface area 1.0 m^2 is pulled at 0.3 m/s. Find the force and power required to maintain this speed, if the fluid separating them is having viscosity 1.5 poise. [**Ans.** 300 N, 89.8 W]
6. An oil film of thickness 1.5 mm is used for lubrication between a square plate of size 0.9 m × 0.9 m and an inclined plane having an angle of inclination 20°. The weight of the square is 392.4 N and it slides down the plane with a uniform velocity of 0.2 m/s. Find the dynamic viscosity of the oil. [**Ans.** 12.42 poise]

7. In a stream of glycerine in motion, at a certain point the velocity gradient is 0.25 metre per sec per metre. The mass density of fluid is 1268.4 kg per cubic metre and kinematic viscosity is 6.30×10^{-4} square metre per second. Calculate the shear stress at the point. [**Ans.** 0.2 N/m^2]

8. Find the kinematic viscosity of an oil having density 980 kg/m^2 when at a certain point in the oil, the shear stress is 0.25 N/m^2 and velocity gradient is 0.3/s. $\left[\textbf{Ans.}\ 0.000849\ \frac{\text{m}^2}{\text{sec}}\ \text{or}\ 8.49\ \text{stokes}\right]$

9. Determine the specific gravity of a fluid having viscosity 0.07 poise and kinematic viscosity 0.042 stokes. [**Ans.** 1.667]

10. Determine the viscosity of a liquid having kinematic viscosity 6 stokes and specific gravity 2.0. [**Ans.** 11.99 poise]

11. If the velocity distribution of a fluid over a plate is given by $u = (3/4)\, y - y^2$, where u is the velocity in metre per second at a distance of y metres above the plate, determine the shear stress at $y = 0.15$ metre. Take dynamic viscosity of the fluid as 8.5×10^{-5} kg-sec/m^2. [**Ans.** 3.825×10^{-5} kgf/m^2]

12. An oil of viscosity 5 poise is used for lubrication between a shaft and sleeve. The diameter of shaft is 0.5 m and it rotates at 200 r.p.m. Calculate the power lost in the oil for a sleeve length of 100 mm. The thickness of the oil film is 1.0 mm. [**Ans.** 2.15 kW]

13. The velocity distribution over a plate is given by $u = \frac{2}{3}\ y - y^2$ in which u is the velocity in m/sec at a distance of y m above the plate. Determine the shear stress at $y = 0$, 0.1 and 0.2 m. Take $\mu = 6$ poise. [**Ans.** 0.4, 0.028 and 0.159 N/m^2]

14. In question 13, find the distance in metres above the plate, at which the shear stress is zero. [**Ans.** 0.333 m]

15. The velocity profile of a viscous fluid over a plate is parabolic with vertex 20 cm from the plate, where the velocity is 120 cm/s. Calculate the velocity gradient and shear stress at distances of 0, 5 and 15 cm from the plate, given the viscosity of the fluid = 6 poise. [**Ans.** 12/s, 7.18 N/m^2; 9/s, 5.385 N/m^2 ; 3/s, 1.795 N/m^2]

16. The weight of a gas is given as 17.658 N/m^3 at 30°C and at an absolute pressure of 29.43 N/cm^2. Determine the gas constant and also the density of the gas. $\left[\textbf{Ans.}\ \frac{1.8\ \text{kg}}{\text{m}^3}, \frac{539.55\ \text{N m}}{\text{kg°K}}\right]$

17. A cylinder of 0.9 m^3 in volume contains air at 0°C and 39.24 N/cm^2 absolute pressure. The air is compressed to 0.45 m^3. Find (*i*) the pressure inside the cylinder assuming isothermal process, (*ii*) pressure and temperature assuming adiabatic process. Take $k = 1.4$ for air. [**Ans.** (*i*) 78.48 N/cm^2, (*ii*) 103.5 N/m^2, 140°C]

18. Calculate the pressure exerted by 4 kg mass of nitrogen gas at a temperature of 15°C if the volume is 0.35 m^3. Molecular weight of nitrogen is 28. [**Ans.** 97.8 N/cm^2]

19. The pressure of a liquid is increased from 60 N/cm^2 to 100 N/cm^2 and volume decreases by 0.2 per cent. Determine the bulk modulus of elasticity. [**Ans.** 2×10^4 N/cm^2]

20. Determine the bulk modulus of elasticity of a fluid which is compressed in a cylinder from a volume of 0.009 m^3 at 70 N/cm^2 pressure to a volume of 0.0085 m^3 at 270 N/cm^2 pressure. [**Ans.** 3.6×10^3 N/cm^2]

21. The surface tension of water in contact with air at 20°C is given as 0.0716 N/m. The pressure inside a droplet of water is to be 0.0147 N/cm^2 greater than the outside pressure, calculate the diameter of the droplet of water. [**Ans.** 1.94 mm]

22. Find the surface tension in a soap bubble of 30 mm diameter when the inside pressure is 1.962 N/m^2 above atmosphere. [**Ans.** 0.00735 N/m]

23. The surface tension of water in contact with air is given as 0.0725 N/m. The pressure outside the droplet of water of diameter 0.02 mm is atmospheric $\left(10.32\ \frac{\text{N}}{\text{cm}^2}\right)$. Calculate the pressure within the droplet of water. [**Ans.** 11.77 N/cm^2]

24. Calculate the capillary rise in a glass tube of 3.0 mm diameter when immersed vertically in (*a*) water, and (*b*) mercury. Take surface tensions for mercury and water as 0.0725 N/m and 0.52 N/m respectively in contact with air. Specific gravity for mercury is given as 13.6. [**Ans.** 0.966 cm, 0.3275 cm]

25. The capillary rise in the glass tube used for measuring water level is not to exceed 0.5 mm. Determine its minimum size, given that surface tension for water in contact with air = 0.07112 N/m. [**Ans.** 5.8 cm]

26. (**SI Units**). One litre of crude oil weighs 9.6 N. Calculate its specific weight, density and specific gravity. [**Ans.** 9600 N/m^3; 979.6 kg/m^3 ; 0.9786]

27. (**SI Units**). A piston 796 mm diameter and 200 mm long works in a cylinder of 800 mm diameter. If the annular space is filled with a lubricating oil of viscosity 5 cp (centi-poise), calculate the speed of descent of the piston in vertical position. The weight of the piston and axial load are 9.81 N. [**Ans.** 7.84 m/s]

28. (**SI Units**). Find the capillary rise of water in a tube 0.03 cm diameter. The surface tension of water is 0.0735 N/m. [**Ans.** 9.99 cm]

29. Calculate the specific weight, density and specific gravity of two litres of a liquid which weight 15 N. [**Ans.** 7500 N/m^3, 764.5 kg/m^3, 0.764]

30. A 150 mm diameter vertical cylinder rotates concentrically inside another cylinder of diameter 151 mm. Both the cylinders are of 250 mm height. The space between the cylinders is filled with a liquid of viscosity 10 poise. Determine the torque required to rotate the inner cylinder at 100 r.p.m. [**Ans.** 13.87 Nm]

31. A shaft of diameter 120 mm is rotating inside a journal bearing of diameter 122 mm at a speed of 360 r.p.m. The space between the shaft and the bearing is filled with a lubricating oil of viscosity 6 poise. Find the power absorbed in oil if the length of bearing is 100 mm. [**Ans.** 115.73 W]

32. A shaft of diameter 100 mm is rotating inside a journal bearing of diameter 102 mm at a space of 360 r.p.m. The space between the shaft and bearing is filled with a lubricating oil of viscosity 5 poise. The length of the bearing is 200 mm. Find the power absorbed in the lubricating oil. [**Ans.** 111.58 W]

33. Assuming that the bulk modulus of elasticity of water is 2.07×10^6 kN/m^2 at standard atmospheric conditions, determine the increase of pressure necessary to produce 1% reduction in volume at the same temperature.

[**Hint.** $K = 2.07 \times 10^6$ kN/m^2 ; $\dfrac{-d\forall}{\forall} = \dfrac{1}{100} = 0.01$.

Increase in pressure $(dp) = K \times \left(\dfrac{-d\forall}{\forall}\right) = 2.07 \times 10^6 \times 0.01 = 2.07 \times 10^4$ kN/m^2.]

34. A square plate of size 1 m × 1 m and weighing 350 N slides down an inclined plane with a uniform velocity of 1.5 m/s. The inclined plane is laid on a slope of 5 vertical to 12 horizontal and has an oil film of 1 mm thickness. Calculate the dynamic viscosity of oil. [*J.N.T.U., Hyderabad, S 2002*]

[**Hint.** $A = 1 \times 1 = 1$ m^2, $W = 350$ N, $u = 1.5$ m/s, $\tan\theta = \dfrac{5}{12} = \dfrac{BC}{AB}$

Component of weight along the plane = $W \times \sin\theta$

where $\sin\theta = \dfrac{BC}{AC} = \dfrac{5}{13}$ $\left(\because\ AC = \sqrt{AB^2 + BC^2} = \sqrt{12^2 + 5^2} = 13\right)$

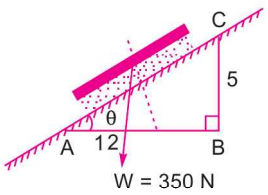


Fig. 1.15

$\therefore$ $F = W \sin\theta = 350 \times \dfrac{5}{13} = 134.615$

Now $\tau = \mu \dfrac{du}{dy}$, where $du = u - 0 = u = 1.5$ m/s and $dy = 1$ mm $= 1 \times 10^{-3}$ m

or $\dfrac{F}{A} = \mu \dfrac{du}{dy}$, $\therefore \mu = \dfrac{F}{A} \times \dfrac{dy}{du} = \dfrac{134.615}{1} \times \dfrac{1 \times 10^{-3}}{1.5} = 0.0897 \dfrac{\text{Ns}}{\text{m}^2} = 0.897$ poise]

2 CHAPTER

PRESSURE AND ITS MEASUREMENT

▶ 2.1 FLUID PRESSURE AT A POINT

Consider a small area dA in large mass of fluid. If the fluid is stationary, then the force exerted by the surrounding fluid on the area dA will always be perpendicular to the surface dA. Let dF is the force acting on the area dA in the normal direction. Then the ratio of $\frac{dF}{dA}$ is known as the intensity of pressure or simply pressure and this ratio is represented by p. Hence mathematically the pressure at a point in a fluid at rest is

$$p = \frac{dF}{dA}.$$

If the force (F) is uniformly distributed over the area (A), then pressure at any point is given by

$$p = \frac{F}{A} = \frac{\text{Force}}{\text{Area}}.$$

$\therefore$ Force or pressure force, $F = p \times A$.

The units of pressure are : (i) kgf/m^2 and kgf/cm ^{2} in MKS units, (ii) Newton/m ^{2} or N/m ^{2} and N/mm^2 in SI units. N/m^2 is known as Pascal and is represented by Pa. Other commonly used units of pressure are :

$$\text{kPa} = \text{kilo pascal} = 1000 \text{ N/m}^2$$
$$\text{bar} = 100 \text{ kPa} = 10^5 \text{ N/m}^2.$$

▶ 2.2 PASCAL'S LAW

It states that the pressure or intensity of pressure at a point in a static fluid is equal in all directions. This is proved as :

The fluid element is of very small dimensions *i.e.*, dx, dy and ds.

Consider an arbitrary fluid element of wedge shape in a fluid mass at rest as shown in Fig. 2.1. Let the width of the element perpendicular to the plane of paper is unity and p_x,

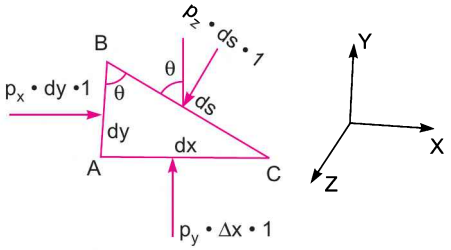


Fig. 2.1 *Forces on a fluid element.*

p_y and p_z are the pressures or intensity of pressure acting on the face AB, AC and BC respectively. Let $\angle ABC = \theta$. Then the forces acting on the element are :

1. Pressure forces normal to the surfaces, and
2. Weight of element in the vertical direction.

The forces on the faces are :

Force on the face AB $= p_x \times$ Area of face AB

$= p_x \times dy \times 1$

Similarly force on the face $AC = p_y \times dx \times 1$

Force on the face BC $= p_z \times ds \times 1$

Weight of element $=$ (Mass of element) $\times g$

$= (\text{Volume} \times \rho) \times g = \left(\frac{AB \times AC}{2} \times 1\right) \times \rho \times g,$

where ρ = density of fluid.

Resolving the forces in x-direction, we have

$$p_x \times dy \times 1 - p\,(ds \times 1) \sin(90° - \theta) = 0$$

or $$p_x \times dy \times 1 - p_z\, ds \times 1 \cos\theta = 0.$$

But from Fig. 2.1, $ds \cos\theta = AB = dy$

$\therefore$ $$p_x \times dy \times 1 - p_z \times dy \times 1 = 0$$

or $$p_x = p_z \qquad ...(2.1)$$

Similarly, resolving the forces in y-direction, we get

$$p_y \times dx \times 1 - p_z \times ds \times 1 \cos(90° - \theta) - \frac{dx \times dy}{2} \times 1 \times \rho \times g = 0$$

or $$p_y \times dx - p_z\, ds \sin\theta - \frac{dxdy}{2} \times \rho \times g = 0.$$

But $ds \sin\theta = dx$ and also the element is very small and hence weight is negligible.

$\therefore$ $$p_y dx - p_z \times dx = 0$$

or $$p_y = p_z \qquad ...(2.2)$$

From equations (2.1) and (2.2), we have

$$p_x = p_y = p_z \qquad ...(2.3)$$

The above equation shows that the pressure at any point in x, y and z directions is equal.

Since the choice of fluid element was completely arbitrary, which means the pressure at any point is the same in all directions.

▶ 2.3 PRESSURE VARIATION IN A FLUID AT REST

The pressure at any point in a fluid at rest is obtained by the Hydrostatic Law which states that the rate of increase of pressure in a vertically downward direction must be equal to the specific weight of the fluid at that point. This is proved as :

Consider a small fluid element as shown in Fig. 2.2

Let ΔA= Cross-sectional area of element

ΔZ = Height of fluid element

p = Pressure on face AB

Z = Distance of fluid element from free surface.

The forces acting on the fluid element are :

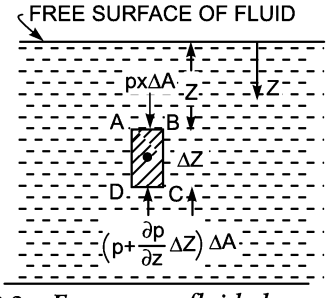


Fig. 2.2 *Forces on a fluid element.*

1. Pressure force on $AB = p \times \Delta A$ and acting perpendicular to face AB in the downward direction.

2. Pressure force on $CD = \left(p + \frac{\partial p}{\partial Z}\Delta Z\right) \times \Delta A$, acting perpendicular to face CD, vertically upward direction.

3. Weight of fluid element = Density $\times g \times$ Volume $= \rho \times g \times (\Delta A \times \Delta Z)$.

4. Pressure forces on surfaces BC and AD are equal and opposite. For equilibrium of fluid element, we have

$$p\Delta A - \left(p + \frac{\partial p}{\partial Z}\Delta Z\right)\Delta A + \rho \times g \times (\Delta A \times \Delta Z) = 0$$

or $$p\Delta A - p\Delta A - \frac{\partial p}{\partial Z}\Delta Z\Delta A + \rho \times g \times \Delta A \times Z = 0$$

or $$-\frac{\partial p}{\partial Z}\Delta Z\Delta A + \rho \times g \times \Delta A\Delta Z = 0$$

or $$\frac{\partial p}{\partial Z}\Delta Z\Delta A = \rho \times g \times \Delta A\Delta Z \quad \text{or} \quad \frac{\partial p}{\partial Z} = \rho \times g \text{ [cancelling } \Delta A\Delta Z \text{ on both sides]}$$

$$\therefore \quad \frac{\partial p}{\partial Z} = \rho \times g = w \qquad (\because \ \rho \times g = w) \qquad ...(2.4)$$

where w = Weight density of fluid.

Equation (2.4) states that rate of increase of pressure in a vertical direction is equal to weight density of the fluid at that point. This is **Hydrostatic Law.**

By integrating the above equation (2.4) for liquids, we get

$$\int dp = \int \rho g dZ$$

or $$p = \rho g Z \qquad ...(2.5)$$

where p is the pressure above atmospheric pressure and Z is the height of the point from free surfaces.

From equation (2.5), we have $$Z = \frac{p}{\rho \times g} \qquad ...(2.6)$$

Here Z is called **pressure head.**

Problem 2.1 *A hydraulic press has a ram of 30 cm diameter and a plunger of 4.5 cm diameter. Find the weight lifted by the hydraulic press when the force applied at the plunger is 500 N.*

Solution. Given :

Dia. of ram, $D = 30 \text{ cm} = 0.3 \text{ m}$

Dia. of plunger, $d = 4.5 \text{ cm} = 0.045 \text{ m}$

Force on plunger, $F = 500 \text{ N}$

Find weight lifted $= W$

Area of ram, $$A = \frac{\pi}{4} D^2 = \frac{\pi}{4} (0.3)^2 = 0.07068 \text{ m}^2$$

Area of plunger, $$a = \frac{\pi}{4} d^2 = \frac{\pi}{4} (0.045)^2 = .00159 \text{ m}^2$$

Pressure intensity due to plunger

$$= \frac{\text{Force on plunger}}{\text{Area of plunger}} = \frac{F}{a} = \frac{500}{.00159} \text{ N/m}^2.$$

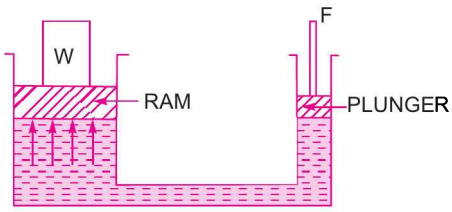


Fig. 2.3

Due to Pascal's law, the intensity of pressure will be equally transmitted in all directions. Hence the pressure intensity at the ram

$$= \frac{500}{.00159} = 314465.4 \text{ N/m}^2$$

But pressure intensity at ram $= \dfrac{\text{Weight}}{\text{Area of ram}} = \dfrac{W}{A} = \dfrac{W}{.07068} \text{ N/m}^2$

$$\frac{W}{.07068} = 314465.4$$

$\therefore$ Weight $= 314465.4 \times .07068 = 22222 \text{ N} = \mathbf{22.222 \text{ kN. Ans.}}$

Problem 2.2 *A hydraulic press has a ram of 20 cm diameter and a plunger of 3 cm diameter. It is used for lifting a weight of 30 kN. Find the force required at the plunger.*

Solution. Given :

Dia. of ram, $D = 20 \text{ cm} = 0.2 \text{ m}$

$\therefore$ Area of ram, $A = \dfrac{\pi}{4} D^2 = \dfrac{\pi}{4}(.2)^2 = 0.0314 \text{ m}^2$

Dia. of plunger $d = 3 \text{ cm} = 0.03 \text{ m}$

$\therefore$ Area of plunger, $a = \dfrac{\pi}{4}(.03)^2 = 7.068 \times 10^{-4} \text{ m}^2$

Weight lifted, $W = 30 \text{ kN} = 30 \times 1000 \text{ N} = 30000 \text{ N}.$

See Fig. 2.3.

Pressure intensity developed due to plunger $= \dfrac{\text{Force}}{\text{Area}} = \dfrac{F}{a}.$

By Pascal's Law, this pressure is transmitted equally in all directions

Hence pressure transmitted at the ram $= \dfrac{F}{a}$

$\therefore$ Force acting on ram = Pressure intensity $\times$ Area of ram

$$= \frac{F}{a} \times A = \frac{F \times .0314}{7.068 \times 10^{-4}} \text{ N}$$

But force acting on ram = Weight lifted = 30000 N

$\therefore$ $$30000 = \frac{F \times .0314}{7.068 \times 10^{-4}}$$

$\therefore$ $$F = \frac{30000 \times 7.068 \times 10^{-4}}{.0314} = \mathbf{675.2 \text{ N. Ans.}}$$

Problem 2.3 *Calculate the pressure due to a column of 0.3 of (a) water, (b) an oil of sp. gr. 0.8, and (c) mercury of sp. gr. 13.6. Take density of water, $\rho = 1000 \text{ kg/m}^3$.*

Solution. Given :

Height of liquid column, $Z = 0.3 \text{ m}.$

The pressure at any point in a liquid is given by equation (2.5) as

$$p = \rho g Z$$

(*a*) For water, $\rho = 1000 \text{ kg/m}^3$

$\therefore$ $p = \rho g Z = 1000 \times 9.81 \times 0.3 = 2943 \text{ N/m}^2$

$$= \frac{2943}{10^4} \text{ N/cm}^2 = \mathbf{0.2943 \text{ N/cm}^2}. \textbf{ Ans.}$$

(*b*) For oil of sp. gr. 0.8,

From equation (1.1*A*), we know that the density of a fluid is equal to specific gravity of fluid multiplied by density of water.

$\therefore$ Density of oil, $\rho_0 = \text{Sp. gr. of oil} \times \text{Density of water}$ (ρ_0 = Density of oil)

$= 0.8 \times \rho = 0.8 \times 1000 = 800 \text{ kg/m}^3$

Now pressure, $p = \rho_0 \times g \times Z$

$$= 800 \times 9.81 \times 0.3 = 2354.4 \frac{\text{N}}{\text{m}^2} = \frac{2354.4}{10^4} \frac{\text{N}}{\text{cm}^2}.$$

$$= \mathbf{0.2354} \frac{\mathbf{N}}{\mathbf{cm^2}}. \textbf{ Ans.}$$

(*c*) For mercury, sp. gr. $= 13.6$

From equation (1.1*A*) we know that the density of a fluid is equal to specific gravity of fluid multiplied by density of water

$\therefore$ Density of mercury, $\rho_s = \text{Specific gravity of mercury} \times \text{Density of water}$

$= 13.6 \times 1000 = 13600 \text{ kg/m}^3$

$\therefore$ $p = \rho_s \times g \times Z$

$$= 13600 \times 9.81 \times 0.3 = 40025 \frac{\text{N}}{\text{m}^2}$$

$$= \frac{40025}{10^4} = \mathbf{4.002} \frac{\mathbf{N}}{\mathbf{cm^2}}. \textbf{ Ans.}$$

Problem 2.4 *The pressure intensity at a point in a fluid is given 3.924 N/cm^2. Find the corresponding height of fluid when the fluid is : (a) water, and (b) oil of sp. gr. 0.9.*

Solution. Given :

Pressure intensity, $p = 3.924 \frac{\text{N}}{\text{cm}^2} = 3.924 \times 10^4 \frac{\text{N}}{\text{m}^2}$.

The corresponding height, Z, of the fluid is given by equation (2.6) as

$$Z = \frac{p}{\rho \times g}$$

(*a*) For water, $\rho = 1000 \text{ kg/m}^3$

$\therefore$ $$Z = \frac{p}{\rho \times g} = \frac{3.924 \times 10^4}{1000 \times 9.81} = \mathbf{4 \text{ m of water}}. \textbf{ Ans.}$$

(*b*) For oil, sp. gr. $= 0.9$

$\therefore$ Density of oil $\rho_0 = 0.9 \times 1000 = 900 \text{ kg/m}^3$

$\therefore$ $$Z = \frac{p}{\rho_0 \times g} = \frac{3.924 \times 10^4}{900 \times 9.81} = \mathbf{4.44 \text{ m of oil}}. \textbf{ Ans.}$$

Problem 2.5 *An oil of sp. gr. 0.9 is contained in a vessel. At a point the height of oil is 40 m. Find the corresponding height of water at the point.*

Solution. Given :

Sp. gr. of oil, $S_0 = 0.9$

Height of oil, $Z_0 = 40$ m

Density of oil, ρ_0 = Sp. gr. of oil × Density of water = $0.9 \times 1000 = 900$ kg/m^3

Intensity of pressure, $p = \rho_0 \times g \times Z_0 = 900 \times 9.81 \times 40 \ \frac{\text{N}}{\text{m}^2}$

$\therefore$ Corresponding height of water = $\dfrac{p}{\text{Density of water} \times g}$

$= \dfrac{900 \times 9.81 \times 40}{1000 \times 9.81} = 0.9 \times 40 =$ **36 m of water. Ans.**

Problem 2.6 *An open tank contains water upto a depth of 2 m and above it an oil of sp. gr. 0.9 for a depth of 1 m. Find the pressure intensity (i) at the interface of the two liquids, and (ii) at the bottom of the tank.*

Solution. Given :

Height of water, $Z_1 = 2$ m

Height of oil, $Z_2 = 1$ m

Sp. gr. of oil, $S_0 = 0.9$

Density of water, $\rho_1 = 1000$ kg/m^3

Density of oil, ρ_2 = Sp. gr. of oil × Density of water

$= 0.9 \times 1000 = 900$ kg/m^3

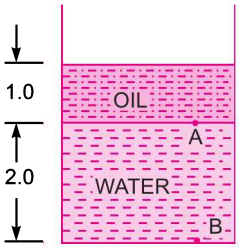


Fig. 2.4

Pressure intensity at any point is given by

$p = \rho \times g \times Z.$

(*i*) At interface, *i.e.*, at *A*

$p = \rho_2 \times g \times 1.0$

$= 900 \times 9.81 \times 1.0$

$= 8829 \ \frac{\text{N}}{\text{m}^2} = \frac{8829}{10^4} =$ **0.8829 N/cm^2. Ans.**

(*ii*) At the bottom, *i.e.*, at *B*

$p = \rho_2 \times gZ_2 + \rho_1 \times g \times Z_1 = 900 \times 9.81 \times 1.0 + 1000 \times 9.81 \times 2.0$

$= 8829 + 19620 = 28449$ N/m$^2 = \frac{28449}{10^4}$ N/cm$^2 =$ **2.8449 N/cm^2. Ans.**

Problem 2.7 *The diameters of a small piston and a large piston of a hydraulic jack are 3 cm and 10 cm respectively. A force of 80 N is applied on the small piston. Find the load lifted by the large piston when :*

(a) the pistons are at the same level.

(b) small piston is 40 cm above the large piston.

The density of the liquid in the jack is given as 1000 kg/m^3.

Solution. Given :

Dia. of small piston, $d = 3$ cm

$\therefore$ Area of small piston, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} \times (3)^2 = 7.068$ cm^2

Dia. of large piston, $D = 10$ cm

$\therefore$ Area of larger piston, $A = \frac{p}{4} \times (10)^2 = 78.54 \text{ cm}^2$

Force on small piston, $F = 80$ N

Let the load lifted $= W$.

(*a*) **When the pistons are at the same level**

Pressure intensity on small piston

$$\frac{F}{a} = \frac{80}{7.068} \text{ N/cm}^2$$

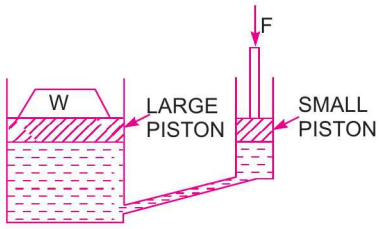


Fig. 2.5

This is transmitted equally on the large piston.

$\therefore$ Pressure intensity on the large piston

$$= \frac{80}{7.068}$$

$\therefore$ Force on the large piston = Pressure × Area

$$= \frac{80}{7.068} \times 78.54 \text{ N} = \mathbf{888.96 \text{ N. Ans.}}$$

(*b*) **When the small piston is 40 cm above the large piston**

Pressure intensity on the small piston

$$= \frac{F}{a} = \frac{80}{7.068} \frac{\text{N}}{\text{cm}^2}$$

$\therefore$ Pressure intensity at section *A-A*

$= \frac{F}{a}$ + Pressure intensity due to height of 40 cm of liquid.

But pressure intensity due to 40 cm of liquid

$$= \rho \times g \times h = 1000 \times 9.81 \times 0.4 \text{ N/m}^2$$

$$= \frac{1000 \times 9.81 \times .40}{10^4} \text{ N/cm}^2 = 0.3924 \text{ N/cm}^2$$

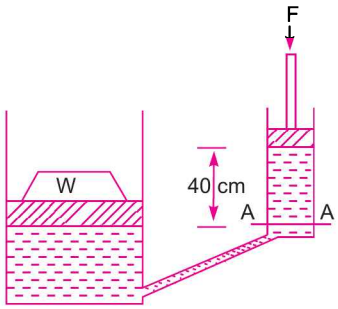


Fig. 2.6

$\therefore$ Pressure intensity at section *A-A*

$$= \frac{80}{7.068} + 0.3924$$

$$= 11.32 + 0.3924 = 11.71 \text{ N/cm}^2$$

$\therefore$ Pressure intensity transmitted to the large piston = 11.71 N/cm^2

$\therefore$ Force on the large piston = Pressure × Area of the large piston

$$= 11.71 \times A = 11.71 \times 78.54 = 919.7 \text{ N.}$$

▶ 2.4 ABSOLUTE, GAUGE, ATMOSPHERIC AND VACUUM PRESSURES

The pressure on a fluid is measured in two different systems. In one system, it is measured above the absolute zero or complete vacuum and it is called the absolute pressure and in other system, pressure is measured above the atmospheric pressure and it is called gauge pressure. Thus :

1. Absolute pressure is defined as the pressure which is measured with reference to absolute vacuum pressure.

2. Gauge pressure is defined as the pressure which is measured with the help of a pressure measuring instrument, in which the atmospheric pressure is taken as datum. The atmospheric pressure on the scale is marked as zero.

3. Vacuum pressure is defined as the pressure below the atmospheric pressure.

The relationship between the absolute pressure, gauge pressure and vacuum pressure are shown in Fig. 2.7.

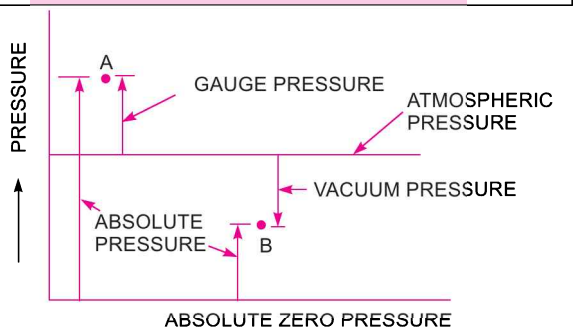


Fig. 2.7 *Relationship between pressures.*

Mathematically :

(*i*) Absolute pressure

= Atmospheric pressure + Gauge pressure

or $$p_{ab} = p_{atm} + p_{gauge}$$

(*ii*) Vacuum pressure

= Atmospheric pressure – Absolute pressure.

Note. (*i*) The atmospheric pressure at sea level at 15°C is 101.3 kN/m^2 or 10.13 N/cm^2 in SI unit. In case of MKS units, it is equal to 1.033 kgf/cm^2.

(*ii*) The atmospheric pressure head is 760 mm of mercury or 10.33 m of water.

Problem 2.8 *What are the gauge pressure and absolute pressure at a point 3 m below the free surface of a liquid having a density of 1.53 × 10^3 kg/m^3 if the atmospheric pressure is equivalent to 750 mm of mercury ? The specific gravity of mercury is 13.6 and density of water = 1000 kg/m^3.*

Solution. Given :

Depth of liquid, $Z_1 = 3$ m

Density of liquid, $\rho_1 = 1.53 \times 10^3$ kg/m^3

Atmospheric pressure head, $Z_0 = 750$ mm of Hg

$$= \frac{750}{1000} = 0.75 \text{ m of Hg}$$

∴ Atmospheric pressure, $p_{atm} = \rho_0 \times g \times Z_0$

where ρ_0 = Density of Hg = Sp. gr. of mercury × Density of water = 13.6 × 1000 kg/m^3

and Z_0 = Pressure head in terms of mercury.

∴ $$p_{atm} = (13.6 \times 1000) \times 9.81 \times 0.75 \text{ N/m}^2 \quad (\because Z_0 = 0.75)$$

$$= 100062 \text{ N/m}^2$$

Pressure at a point, which is at a depth of 3 m from the free surface of the liquid is given by,

$$p = \rho_1 \times g \times Z_1$$

$$= (1.53 \times 1000) \times 9.81 \times 3 = 45028 \text{ N/m}^2$$

∴ Gauge pressure, $p =$ **45028 N/m^2. Ans.**

Now absolute pressure = Gauge pressure + Atmospheric pressure

= 45028 + 100062 = **145090 N/m^2. Ans.**

▶ 2.5 MEASUREMENT OF PRESSURE

The pressure of a fluid is measured by the following devices :

1. Manometers 2. Mechanical Gauges.

2.5.1 Manometers. Manometers are defined as the devices used for measuring the pressure at a point in a fluid by balancing the column of fluid by the same or another column of the fluid. They are classified as :

(*a*) Simple Manometers, (*b*) Differential Manometers.

2.5.2 Mechanical Gauges. Mechanical gauges are defined as the devices used for measuring the pressure by balancing the fluid column by the spring or dead weight. The commonly used mechanical pressure gauges are :

(*a*) Diaphragm pressure gauge, (*b*) Bourdon tube pressure gauge,
(*c*) Dead-weight pressure gauge, and (*d*) Bellows pressure gauge.

▶ 2.6 SIMPLE MANOMETERS

A simple manometer consists of a glass tube having one of its ends connected to a point where pressure is to be measured and other end remains open to atmosphere. Common types of simple manometers are :

1. Piezometer,`
2. U-tube Manometer, and
3. Single Column Manometer.

2.6.1 Piezometer. It is the simplest form of manometer used for measuring gauge pressures. One end of this manometer is connected to the point where pressure is to be measured and other end is open to the atmosphere as shown in Fig. 2.8. The rise of liquid gives the pressure head at that point. If at a point *A*, the height of liquid say water is *h* in piezometer tube, then pressure at *A*

$$= \rho \times g \times h \ \frac{\text{N}}{\text{m}^2}.$$

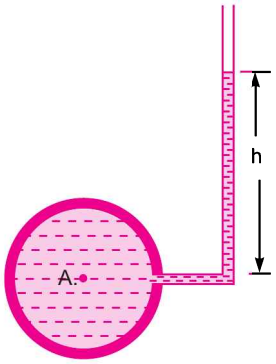
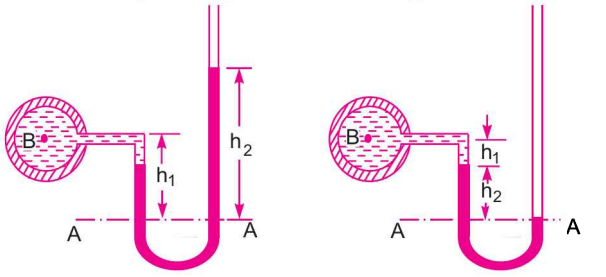


Fig. 2.8 *Piezometer.*

2.6.2 U-tube Manometer. It consists of glass tube bent in U-shape, one end of which is connected to a point at which pressure is to be measured and other end remains open to the atmosphere as shown in Fig. 2.9. The tube generally contains mercury or any other liquid whose specific gravity is greater than the specific gravity of the liquid whose pressure is to be measured.



(a) For gauge pressure (b) For vacuum pressure

Fig. 2.9 *U-tube Manometer.*

(*a*) **For Gauge Pressure.** Let *B* is the point at which pressure is to be measured, whose value is *p*. The datum line is *A*-*A*.

Let h_1 = Height of light liquid above the datum line
h_2 = Height of heavy liquid above the datum line
S_1 = Sp. gr. of light liquid
ρ_1 = Density of light liquid = $1000 \times S_1$
S_2 = Sp. gr. of heavy liquid
ρ_2 = Density of heavy liquid = $1000 \times S_2$

As the pressure is the same for the horizontal surface. Hence pressure above the horizontal datum line *A-A* in the left column and in the right column of U-tube manometer should be same.

Pressure above *A-A* in the left column $= p + \rho_1 \times g \times h_1$

Pressure above *A-A* in the right column $= \rho_2 \times g \times h_2$

Hence equating the two pressures $p + \rho_1 g h_1 = \rho_2 g h_2$

$\therefore$ $p = (\rho_2 g h_2 - \rho_1 \times g \times h_1)$. ...(2.7)

(*b*) **For Vacuum Pressure.** For measuring vacuum pressure, the level of the heavy liquid in the manometer will be as shown in Fig. 2.9 (*b*). Then

Pressure above *A-A* in the left column $= \rho_2 g h_2 + \rho_1 g h_1 + p$

Pressure head in the right column above *A-A* $= 0$

$\therefore$ $\rho_2 g h_2 + \rho_1 g h_1 + p = 0$

$\therefore$ $p = -(\rho_2 g h_2 + \rho_1 g h_1)$. ...(2.8)

Problem 2.9 *The right limb of a simple U-tube manometer containing mercury is open to the atmosphere while the left limb is connected to a pipe in which a fluid of sp. gr. 0.9 is flowing. The centre of the pipe is 12 cm below the level of mercury in the right limb. Find the pressure of fluid in the pipe if the difference of mercury level in the two limbs is 20 cm.*

Solution. Given :

Sp. gr. of fluid, $S_1 = 0.9$

$\therefore$ Density of fluid, $\rho_1 = S_1 \times 1000 = 0.9 \times 1000 = 900$ kg/m^3

Sp. gr. of mercury, $S_2 = 13.6$

$\therefore$ Density of mercury, $\rho_2 = 13.6 \times 1000$ kg/m^3

Difference of mercury level, $h_2 = 20$ cm $= 0.2$ m

Height of fluid from *A-A*, $h_1 = 20 - 12 = 8$ cm $= 0.08$ m

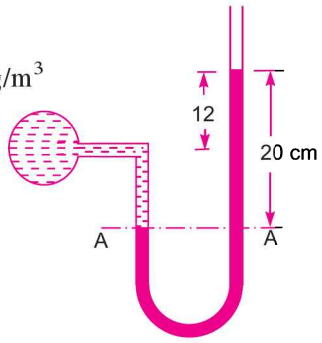


Fig. 2.10

Let p = Pressure of fluid in pipe

Equating the pressure above *A-A*, we get

$$p + \rho_1 g h_1 = \rho_2 g h_2$$

or $$p + 900 \times 9.81 \times 0.08 = 13.6 \times 1000 \times 9.81 \times .2$$

$$p = 13.6 \times 1000 \times 9.81 \times .2 - 900 \times 9.81 \times 0.08$$

$$= 26683 - 706 = 25977 \text{ N/m}^2 = \mathbf{2.597 \text{ N/cm}^2}. \textbf{ Ans.}$$

Problem 2.10 *A simple U-tube manometer containing mercury is connected to a pipe in which a fluid of sp. gr. 0.8 and having vacuum pressure is flowing. The other end of the manometer is open to atmosphere. Find the vacuum pressure in pipe, if the difference of mercury level in the two limbs is 40 cm and the height of fluid in the left from the centre of pipe is 15 cm below.*

Solution. Given :

Sp. gr. of fluid, $S_1 = 0.8$

Sp. gr. of mercury, $S_2 = 13.6$

Density of fluid, $\rho_1 = 800$

Density of mercury, $\rho_2 = 13.6 \times 1000$

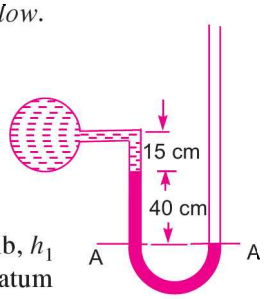


Fig. 2.11

Difference of mercury level, $h_2 = 40$ cm $= 0.4$ m. Height of liquid in left limb, $h_1 = 15$ cm $= 0.15$ m. Let the pressure in pipe $= p$. Equating pressure above datum line *A-A*, we get

$$\rho_2 g h_2 + \rho_1 g h_1 + p = 0$$

$\therefore$ $p = -[\rho_2 g h_2 + \rho_1 g h_1]$

$= -[13.6 \times 1000 \times 9.81 \times 0.4 + 800 \times 9.81 \times 0.15]$

$= -[53366.4 + 1177.2] = -54543.6$ N/m^2 = **– 5.454 N/cm^2. Ans.**

Problem 2.11 *A U-Tube manometer is used to measure the pressure of water in a pipe line, which is in excess of atmospheric pressure. The right limb of the manometer contains mercury and is open to atmosphere. The contact between water and mercury is in the left limb. Determine the pressure of water in the main line, if the difference in level of mercury in the limbs of U-tube is 10 cm and the free surface of mercury is in level with the centre of the pipe. If the pressure of water in pipe line is reduced to 9810 N/m^2, calculate the new difference in the level of mercury. Sketch the arrangements in both cases.*

Solution. Given :

Difference of mercury = 10 cm = 0.1 m

The arrangement is shown in Fig. 2.11 (*a*)

Ist Part

Let p_A = (pressure of water in pipe line (*i.e.,* at point *A*)

The points *B* and *C* lie on the same horizontal line. Hence pressure at *B* should be equal to pressure at *C*. But pressure at *B*

= Pressure at *A* + Pressure due to 10 cm (or 0.1 m) of water

$= p_A + \rho \times g \times h$

where $\rho = 1000$ kg/m^3 and $h = 0.1$ m

$= p_A + 1000 \times 9.81 \times 0.1$

$= p_A + 981$ N/m^2 ...(*i*)

Pressure at *C* = Pressure at *D* + Pressure due to 10 cm of mercury

$= 0 + \rho_0 \times g \times h_0$

where ρ_0 for mercury $= 13.6 \times 1000$ kg/m^3

and $h_0 = 10$ cm = 0.1 m

$\therefore$ Pressure at $C = 0 + (13.6 \times 1000) \times 9.81 \times 0.1$

$= 13341.6$ N ...(*ii*)

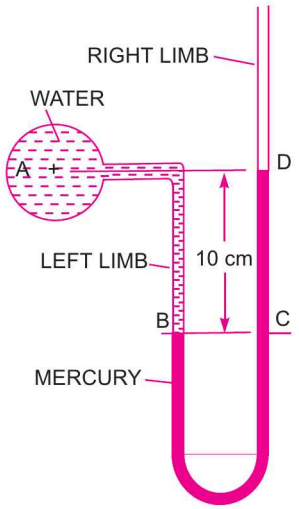


Fig. 2.11 (*a*)

But pressure at *B* is equal to pressure at *C*. Hence equating the equations (*i*) and (*ii*), we get

$p_A + 981 = 13341.6$

$\therefore$ $p_A = 13341.6 - 981$

$= \mathbf{12360.6 \frac{N}{m^2}}$ **. Ans.**

IInd Part

Given, $p_A = 9810$ N/m^2

Find new difference of mercury level. The arrangement is shown in Fig. 2.11 (*b*). In this case the pressure at *A* is 9810 N/m^2 which is less than the 12360.6 N/m^2. Hence mercury in left limb will rise. The rise of mercury in left limb will be equal to the fall of mercury in right limb as the total volume of mercury remains same.

Let x = Rise of mercury in left limb in cm

Then fall of mercury in right limb = x cm

The points *B*, *C* and *D* show the initial conditions whereas points *B**, *C** and *D** show the final conditions.

The pressure at B^* = Pressure at C^*

or Pressure at A + Pressure due to $(10 - x)$ cm of water
= Pressure at D^* + Pressure due to
$(10 - 2x)$ cm of mercury

or $p_A + \rho_1 \times g \times h_1 = p_D{}^* + \rho_2 \times g \times h_2$

or $1910 + 1000 \times 9.81 \times \left(\frac{10 - x}{100}\right)$

$= 0 + (13.6 \times 1000) \times 9.81 \times \left(\frac{10 - 2x}{100}\right)$

Dividing by 9.81, we get

or $1000 + 100 - 10x = 1360 - 272x$

or $272x - 10x = 1360 - 1100$

or $262x = 260$

$\therefore \quad x = \frac{260}{262} = 0.992$ cm

$\therefore$ New difference of mercury = $10 - 2x$ cm $= 10 - 2 \times 0.992$
$= \mathbf{8.016\ cm.\ Ans.}$

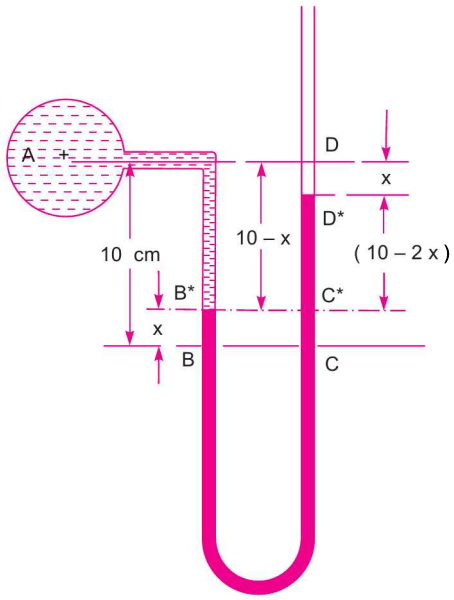


Fig. 2.11 (*b*)

Problem 2.12 *Fig. 2.12 shows a conical vessel having its outlet at A to which a U-tube manometer is connected. The reading of the manometer given in the figure shows when the vessel is empty. Find the reading of the manometer when the vessel is completely filled with water.*

Solution. Vessel is empty. Given :

Difference of mercury level $h_2 = 20$ cm

Let h_1 = Height of water above X-X

Sp. gr. of mercury, $S_2 = 13.6$

Sp. gr. of water, $S_1 = 1.0$

Density of mercury, $\rho_2 = 13.6 \times 1000$

Density of water, $\rho_1 = 1000$

Equating the pressure above datum line X-X, we have

$\rho_2 \times g \times h_2 = \rho_1 \times g \times h_1$

or $13.6 \times 1000 \times 9.81 \times 0.2 = 1000 \times 9.81 \times h_1$

$h_1 = 2.72$ m of water.

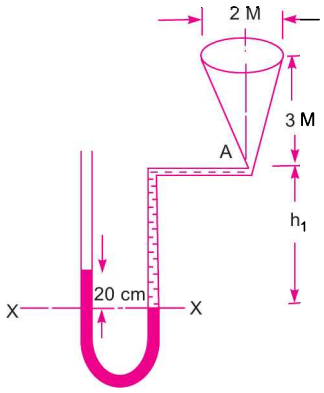


Fig. 2.12

Vessel is full of water. When vessel is full of water, the pressure in the right limb will increase and mercury level in the right limb will go down. Let the distance through which mercury goes down in the right limb be, y cm as shown in Fig. 2.13. The mercury will rise in the left by a distance of y cm. Now the datum line is Z-Z. Equating the pressure above the datum line Z-Z.

Pressure in left limb = Pressure in right limb

$13.6 \times 1000 \times 9.81 \times (0.2 + 2y/100)$

$= 1000 \times 9.81 \times (3 + h_1 + y/100)$

or $13.6 \times (0.2 + 2y/100) = (3 + 2.72 + y/100)$ $(\because \quad h_1 = 2.72 \text{ cm})$

or $2.72 + 27.2y/100 = 3 + 2.72 + y/100$

or $(27.2y - y)/100 = 3.0$

or $26.2y = 3 \times 100 = 300$

$\therefore \quad y = \dfrac{300}{26.2} = 11.45 \text{ cm}$

The difference of mercury level in two limbs

$= (20 + 2y)$ cm of mercury

$= 20 + 2 \times 11.45 = 20 + 22.90$

$= 42.90$ cm of mercury

$\therefore$ Reading of manometer = **42.90 cm. Ans.**

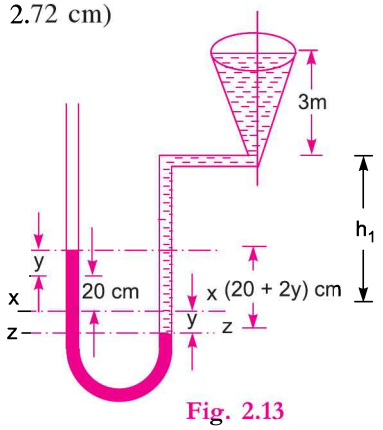


Fig. 2.13

Problem 2.13 *A pressure gauge consists of two cylindrical bulbs B and C each of 10 sq. cm cross-sectional area, which are connected by a U-tube with vertical limbs each of 0.25 sq. cm cross-sectional area. A red liquid of specific gravity 0.9 is filled into C and clear water is filled into B, the surface of separation being in the limb attached to C. Find the displacement of the surface of separation when the pressure on the surface in C is greater than that in B by an amount equal to 1 cm head of water.*

Solution. Given :

Area of each bulb B and C, $A = 10 \text{ cm}^2$

Area of each vertical limb, $a = 0.25 \text{ cm}^2$

Sp. gr. of red liquid $= 0.9$ $\therefore$ Its density $= 900 \text{ kg/m}^3$

Let X-X = Initial separation level

h_C = Height of red liquid above X-X

h_B = Height of water above X-X

Pressure above X-X in the left limb $= 1000 \times 9.81 \times h_B$

Pressure above X-X in the right limb $= 900 \times 9.81 \times h_C$

Equating the two pressure, we get

$1000 \times 9.81 \times h_B = 900 \times 9.81 \times h_C$

$\therefore \quad h_B = 0.9\, h_C \quad ...(i)$

When the pressure head over the surface in C is increased by 1 cm of water, let the separation level falls by an amount equal to Z. Then Y-Y becomes the final separation level.

Now fall in surface level of C multiplied by cross-sectional area of bulb C must be equal to the fall in separation level multiplied by cross-sectional area of limb.

$\therefore$ Fall in surface level of C

$$= \frac{\text{Fall in separation level} \times a}{A}$$

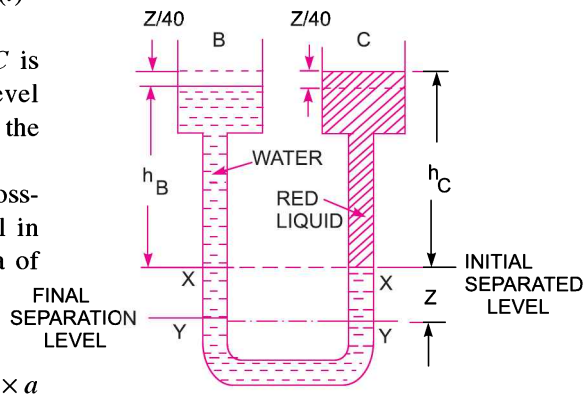


Fig. 2.14

$$= \frac{Z \times a}{A} = \frac{Z \times 0.25}{10} = \frac{Z}{40}.$$

Also fall in surface level of C

= Rise in surface level of B

$$= \frac{Z}{40}$$

The pressure of 1 cm (or 0.01 m) of water = ρgh = 1000 × 9.81 × 0.01= 98.1 N/m^2

Consider final separation level Y-Y

Pressure above Y-Y in the left limb = $1000 \times 9.81 \left(Z + h_B + \frac{Z}{40} \right)$

Pressure above Y-Y in the right limb = $900 \times 9.81 \left(Z + h_C - \frac{Z}{40} \right) + 98.1$

Equating the two pressure, we get

$$1000 \times 9.81 \left(Z + h_B + \frac{Z}{40} \right) = \left(Z + h_C - \frac{Z}{40} \right) 900 \times 9.81 + 98.1$$

Dividing by 9.81, we get

$$1000 \left(Z + h_B + \frac{Z}{40} \right) = 900 \left(Z + h_C - \frac{Z}{40} \right) + 10$$

Dividing by 1000, we get $Z + h_B + \frac{Z}{40} = 0.9 \left(Z + h_C - \frac{Z}{40} \right) + 0.01$

But from equation (i), $h_B = 0.9\, h_C$

$$\therefore \quad Z + 0.9\, h_C + \frac{Z}{40} = \frac{39Z}{40} \times 0.9 + 0.9\, h_C + 0.01$$

or $$\frac{41Z}{40} = \frac{39}{40} \times .9Z + .01$$

or $$Z \left(\frac{41}{40} - \frac{39 \times .9}{40} \right) = .01 \quad \text{or} \quad Z \left(\frac{41 - 35.1}{40} \right) = .01$$

$$\therefore \quad Z = \frac{40 \times 0.01}{5.9} = \mathbf{0.0678\ m = 6.78\ cm.\ Ans.}$$

2.6.3 Single Column Manometer. Single column manometer is a modified form of a U-tube manometer in which a reservoir, having a large cross-sectional area (about 100 times) as compared to the area of the tube is connected to one of the limbs (say left limb) of the manometer as shown in Fig. 2.15. Due to large cross-sectional area of the reservoir, for any variation in pressure, the change in the liquid level in the reservoir will be very small which may be neglected and hence the pressure is given by the height of liquid in the other limb. The other limb may be vertical or inclined. Thus there are two types of single column manometer as :

1. Vertical Single Column Manometer.
2. Inclined Single Column Manometer.

1. Vertical Single Column Manometer

Fig. 2.15 shows the vertical single column manometer. Let X-X be the datum line in the reservoir and in the right limb of the manometer, when it is not connected to the pipe. When the manometer is

connected to the pipe, due to high pressure at A, the heavy liquid in the reservoir will be pushed downward and will rise in the right limb.

Let Δh = Fall of heavy liquid in reservoir

h_2 = Rise of heavy liquid in right limb

h_1 = Height of centre of pipe above X-X

p_A = Pressure at A, which is to be measured

A = Cross-sectional area of the reservoir

a = Cross-sectional area of the right limb

S_1 = Sp. gr. of liquid in pipe

S_2 = Sp. gr. of heavy liquid in reservoir and right limb

ρ_1 = Density of liquid in pipe

ρ_2 = Density of liquid in reservoir

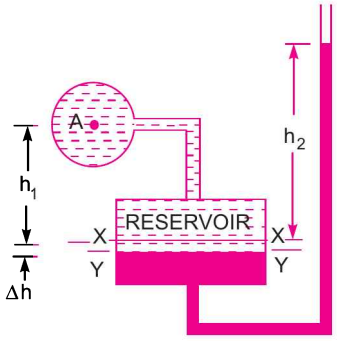


Fig. 2.15 *Vertical single column manometer.*

Fall of heavy liquid in reservoir will cause a rise of heavy liquid level in the right limb.

$$\therefore \quad A \times \Delta h = a \times h_2$$

$$\therefore \quad \Delta h = \frac{a \times h_2}{A} \qquad ...(i)$$

Now consider the datum line Y-Y as shown in Fig. 2.15. Then pressure in the right limb above Y-Y.

$$= \rho_2 \times g \times (\Delta h + h_2)$$

Pressure in the left limb above Y-Y $= \rho_1 \times g \times (\Delta h + h_1) + p_A$

Equating these pressures, we have

$$\rho_2 \times g \times (\Delta h + h_2) = \rho_1 \times g \times (\Delta h + h_1) + p_A$$

or
$$p_A = \rho_2 g\,(\Delta h + h_2) - \rho_1 g(\Delta h + h_1)$$
$$= \Delta h[\rho_2 g - \rho_1 g] + h_2\rho_2 g - h_1\rho_1 g$$

But from equation (i), $\Delta h = \dfrac{a \times h_2}{A}$

$$\therefore \quad p_A = \frac{a \times h_2}{A}\,[\rho_2 g - \rho_1 g] + h_2\rho_2 g - h_1\rho_1 g \qquad ...(2.9)$$

As the area A is very large as compared to a, hence ratio $\dfrac{a}{A}$ becomes very small and can be neglected.

Then $p_A = h_2\rho_2 g - h_1\rho_1 g$ ...(2.10)

From equation (2.10), it is clear that as h_1 is known and hence by knowing h_2 or rise of heavy liquid in the right limb, the pressure at A can be calculated.

2. Inclined Single Column Manometer

Fig. 2.16 shows the inclined single column manometer. This manometer is more sensitive. Due to inclination the distance moved by the heavy liquid in the right limb will be more.

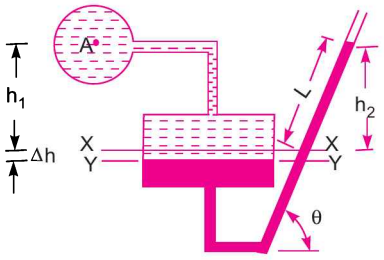


Fig. 2.16 *Inclined single column manometer.*

Let L = Length of heavy liquid moved in right limb from X-X

θ = Inclination of right limb with horizontal

h_2 = Vertical rise of heavy liquid in right limb from X-X = $L \times \sin \theta$

From equation (2.10), the pressure at A is

$$p_A = h_2\rho_2 g - h_1\rho_1 g.$$

Substituting the value of h_2, we get

$$p_A = \sin \theta \times \rho_2 g - h_1\rho_1 g. \qquad ...(2.11)$$

Problem 2.14 *A single column manometer is connected to a pipe containing a liquid of sp. gr. 0.9 as shown in Fig. 2.17. Find the pressure in the pipe if the area of the reservoir is 100 times the area of the tube for the manometer reading shown in Fig. 2.17. The specific gravity of mercury is 13.6.*

Solution. Given :

Sp. gr. of liquid in pipe, $S_1 = 0.9$

$\therefore$ Density $\rho_1 = 900 \text{ kg/m}^3$

Sp. gr. of heavy liquid, $S_2 = 13.6$

Density, $\rho_2 = 13.6 \times 1000$

$$\frac{\text{Area of reservoir}}{\text{Area of right limb}} = \frac{A}{a} = 100$$

Height of liquid, $h_1 = 20 \text{ cm} = 0.2 \text{ m}$

Rise of mercury in right limb, $h_2 = 40 \text{ cm} = 0.4 \text{ m}$

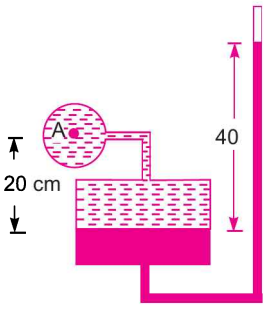


Fig. 2.17

Let p_A = Pressure in pipe

Using equation (2.9), we get

$$p_A = \frac{a}{A} h_2[\rho_2 g - \rho_1 g] + h_2\rho_2 g - h_1\rho_1 g$$

$$= \frac{1}{100} \times 0.4[13.6 \times 1000 \times 9.81 - 900 \times 9.81] + 0.4 \times 13.6 \times 1000 \times 9.81 - 0.2 \times 900 \times 9.81$$

$$= \frac{0.4}{100} [133416 - 8829] + 53366.4 - 1765.8$$

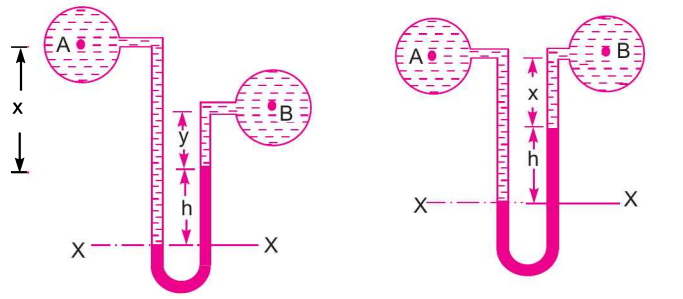
$$= 533.664 + 53366.4 - 1765.8 \text{ N/m}^2 = 52134 \text{ N/m}^2 = \mathbf{5.21 \text{ N/cm}^2}. \textbf{ Ans.}$$

▶ 2.7 DIFFERENTIAL MANOMETERS

Differential manometers are the devices used for measuring the difference of pressures between two points in a pipe or in two different pipes. A differential manometer consists of a U-tube, containing a heavy liquid, whose two ends are connected to the points, whose difference of pressure is to be measured. Most commonly types of differential manometers are :

1. U-tube differential manometer and
2. Inverted U-tube differential manometer.

2.7.1 U-tube Differential Manometer. Fig. 2.18 shows the differential manometers of U-tube type.



(a)Two pipes at different levels (b) A and B are at the same level

Fig. 2.18 *U-tube differential manometers.*

In Fig. 2.18 (*a*), the two points *A* and *B* are at different level and also contains liquids of different sp. gr. These points are connected to the U-tube differential manometer. Let the pressure at *A* and *B* are p_A and p_B.

Let h = Difference of mercury level in the U-tube.

y = Distance of the centre of *B*, from the mercury level in the right limb.

x = Distance of the centre of *A*, from the mercury level in the right limb.

ρ_1 = Density of liquid at *A*.

ρ_2 = Density of liquid at *B*.

ρ_g = Density of heavy liquid or mercury.

Taking datum line at *X-X*.

Pressure above *X-X* in the left limb = $\rho_1 g(h + x) + p_A$

where p_A = pressure at *A*.

Pressure above *X-X* in the right limb = $\rho_g \times g \times h + \rho_2 \times g \times y + p_B$

where p_B = Pressure at *B*.

Equating the two pressure, we have

$$\rho_1 g(h + x) + p_A = \rho_g \times g \times h + \rho_2 g y + p_B$$

$$\therefore \quad p_A - p_B = \rho_g \times g \times h + \rho_2 g y - \rho_1 g(h + x)$$

$$= h \times g(\rho_g - \rho_1) + \rho_2 g y - \rho_1 g x \qquad ...(2.12)$$

$\therefore$ Difference of pressure at *A* and *B* = $h \times g(\rho_g - \rho_1) + \rho_2 g y - \rho_1 g x$

In Fig. 2.18 (*b*), the two points *A* and *B* are at the same level and contains the same liquid of density ρ_1. Then

Pressure above *X-X* in right limb = $\rho_g \times g \times h + \rho_1 \times g \times x + p_B$

Pressure above *X-X* in left limb = $\rho_1 \times g \times (h + x) + p_A$

Equating the two pressure

$$\rho_g \times g \times h + \rho_1 g x + p_B = \rho_1 \times g \times (h + x) + p_A$$

$$\therefore \quad p_A - p_B = \rho_g \times g \times h + \rho_1 g x - \rho_1 g(h + x)$$

$$= g \times h(\rho_g - \rho_1). \qquad ...(2.13)$$

Problem 2.15 *A pipe contains an oil of sp. gr. 0.9. A differential manometer connected at the two points A and B shows a difference in mercury level as 15 cm. Find the difference of pressure at the two points.*

Solution. Given :

Sp. gr. of oil, $S_1 = 0.9$ $\therefore$ Density, $\rho_1 = 0.9 \times 1000 = 900$ kg/m^3

Difference in mercury level, $h = 15$ cm $= 0.15$ m

Sp. gr. of mercury, $S_g = 13.6$ $\therefore$ Density, $\rho_g = 13.6 \times 1000$ kg/m^3

The difference of pressure is given by equation (2.13)

or $$p_A - p_B = g \times h(\rho_g - \rho_1)$$
$$= 9.81 \times 0.15\ (13600 - 900) = \mathbf{18688\ N/m^2.\ Ans.}$$

Problem 2.16 *A differential manometer is connected at the two points A and B of two pipes as shown in Fig. 2.19. The pipe A contains a liquid of sp. gr. = 1.5 while pipe B contains a liquid of sp. gr. = 0.9. The pressures at A and B are 1 kgf/cm^2 and 1.80 kgf/cm^2 respectively. Find the difference in mercury level in the differential manometer.*

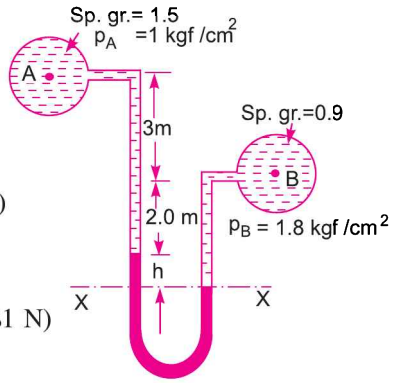


Fig. 2.19

Solution. Given :

Sp. gr. of liquid at A, $S_1 = 1.5$ $\therefore$ $\rho_1 = 1500$

Sp. gr. of liquid at B, $S_2 = 0.9$ $\therefore$ $\rho_2 = 900$

Pressure at A, $p_A = 1$ kgf/cm$^2 = 1 \times 10^4$ kgf/m^2

$= 10^4 \times 9.81$ N/m^2 ($\because$ 1 kgf = 9.81 N)

Pressure at B, $p_B = 1.8$ kgf/cm^2

$= 1.8 \times 10^4$ kgf/m^2

$= 1.8 \times 10^4 \times 9.81$ N/m^2 ($\because$ 1 kgf = 9.81 N)

Density of mercury $= 13.6 \times 1000$ kg/m^3

Taking X-X as datum line.

Pressure above X-X in the left limb

$$= 13.6 \times 1000 \times 9.81 \times h + 1500 \times 9.81 \times (2 + 3) + p_A$$
$$= 13.6 \times 1000 \times 9.81 \times h + 7500 \times 9.81 + 9.81 \times 10^4$$

Pressure above X-X in the right limb $= 900 \times 9.81 \times (h + 2) + p_B$

$$= 900 \times 9.81 \times (h + 2) + 1.8 \times 10^4 \times 9.81$$

Equating the two pressure, we get

$$13.6 \times 1000 \times 9.81h + 7500 \times 9.81 + 9.81 \times 10^4$$
$$= 900 \times 9.81 \times (h + 2) + 1.8 \times 10^4 \times 9.81$$

Dividing by 1000×9.81, we get

$$13.6h + 7.5 + 10 = (h + 2.0) \times .9 + 18$$

or $$13.6h + 17.5 = 0.9h + 1.8 + 18 = 0.9h + 19.8$$

or $$(13.6 - 0.9)h = 19.8 - 17.5 \text{ or } 12.7h = 2.3$$

$\therefore$ $$h = \frac{2.3}{12.7} = 0.181 \text{ m} = \mathbf{18.1\ cm.\ Ans.}$$

Problem 2.17 *A differential manometer is connected at the two points A and B as shown in Fig. 2.20. At B air pressure is 9.81 N/cm^2 (abs), find the absolute pressure at A.*

Solution. Given :

Air pressure at $B = 9.81$ N/cm^2

or $p_B = 9.81 \times 10^4$ N/m^2

Density of oil $= 0.9 \times 1000 = 900 \text{ kg/m}^3$

Density of mercury $= 13.6 \times 1000 \text{ kg/m}^3$

Let the pressure at A is p_A

Taking datum line at X-X

Pressure above X-X in the right limb

$$= 1000 \times 9.81 \times 0.6 + p_B$$
$$= 5886 + 98100 = 103986$$

Pressure above X-X in the left limb

$$= 13.6 \times 1000 \times 9.81 \times 0.1 + 900 \times 9.81 \times 0.2 + p_A$$
$$= 13341.6 + 1765.8 + p_A$$

Equating the two pressure heads

$$103986 = 13341.6 + 1765.8 + p_A$$

$$\therefore \quad p_A = 103986 - 15107.4 = 88876.8$$

$$\therefore \quad p_A = 88876.8 \text{ N/m}^2 = \frac{88876.8 \text{N}}{10000 \text{ cm}^2} = 8.887 \frac{\text{N}}{\text{cm}^2}.$$

$\therefore$ Absolute pressure at A = **8.887 N/cm^2. Ans.**

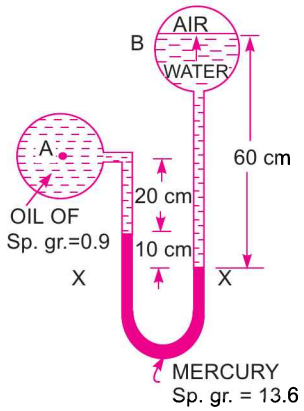


Fig. 2.20

2.7.2 Inverted U-tube Differential Manometer. It consists of an inverted U-tube, containing a light liquid. The two ends of the tube are connected to the points whose difference of pressure is to be measured. It is used for measuring difference of low pressures. Fig. 2.21 shows an inverted U-tube differential manometer connected to the two points A and B. Let the pressure at A is more than the pressure at B.

Let h_1 = Height of liquid in left limb below the datum line X-X

h_2 = Height of liquid in right limb

h = Difference of light liquid

ρ_1 = Density of liquid at A

ρ_2 = Density of liquid at B

ρ_s = Density of light liquid

p_A = Pressure at A

p_B = Pressure at B.

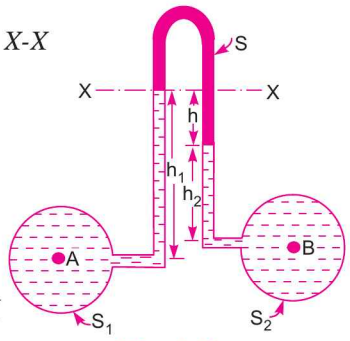


Fig. 2.21

Taking X-X as datum line. Then pressure in the left limb below X-X

$$= p_A - \rho_1 \times g \times h_1.$$

Pressure in the right limb below X-X

$$= p_B - \rho_2 \times g \times h_2 - \rho_s \times g \times h$$

Equating the two pressure

$$p_A - \rho_1 \times g \times h_1 = p_B - \rho_2 \times g \times h_2 - \rho_s \times g \times h$$

or
$$p_A - p_B = \rho_1 \times g \times h_1 - \rho_2 \times g \times h_2 - \rho_s \times g \times h. \qquad ...(2.14)$$

Problem 2.18 *Water is flowing through two different pipes to which an inverted differential manometer having an oil of sp. gr. 0.8 is connected. The pressure head in the pipe A is 2 m of water, find the pressure in the pipe B for the manometer readings as shown in Fig. 2.22.*

Solution. Given :

Pressure head at $A = \frac{p_A}{\rho g} = 2$ m of water

$$\therefore \quad p_A = \rho \times g \times 2 = 1000 \times 9.81 \times 2 = 19620 \text{ N/m}^2$$

Fig. 2.22 shows the arrangement. Taking X-X as datum line.

Pressure below X-X in the left limb $= p_A - \rho_1 \times g \times h_1$

$= 19620 - 1000 \times 9.81 \times 0.3 = 16677$ N/m^2.

Pressure below X-X in the right limb

$= p_B - 1000 \times 9.81 \times 0.1 - 800 \times 9.81 \times 0.12$

$= p_B - 981 - 941.76 = p_B - 1922.76$

Equating the two pressure, we get

$16677 = p_B - 1922.76$

or $p_B = 16677 + 1922.76 = 18599.76$ N/m^2

or $p_B =$ **1.8599 N/cm^2. Ans.**

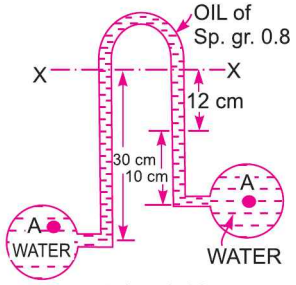


Fig. 2.22

Problem 2.19 *In Fig. 2.23, an inverted differential manometer is connected to two pipes A and B which convey water. The fluid in manometer is oil of sp. gr. 0.8. For the manometer readings shown in the figure, find the pressure difference between A and B.*

Solution. Given :

Sp. gr. of oil $= 0.8 \quad \therefore \quad \rho_s = 800$ kg/m^3

Difference of oil in the two limbs

$= (30 + 20) - 30 = 20$ cm

Taking datum line at X-X

Pressure in the left limb below X-X

$= p_A - 1000 \times 9.81 \times 0$

$= p_A - 2943$

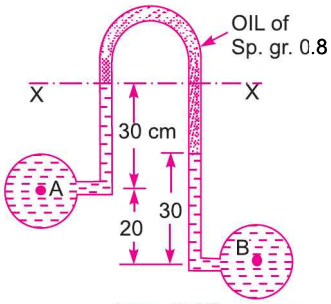


Fig. 2.23

Pressure in the right limb below X-X

$= p_B - 1000 \times 9.81 \times 0.3 - 800 \times 9.81 \times 0.2$

$= p_B - 2943 - 1569.6 = p_B - 4512.6$

Equating the two pressure $p_A - 2943 = p_B - 4512.6$

$\therefore \quad p_B - p_A = 4512.6 - 2943 =$ **1569.6 N/m^2. Ans.**

Problem 2.20 *Find out the differential reading 'h' of an inverted U-tube manometer containing oil of specific gravity 0.7 as the manometric fluid when connected across pipes A and B as shown in Fig. 2.24 below, conveying liquids of specific gravities 1.2 and 1.0 and immiscible with manometric fluid. Pipes A and B are located at the same level and assume the pressures at A and B to be equal.*

Solution. Given :

Fig. 2.24 shows the arrangement. Taking X-X as datum line.

Let $p_A =$ Pressure at A

$p_B =$ Pressure at B

Density of liquid in pipe A $= $ Sp. gr. $\times 1000$

$= 1.2 \times 1000$

$= 1200$ kg/m^2

Density of liquid in pipe B $= 1 \times 1000 = 1000$ kg/m^3

Density of oil $= 0.7 \times 1000 = 700$ kg/m^3

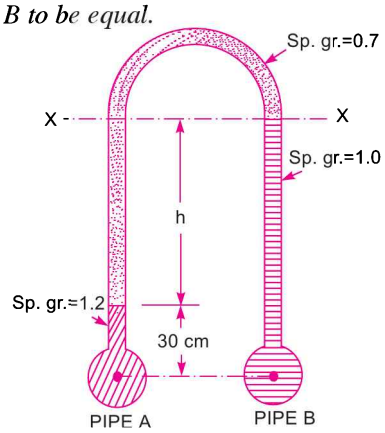


Fig. 2.24

Now pressure below X-X in the left limb

$$= p_A - 1200 \times 9.81 \times 0.3 - 700 \times 9.81 \times h$$

Pressure below X-X in the right limb

$$= p_B - 1000 \times 9.81 \times (h + 0.3)$$

Equating the two pressure, we get

$$p_A - 1200 \times 9.81 \times 0.3 - 700 \times 9.81 \times h = p_B - 1000 \times 9.81\ (h + 0.3)$$

But $p_A = p_B$ (given)

$\therefore$ $- 1200 \times 9.81 \times 0.3 - 700 \times 9.81 \times h = - 1000 \times 9.81\ (h + 0.3)$

Dividing by 1000×9.81

$$- 1.2 \times 0.3 - 0.7h = - (h + 0.3)$$

or $0.3 \times 1.2 + 0.7h = h + 0.3$ or $0.36 - 0.3 = h - 0.7h = 0.3h$

$\therefore$ $$h = \frac{0.36 - 0.30}{0.30} = \frac{0.06}{0.30} \text{ m}$$

$$= \frac{1}{5} \text{ m} = \frac{1}{5} \times 100 = \mathbf{20\ cm.\ Ans.}$$

Problem 2.21 *An inverted U-tube manometer is connected to two horizontal pipes A and B through which water is flowing. The vertical distance between the axes of these pipes is 30 cm. When an oil of specific gravity 0.8 is used as a gauge fluid, the vertical heights of water columns in the two limbs of the inverted manometer (when measured from the respective centre lines of the pipes) are found to be same and equal to 35 cm. Determine the difference of pressure between the pipes.*

Solution. Given :

Specific gravity of measuring liquid = 0.8

The arrangement is shown in Fig. 2.24 (a).

Let p_A = pressure at A

p_B = pressure at B.

The points C and D lie on the same horizontal line.

Hence pressure at C should be equal to pressure at D.

But pressure at C $= p_A - \rho g\ h$

$= p_A - 1000 \times 9.81 \times (0.35)$

And pressure at D $= p_B - \rho_1 g h_1 - \rho_2 g h_2$

$= p_B - 1000 \times 9.81 \times (0.35) - 800 \times 9.81 \times 0.3$

But pressure at C = pressure at D

$\therefore$ $p_A - 1000 \times 9.81 \times .35$

$= p_B - 1000 \times 9.81 \times 0.35 - 800 \times 9.81 \times 0.3$

or $800 \times 9.81 \times 0.3 = p_B - p_A$

or $$p_B - p_A = 800 \times 9.81 \times 0.3 = \mathbf{2354.4\ \frac{N}{m^2}}\text{. Ans.}$$

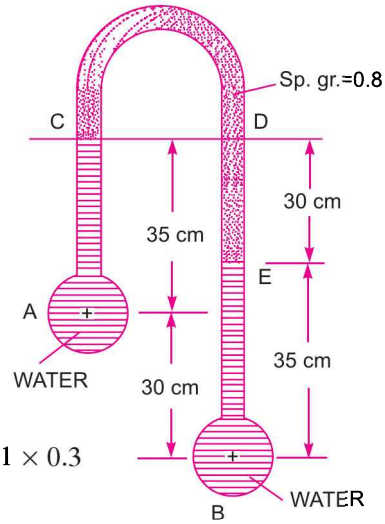


Fig. 2.24 *(a)*

▶ 2.8 PRESSURE AT A POINT IN COMPRESSIBLE FLUID

For compressible fluids, density (ρ) changes with the change of pressure and temperature. Such problems are encountered in aeronautics, oceanography and meteorology where we are concerned with atmospheric* air where density, pressure and temperature changes with elevation. Thus for fluids with variable density, equation (2.4) cannot be integrated, unless the relationship between p and ρ is known. For gases the equation of state is

$$\frac{p}{\rho} = RT \qquad ...(2.15)$$

or $$\rho = \frac{p}{RT}$$

Now equation (2.4) is $$\frac{dp}{dZ} = w = \rho g = \frac{p}{RT} \times g$$

$\therefore$ $$\frac{dp}{p} = \frac{g}{RT} dZ \qquad ...(2.16)$$

In equation (2.4), Z is measured vertically downward. But if Z is measured vertically up, then $\frac{dp}{dZ} = -\rho g$ and hence equation (2.16) becomes

$$\frac{dp}{p} = \frac{-g}{RT} dZ \qquad ...(2.17)$$

2.8.1 Isothermal Process. **Case I.** If temperature T is constant which is true for **isothermal process**, equation (2.17) can be integrated as

$$\int_{p_0}^{p} \frac{dp}{p} = -\int_{Z_0}^{Z} \frac{g}{RT} dz = -\frac{g}{RT}\int_{Z_0}^{Z} dz$$

or $$\log \frac{p}{p_0} = \frac{-g}{RT}[Z - Z_0]$$

where p_0 is the pressure where height is Z_0. If the datum line is taken at Z_0, then $Z_0 = 0$ and p_0 becomes the pressure at datum line.

$\therefore$ $$\log \frac{p}{p_0} = \frac{-g}{RT} Z$$

$$\frac{p}{p_0} = e^{-gZ/RT}$$

or pressure at a height Z is given by $p = p_0 e^{-gZ/RT}$ ...(2.18)

2.8.2 Adiabatic Process. If temperature T is not constant but the process follows adiabatic law then the relation between pressure and density is given by

$$\frac{p}{\rho^k} = \text{Constant} = C \qquad ...(i)$$

---

* The standard atmospheric pressure, temperature and density referred to STP at the sea-level are : Pressure = 101.325 kN/m^2 ; Temperature = 15°C and Density = 1.225 kg/m^3.

where k is ratio of specific constant.

$$\therefore \quad \rho^k = \frac{p}{C}$$

or
$$\rho = \left(\frac{p}{C}\right)^{1/k} \qquad ...(ii)$$

Then equation (2.4) for Z measured vertically up becomes,

$$\frac{dp}{dZ} = -\rho g = -\left(\frac{p}{C}\right)^{1/k} g$$

or
$$\frac{dp}{\left(\frac{p}{C}\right)^{1/k}} = -g dZ \text{ or } C^{1/k}\frac{dp}{\frac{1}{p^k}} = -g dZ$$

Integrating, we get $\int_{p_0}^{p} C^{1/k}\, p^{-1/k}\, dp = \int_{Z_0}^{Z} -g dZ$

or
$$C^{1/k}\left[\frac{p^{-1/k+1}}{-\frac{1}{k}+1}\right]_{p_0}^{p} = -g\,[Z]_{Z_0}^{Z}$$

or
$$\left[\frac{C^{1/k}\, p^{-1/k+1}}{-\frac{1}{k}+1}\right]_{p_0}^{p} = -g\,[Z]_{Z_0}^{Z} \qquad [C \text{ is a constant, can be taken inside}]$$

But from equation (i), $\quad C^{1/k} = \left(\frac{p}{\rho^k}\right)^{1/k} = \frac{p^{1/k}}{\rho}$

Substituting this value of $C^{1/k}$ above, we get

$$\left[\frac{p^{1/k}}{\rho}\frac{p^{-1/k+1}}{-\frac{1}{k}+1}\right]_{p_0}^{p} = -g[Z - Z_0]$$

or
$$\left[\frac{p^{\frac{1-1+k}{k}}}{\rho^{\frac{k-1}{k}}}\right]_{p_0}^{p} = -g[Z - Z_0] \text{ or } \left[\frac{k}{k-1}\frac{p}{\rho}\right]_{p_0}^{p} = -g[Z - Z_0]$$

or
$$\frac{k}{k-1}\left[\frac{p}{\rho} - \frac{p_0}{\rho_0}\right] = -g[Z - Z_0]$$

If datum line is taken at Z_0, where pressure, temperature and density are p_0, T_0 and ρ_0, then $Z_0 = 0$.

$$\therefore \quad \frac{k}{k-1}\left[\frac{p}{\rho} - \frac{p_0}{\rho_0}\right] = -gZ \quad \text{or} \quad \frac{p}{\rho} - \frac{p_0}{\rho_0} = -gZ\,\frac{(k-1)}{k}$$

or
$$\frac{p}{\rho} = \frac{p_0}{\rho_0} - gZ\,\frac{(k-1)}{k} = \frac{p_0}{\rho_0}\left[1 - \frac{k-1}{k}\,gZ\,\frac{\rho_0}{p_0}\right]$$

or $$\frac{p}{\rho} \times \frac{\rho_0}{p_0} = \left[1 + \frac{k-1}{k} gZ \frac{\rho_0}{p_0}\right] \quad ...(iii)$$

But from equation (*i*), $\frac{p}{\rho^k} = \frac{p_0}{\rho_0^k}$ or $\left(\frac{\rho_0}{\rho}\right)^k = \frac{p_0}{p}$ or $\frac{\rho_0}{\rho} = \left(\frac{p_0}{p}\right)^{1/k}$

Substituting the value of $\frac{\rho_0}{\rho}$ in equation (*iii*), we get

$$\frac{p}{p_0} \times \left(\frac{p_0}{p}\right)^{1/k} = \left[1 - \frac{k-1}{k} gZ \frac{\rho_0}{p_0}\right]$$

or $$\frac{p}{p_0} \times \left(\frac{p}{p_0}\right)^{-1/k} = \left[1 - \frac{k-1}{k} gZ \frac{\rho_0}{p_0}\right]$$

or $$\left(\frac{p}{p_0}\right)^{1-\frac{1}{k}} = \left(\frac{p}{p_0}\right)^{\frac{k-1}{k}} = \left[1 - \frac{k-1}{k} gZ \frac{\rho_0}{p_0}\right]$$

$\therefore$ $$\frac{p}{p_0} = \left[1 - \frac{k-1}{k} gZ \frac{\rho_0}{\rho_0}\right]^{\frac{k}{k-1}}$$

$\therefore$ Pressure at a height Z from ground level is given by

$$p = p_0 \left[1 - \frac{k-1}{k} gZ \frac{\rho_0}{p_0}\right]^{\frac{k}{k-1}} \quad ...(2.19)$$

In equation (2.19), p_0 = pressure at ground level, where $Z_0 = 0$

ρ_0 = density of air at ground level

Equation of state is $\frac{p_0}{\rho_0} = RT_0$ or $\frac{\rho_0}{p_0} = \frac{1}{RT_0}$

Substituting the values of $\frac{\rho_0}{p_0}$ in equation (2.19), we get

$$p = p_0 \left[1 - \frac{k-1}{k} \frac{gZ}{RT_0}\right]^{\frac{k}{k-1}} \quad ...(2.20)$$

2.8.3 Temperature at any Point in Compressible Fluid. For the adiabatic process, the temperature at any height in air is calculated as :

Equation of state at ground level and at a height Z from ground level is written as

$$\frac{p_0}{\rho_0} = RT_0 \text{ and } \frac{p}{\rho} = RT$$

Dividing these equations, we get

$$\left(\frac{p_0}{\rho_0}\right) \div \frac{p}{\rho} = \frac{RT_0}{RT} = \frac{T_0}{T} \quad \text{or} \quad \frac{p_0}{\rho_0} \times \frac{\rho}{p} = \frac{T_0}{T}$$

or $$\frac{T}{T_0} = \frac{\rho_0}{p_0} \times \frac{p}{\rho} = \frac{p}{p_0} \times \frac{\rho_0}{\rho} \quad ...(i)$$

But $\frac{p}{p_0}$ from equation (2.20) is given by

$$\frac{p}{p_0} = \left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]^{\frac{k}{k-1}}$$

Also for adiabatic process $\frac{p}{\rho^k} = \frac{p_0}{\rho_0^k}$ or $\left(\frac{\rho_0}{\rho}\right)^k = \frac{p_0}{p}$

or
$$\frac{\rho_0}{\rho} = \left(\frac{p_0}{p}\right)^{\frac{1}{k}} = \left(\frac{p}{p_0}\right)^{-\frac{1}{k}}$$

$$= \left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]^{\left(\frac{k}{k-1}\right)\times\left(-\frac{1}{k}\right)} = \left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]^{-\frac{1}{k-1}}$$

Substituting the values of $\frac{p}{p_0}$ and $\frac{\rho_0}{\rho}$ in equation (*i*), we get

$$\frac{T}{T_0} = \left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]^{\frac{k}{k-1}} \times \left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]^{-\frac{1}{k-1}}$$

$$= \left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]^{\frac{k}{k-1}-\frac{1}{k-1}} = \left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]$$

$\therefore$
$$T = T_0\left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right] \qquad ...(2.21)$$

2.8.4 Temperature Lapse-Rate (L). It is defined as the rate at which the temperature changes with elevation. To obtain an expression for the temperature lapse-rate, the temperature given by equation (2.21) is differentiated with respect to Z as

$$\frac{dT}{dZ} = \frac{d}{dZ}\left[T_0\left(1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right)\right]$$

where T_0, K, g and R are constant

$\therefore$
$$\frac{dT}{dZ} = -\frac{k-1}{k}\times\frac{g}{RT_0}\times T_0 = \frac{-g}{R}\left(\frac{k-1}{k}\right)$$

The temperature lapse-rate is denoted by L and hence

$$L = \frac{dT}{dZ} = \frac{-g}{R}\left(\frac{k-1}{k}\right) \qquad ...(2.22)$$

In equation (2.22), if (*i*) $k = 1$ which means isothermal process, $\frac{dT}{dZ} = 0$, which means temperature is constant with height.

(*ii*) If $k > 1$, the lapse-rate is negative which means temperature decreases with the increase in height.

In atmosphere, the value of k varies with height and hence the value of temperature lapse-rate also varies. From the sea-level upto an elevation of about 11000 m (or 11 km), the temperature of air decreases uniformly at the rate of 0.0065°C/m. from 11000 m to 32000 m, the temperature remains constant at – 56.5°C and hence in this range lapse-rate is zero. Temperature rises again after 32000 m in air.

Problem 2.22 (SI Units) *If the atmosphere pressure at sea level is 10.143 N/cm^2, determine the pressure at a height of 2500 m assuming the pressure variation follows (i) Hydrostatic law, and (ii) isothermal law. The density of air is given as 1.208 kg/m^3.*

Solution. Given :

Pressure at sea-level, $p_0 = 10.143 \text{ N/cm}^2 = 10.143 \times 10^4 \text{ N/m}^2$

Height, $Z = 2500 \text{ m}$

Density of air, $\rho_0 = 1.208 \text{ kg/m}^3$

(*i*) **Pressure by hydrostatic law.** For hydrostatic law, ρ is assumed constant and hence p is given by equation $\frac{dp}{dZ} = -\rho g$

Integrating, we get $\int_{p_0}^{p} dp = \int -\rho g \, dZ = -\rho g \int_{Z_0}^{Z} dZ$

or $p - p_0 = -\rho g \, [Z - Z_0]$

For datum line at sea-level, $Z_0 = 0$

$\therefore$ $p - p_0 = -\rho g Z$ or $p = p_0 - \rho g Z$

$= 10.143 \times 10^4 - 1.208 \times 9.81 \times 2500$ [$\because$ $\rho = \rho_0 = 1.208$]

$= 101430 - 29626 = 71804 \frac{\text{N}}{\text{m}^2}$ or $\frac{71804}{10^4} \text{ N/cm}^2$

$= \mathbf{7.18 \text{ N/cm}^2}$**. Ans.**

(*ii*) **Pressure by Isothermal Law.** Pressure at any height Z by isothermal law is given by equation (2.18) as

$$p = p_0 \, e^{-gZ/RT}$$

$$= 10.143 \times 10^4 \, e^{-\frac{gZ \times \rho_0}{\rho_0}} \qquad \left[\because \frac{p_0}{\rho_0} = RT \text{ and } \rho_0 g = w_0\right]$$

$$= 10.143 \times 10^4 \, e^{-\frac{Z\rho_0 \times g}{p_0}}$$

$$= 10.143 \times 10^4 \, e^{(-2500 \times 1.208 \times 9.81)/10.143 \times 10^4}$$

$$= 101430 \times e^{-.292} = 101430 \times \frac{1}{1.3391} = 75743 \text{ N/m}^2$$

$$= \frac{75743}{10^4} \text{ N/cm}^2 = \mathbf{7.574 \text{ N/cm}^2}\text{. Ans.}$$

Problem 2.23 *The barometric pressure at sea level is 760 mm of mercury while that on a mountain top is 735 mm. If the density of air is assumed constant at 1.2 kg/m^3, what is the elevation of the mountain top?*

Solution. Given :

Pressure* at sea, $p_0 = 760 \text{ mm of Hg}$

$= \frac{760}{1000} \times 13.6 \times 1000 \times 9.81 \text{ N/m}^2 = 101396 \text{ N/m}^2$

---

* Here pressure head (Z) is given as 760 mm of Hg. Hence ($p/\rho g$) = 760 mm of Hg. The density (ρ) for mercury = 13.6×1000 kg/m^3. Hence pressure (p) will be equal to $\rho \times g \times Z$ *i.e.,* $13.6 \times 1000 \times 9.81 \times \frac{760}{1000}$ N/m^2.

Pressure at mountain, p = 735 mm of Hg

$$= \frac{735}{1000} \times 13.6 \times 1000 \times 9.81 = 98060 \text{ N/m}^2$$

Density of air, $\rho = 1.2 \text{ kg/m}^3$

Let h = Height of the mountain from sea-level.

We know that as the elevation above the sea-level increases, the atmospheric pressure decreases. Here the density of air is given constant, hence the pressure at any height 'h' above the sea-level is given by the equation,

$$p = p_0 - \rho \times g \times h$$

or $$h = \frac{p_0 - p}{\rho \times g} = \frac{101396 - 98060}{1.2 \times 9.81} = \mathbf{283.33 \text{ m. Ans.}}$$

Problem 2.24 *Calculate the pressure at a height of 7500 m above sea level if the atmospheric pressure is 10.143 N/cm^2 and temperature is 15°C at the sea-level, assuming (i) air is incompressible, (ii) pressure variation follows isothermal law, and (iii) pressure variation follows adiabatic law. Take the density of air at the sea-level as equal to 1.285 kg/m^3. Neglect variation of g with altitude.*

Solution. Given :

Height above sea-level, Z = 7500 m

Pressure at sea-level, $p_0 = 10.143 \text{ N/cm}^2 = 10.143 \times 10^4 \text{ N/m}^2$

Temperature at sea-level, t_0 = 15°C

$\therefore$ $T_0 = 273 + 15 = 288°\text{K}$

Density of air, $\rho = \rho_0 = 1.285 \text{ kg/m}^3$

(*i*) *Pressure when air is incompressible :*

$$\frac{dp}{dZ} = -\rho g$$

$\therefore$ $$\int_{p_0}^{p} dp = -\int_{Z_0}^{Z} \rho g dz \quad \text{or} \quad p - p_0 = -\rho g[Z - Z_0]$$

or $$p = p_0 - \rho g Z \qquad \{\because \ Z_0 = \text{datum line} = 0\}$$

$$= 10.143 \times 10^4 - 1.285 \times 9.81 \times 7500$$

$$= 101430 - 94543 = 6887 \text{ N/m}^2 = \mathbf{0.688 \ \frac{N}{cm^2}. \ Ans.}$$

(*ii*) *Pressure variation follows isothermal law :*

Using equation (2.18), we have $p = p_0 e^{-gZ/RT}$

$$= p_0 \, e^{-gZ\rho_0/p_0} \qquad \left\{\because \ \frac{p_0}{\rho_0} = RT \ \therefore \ \frac{\rho_0}{p_0} = \frac{1}{RT}\right\}$$

$$= 101430 \, e^{-gZ\rho_0/p_0} = 101430 \, e^{-7500 \times 1.285 \times 9.81/101430}$$

$$= 101430 \, e^{-.9320} = 101430 \times .39376$$

$$= \mathbf{39939 \text{ N/m}^2 \text{ or } 3.993 \text{ N/cm}^2. \text{ Ans.}}$$

(*iii*) *Pressure variation follows adiabatic law :* [k = 1.4]

Using equation (2.19), we have $$p = p_0 \left[1 - \frac{k-1}{k} gZ \frac{\rho_0}{p_0}\right]^{k/(k-1)}$$, where $\rho_0 = 1.285 \text{ kg/m}^3$

$$\therefore \quad p = 101430\left[1-\frac{(1.4-1.0)}{1.4}\times 9.81\times\frac{(7500\times 1.285)}{101430}\right]^{\frac{1.4}{1.4-1.0}}$$

$$= 101430\ [1 - .2662]^{1.4/.4} = 101430 \times (.7337)^{3.5}$$

$$= \mathbf{34310\ N/m^2\ or\ 3.431\frac{N}{cm^2}.\ Ans.}$$

Problem 2.25 *Calculate the pressure and density of air at a height of 4000 m from sea-level where pressure and temperature of the air are 10.143 N/cm^2 and 15°C respectively. The temperature lapse rate is given as 0.0065°C/m. Take density of air at sea-level equal to 1.285 kg/m^3.*

Solution. Given :

Height, $Z = 4000$ m

Pressure at sea-level, $p_0 = 10.143\ \text{N/cm}^2 = 10.143 \times 10^4 = 101430\ \frac{\text{N}}{\text{m}^2}$

Temperature at sea-level, $t_0 = 15°\text{C}$

$\therefore \quad T_0 = 273 + 15 = 288°\text{K}$

Temperature lapse-rate, $L = \frac{dT}{dZ} = -0.0065°\text{K/m}$

$\rho_0 = 1.285\ \text{kg/m}^3$

Using equation (2.22), we have $L = \frac{dT}{dZ} = -\frac{g}{R}\left(\frac{k-1}{k}\right)$

or $-0.0065 = -\frac{9.81}{R}\left(\frac{k-1}{k}\right)$, where $R = \frac{p_0}{\rho_0 T_0} = \frac{101430}{1.285\times 288} = 274.09$

$\therefore \quad -0.0065 = \frac{-9.81}{274.09}\times\left(\frac{k-1}{k}\right)$

$\therefore \quad \frac{k-1}{k} = \frac{0.0065\times 274.09}{9.81} = 0.1815$

$\therefore \quad k[1 - .1815] = 1$

$\therefore \quad k = \frac{1}{1-.1815} = \frac{1}{.8184} = 1.222$

This means that the value of power index $k = 1.222$.

(*i*) **Pressure** at 4000 m height is given by equation (2.19) as

$$p = p_0\left[1-\frac{k-1}{k}gZ\frac{\rho_0}{p_0}\right]^{\frac{k}{k-1}},\ \text{where } k = 1.222 \text{ and } \rho_0 = 1.285$$

$$\therefore \quad p = 101430\left[1-\left(\frac{1.222-1.0}{1.222}\right)\times 9.81\times\frac{4000\times 1.285}{101430}\right]^{\frac{1.222}{1.222-1.0}}$$

$$= 101430\ [1 - 0.09]^{5.50} = 101430 \times .595$$

$$= 60350\ \text{N/m}^2 = \mathbf{6.035\ \frac{N}{cm^2}.\ Ans.}$$

(*ii*) **Density.** Using equation of state, we get

$$\frac{p}{\rho} = RT$$

where p = Pressure at 4000 m height
ρ = Density at 4000 m height
T = Temperature at 4000 m height

Now T is calculated from temperature lapse-rate as

$$t \text{ at } 4000 \text{ m} = t_0 + \frac{dT}{dZ} \times 4000 = 15 - .0065 \times 4000 = 15 - 26 = -11°\text{C}$$

$\therefore$ $$T = 273 + t = 273 - 11 = 262°\text{K}$$

$\therefore$ Density is given by $$\rho = \frac{p}{RT} = \frac{60350}{274.09 \times 262} \text{ kg/m}^3 = \mathbf{0.84 \text{ kg/m}^3}. \textbf{ Ans.}$$

Problem 2.26 *An aeroplane is flying at an altitude of 5000 m. Calculate the pressure around the aeroplane, given the lapse-rate in the atmosphere as 0.0065°K/m. Neglect variation of g with altitude. Take pressure and temperature at ground level as 10.143 N/cm^2 and 15°C and density of air as 1.285 kg/cm^3.*

Solution. Given :

Height, $Z = 5000$ m

Lapse-rate, $L = \frac{dT}{dZ} = -.0065°\text{K/m}$

Pressure at ground level, $p_0 = 10.143 \times 10^4 \text{ N/m}^2$

$t_0 = 15°\text{C}$

$\therefore$ $T_0 = 273 + 15 = 288°\text{K}$

Density, $\rho_0 = 1.285 \text{ kg/m}^3$

$\therefore$ Temperature at 5000 m height $= T_0 + \frac{dT}{dZ} \times \text{Height} = 288 - .0065 \times 5000$

$= 288 - 32.5 = 255.5°\text{K}$

First find the value of power index k as

From equation (2.22), we have $$L = \frac{dT}{dZ} = -\frac{g}{R}\left(\frac{k-1}{k}\right)$$

or $$-.0065 = -\frac{9.81}{R}\left(\frac{k-1}{k}\right)$$

where $$R = \frac{p_0}{\rho_0 T_0} = \frac{101430}{1.285 \times 288} = 274.09$$

$\therefore$ $$-.0065 = -\frac{9.81}{274.09}\left(\frac{k-1}{k}\right)$$

$\therefore$ $$k = 1.222$$

The pressure is given by equation (2.19) as

$$p = p_0 \left[1 - \frac{k-1}{k} gZ \frac{\rho_0}{p_0}\right]^{\left(\frac{k}{k-1}\right)}$$

$$= 101430 \left[1 - \left(\frac{1.222 - 1.0}{1.222}\right) \times 9.81 \times \frac{5000 \times 1.285}{101430}\right]^{\frac{1.222}{1.222 - 1.0}}$$

$$= 101430 \left[1 - \frac{.222}{1.222} \times 9.81 \times \frac{5000 \times 1.285}{101430}\right]^{\frac{1.222}{.222}}$$

$= 101430\ [1 - 0.11288]^{5.50} = 101430 \times 0.5175 = 52490\ \text{N/m}^3$

$= 5.249\ \text{N/cm}^2$. Ans.

HIGHLIGHTS

1. The pressure at any point in a fluid is defined as the force per unit area.
2. The Pascal's law states that intensity of pressure for a fluid at rest is equal in all directions.
3. Pressure variation at a point in a fluid at rest is given by the hydrostatic law which states that the rate of increase of pressure in the vertically downward direction is equal to the specific weight of the fluid, $\frac{dp}{dZ} = w = \rho \times g$.
4. The pressure at any point in a incompressible fluid (liquid) is equal to the product of density of fluid at that point, acceleration due to gravity and vertical height from free surface of fluid, $p = \rho \times g \times Z$.
5. Absolute pressure is the pressure in which absolute vacuum pressure is taken as datum while gauge pressure is the pressure in which the atmospheric pressure is taken as datum,
$$p_{\text{abs.}} = p_{\text{atm}} + p_{\text{gauge.}}$$
6. Manometer is a device used for measuring pressure at a point in a fluid.
7. Manometers are classified as (*a*) Simple manometers and (*b*) Differential manometers.
8. Simple manometers are used for measuring pressure at a point while differential manometers are used for measuring the difference of pressures between the two points in a pipe, or two different pipes.
9. A single column manometer (or micrometer) is used for measuring small pressures, where accuracy is required.
10. The pressure at a point in static compressible fluid is obtained by combining two equations, *i.e.*, equation of state for a gas and equation given by hydrostatic law.
11. The pressure at a height Z in a static compressible fluid (gas) under going isothermal compression $\left(\frac{p}{\rho} = \text{const.}\right)$
$$p = p_0\, e^{-gZ/RT}$$
where p_0 = Absolute pressure at sea-level or at ground level
Z = Height from sea or ground level
R = Gas constant
T = Absolute temperature.
12. The pressure and temperature at a height Z in a static compressible fluid (gas) undergoing adiabatic compression (p/ρ^k = const.)
$$p = p_0 \left[1 - \frac{k-1}{k} gZ \frac{\rho_0}{p_0}\right]^{\frac{k}{k-1}} = p_0 \left[1 - \frac{k-1}{k} \frac{gZ}{RT_0}\right]^{\frac{k}{k-1}}$$

and temperature, $$T = T_0\left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]$$

where p_0, T_0 are pressure and temperature at sea-level $k = 1.4$ for air.

13. The rate at which the temperature changes with elevation is known as Temperature Lapse-Rate. It is given by

$$L = \frac{-g}{R}\left(\frac{k-1}{k}\right)$$

if (*i*) $k = 1$, temperature is zero.

(*ii*) $k > 1$, temperature decreases with the increase of height.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define pressure. Obtain an expression for the pressure intensity at a point in a fluid.
2. State and prove the Pascal's law.
3. What do you understand by Hydrostatic Law ?
4. Differentiate between : (*i*) Absolute and gauge pressure, (*ii*) Simple manometer and differential manometer, and (*iii*) Piezometer and pressure gauges.
5. What do you mean by vacuum pressure ?
6. What is a manometer ? How are they classified ?
7. What do you mean by single column manometers ? How are they used for the measurement of pressure ?
8. What is the difference between U-tube differential manometers and inverted U-tube differential manometers ? Where are they used ?
9. Distinguish between manometers and mechanical gauges. What are the different types of mechanical pressure gauges ?
10. Derive an expression for the pressure at a height Z from sea-level for a static air when the compression of the air is assumed isothermal. The pressure and temperature at sea-levels are p_0 and T_0 respectively.
11. Prove that the pressure and temperature for an adiabatic process at a height Z from sea-level for a static air are :

$$p_0 = p_0\left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]^{\frac{k}{k-1}} \text{ and } T = T_0\left[1 - \frac{k-1}{k}\frac{gZ}{RT_0}\right]$$

where p_0 and T_0 are the pressure and temperature at sea-level.
12. What do you understand by the term, 'Temperature Lapse-Rate'? Obtain an expression for the temperature Lapse-Rate.
13. What is hydrostatic pressure distribution? Give one example where pressure distribution is non-hydrostatic.
14. Explain briefly the working principle of Bourdon Pressure Gauge with a neat sketch.

(J.N.T.U., Hyderabad, S 2002)

(B) NUMERICAL PROBLEMS

1. A hydraulic press has a ram of 30 cm diameter and a plunger of 5 cm diameter. Find the weight lifted by the hydraulic press when the force applied at the plunger is 400 N. **[Ans.** 14.4 kN]
2. A hydraulic press has a ram of 20 cm diameter and a plunger of 4 cm diameter. It is used for lifting a weight of 20 kN. Find the force required at the plunger. **[Ans.** 800 N]

3. Calculate the pressure due to a column of 0.4 m of (*a*) water, (*b*) an oil of sp. gr. 0.9, and (*c*) mercury of sp. gr. 13.6. Take density of water, $\rho = 1000 \frac{kg}{m^3}$. [**Ans.** (*a*) 0.3924 N/cm^2, (*b*) 0.353 N/cm^2, (*c*) 5.33 N/cm^2]
4. The pressure intensity at a point in a fluid is given 4.9 N/cm^2. Find the corresponding height of fluid when it is : (*a*) water, and (*b*) an oil of sp. gr. 0.8. [**Ans.** (*a*) 5 m of water, (*b*) 6.25 m of oil]
5. An oil of sp. gr. 0.8 is contained in a vessel. At a point the height of oil is 20 m. Find the corresponding height of water at that point. [**Ans.** 16 m]
6. An open tank contains water upto a depth of 1.5 m and above it an oil of sp. gr. 0.8 for a depth of 2 m. Find the pressure intensity : (*i*) at the interface of the two liquids, and (*ii*) at the bottom of the tank. [**Ans.** (*i*) 1.57 N/cm^2, (*ii*) 3.04 N/cm^2]
7. The diameters of a small piston and a large piston of a hydraulic jack are 2 cm and 10 cm respectively. A force of 60 N is applied on the small piston. Find the load lifted by the large piston, when : (*a*) the pistons are at the same level, and (*b*) small piston is 20 cm above the large piston. The density of the liquid in the jack is given as 1000 $\frac{kg}{m^2}$. [**Ans.** (*a*) 1500 N, (*b*) 1520.5 N]
8. Determine the gauge and absolute pressure at a point which is 2.0 m below the free surface of water. Take atmospheric pressure as 10.1043 N/cm^2. [**Ans.** 1.962 N/cm^2 (gauge), 12.066 N/cm^2 (abs.)]
9. A simple manometer is used to measure the pressure of oil (sp. gr. = 0.8) flowing in a pipe line. Its right limb is open to the atmosphere and left limb is connected to the pipe. The centre of the pipe is 9 cm below the level of mercury (sp. gr. 13.6) in the right limb. If the difference of mercury level in the two limbs is 15 cm, determine the absolute pressure of the oil in the pipe in N/cm^2. [**Ans.** 12.058 N/cm^2]
10. A simple manometer (U-tube) containing mercury is connected to a pipe in which an oil of sp. gr. 0.8 is flowing. The pressure in the pipe is vacuum. The other end of the manometer is open to the atmosphere. Find the vacuum, pressure in pipe, if the difference of mercury level in the two limbs is 20 cm and height of oil in the left limb from the centre of the pipe is 15 cm below. [**Ans.** – 27.86 N/cm^2]
11. A single column vertical manometer (*i.e.,* micrometer) is connected to a pipe containing oil of sp. gr. 0.9. The area of the reservoir is 80 times the area of the manometer tube. The reservoir contains mercury of sp. gr. 13.6. The level of mercury in the reservoir is at a height of 30 cm below the centre of the pipe and difference of mercury levels in the reservoir and right limb is 50 cm. Find the pressure in the pipe. [**Ans.** 6.474 N/cm^2]
12. A pipe contains an oil of sp. gr. 0.8. A differential manometer connected at the two points A and B of the pipe shows a difference in mercury level as 20 cm. Find the difference of pressure at the two points. [**Ans.** 25113.6 N/m^2]
13. A U-tube differential manometer connects two pressure pipes *A* and *B*. Pipe *A* contains carbon tetrachloride having a specific gravity 1.594 under a pressure of 11.772 N/cm^2 and pipe *B* contains oil of sp. gr. 0.8 under a pressure of 11.772 N/cm^2. The pipe *A* lies 2.5 m above pipe *B*. Find the difference of pressure measured by mercury as fluid filling U-tube. [**Ans.** 31.36 cm of mercury]
14. A differential manometer is connected at the two points *A* and *B* as shown in Fig. 2.25. At *B* air pressure is 7.848 N/cm^2 (abs.), find the absolute pressure at *A*. [**Ans.** 6.91 N/cm^2]

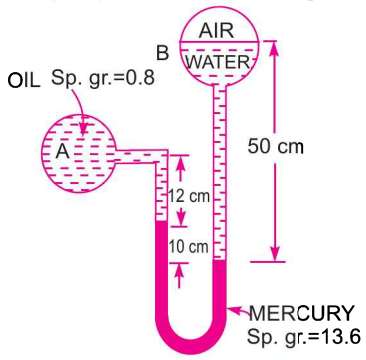


Fig. 2.25

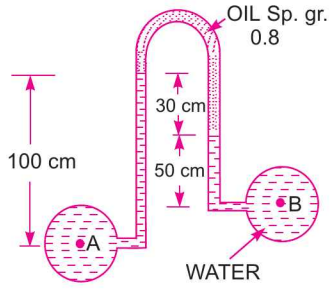


Fig. 2.26

15. An inverted differential manometer containing an oil of sp. gr. 0.9 is connected to find the difference of pressures at two points of a pipe containing water. If the manometer reading is 40 cm, find the difference of pressures. [**Ans.** 392.4 N/m^2]

16. In above Fig. 2.26 shows an inverted differential manometer connected to two pipes *A* and *B* containing water. The fluid in manometer is oil of sp. gr. 0.8. For the manometer readings shown in the figure, find the difference of pressure head between *A* and *B*. [**Ans.** 0.26 m of water]

17. If the atmospheric pressure at sea-level is 10.143 N/cm^2, determine the pressure at a height of 2000 m assuming that the pressure variation follows : (*i*) Hydrostatic law, and (*ii*) Isothermal law. The density of air is given as 1.208 kg/m^3. [**Ans.** (*i*) 7.77 N/cm^2, (*ii*) 8.03 N/cm^3]

18. Calculate the pressure at a height of 8000 m above sea-level if the atmospheric pressure is 101.3 kN/m^2 and temperature is 15°C at the sea-level assuming (*i*) air is incompressible, (*ii*) pressure variation follows adiabatic law, and (*iii*) pressure variation follows isothermal law. Take the density of air at the sea-level as equal to 1.285 kg/m^3. Neglect variation of *g* with altitude.

[**Ans.** (*i*) 607.5 N/m^2, (*ii*) 31.5 kN/m^2 (*iii*) 37.45 kN/m^2]

19. Calculate the pressure and density of air at a height of 3000 m above sea-level where pressure and temperature of the air are 10.143 N/cm^2 and 15°C respectively. The temperature lapse-rate is given as 0.0065° K/m. Take density of air at sea-level equal to 1.285 kg/m^3. [**Ans.** 6.896 N/cm^2, 0.937 kg/m^3]

20. An aeroplane is flying at an altitude of 4000 m. Calculate the pressure around the aeroplane, given the lapse-rate in the atmosphere as 0.0065°K/m. Neglect variation of *g* with altitude. Take pressure and temperature at ground level as 10.143 N/cm^2 and 15°C respectively. The density of air at ground level is given as 1.285 kg/m^3. [**Ans.** 6.038 N/cm^2]

21. The atmospheric pressure at the sea-level is 101.3 kN/m^2 and the temperature is 15°C. Calculate the pressure 8000 m above sea-level, assuming (*i*) air is incompressible, (*ii*) isothermal variation of pressure and density, and (*iii*) adiabatic variation of pressure and density. Assume density of air at sea-level as 1.285 kg/m^3. Neglect variation of '*g*' with altitude.

[**Ans.** (*i*) 501.3 N/m^2, (*ii*) 37.45 kN/m^2, (*iii*) 31.5 kN/m^2]

22. An oil of sp. gr. is 0.8 under a pressure of 137.2 kN/m^2

(*i*) What is the pressure head expressed in metre of water ?

(*ii*) What is the pressure head expressed in metre of oil ? [**Ans.** (*i*) 14 m, (*ii*) 17.5 m]

23. The atmospheric pressure at the sea-level is 101.3 kN/m^2 and temperature is 15°C. Calculate the pressure 8000 m above sea-level, assuming : (*i*) isothermal variation of pressure and density, and (*ii*) adiabatic variation of pressure and density. Assume density of air at sea-level as 1.285 kg/m^3. Neglect variation of '*g*' with altitude.

Derive the formula that you may use. [**Ans.** (*i*) 37.45 kN/m^2, (*ii*) 31.5 kN/m^2]

24. What are the gauge pressure and absolute pressure at a point 4 m below the free surface of a liquid of specific gravity 1.53, if atmospheric pressure is equivalent to 750 mm of mercury.

[**Ans.** 60037 N/m^2 and 160099 N/m^2]

25. Find the gauge pressure and absolute pressure in N/m^2 at a point 4 m below the free surface of a liquid of sp. gr. 1.2, if the atmospheric pressure is equivalent to 750 mm of mercury.

[**Ans.** 47088 N/m^2 ; 147150 N/m^2]

26. A tank contains a liquid of specific gravity 0.8. Find the absolute pressure and gauge pressure at a point, which is 2 m below the free surface of the liquid. The atmospheric pressure head is equivalent to 760 mm of mercury. [**Ans.** 117092 N/m^2 ; 15696 N/m^2]

3 CHAPTER

HYDROSTATIC FORCES ON SURFACES

▶ 3.1 INTRODUCTION

This chapter deals with the fluids (*i.e.,* liquids and gases) at rest. This means that there will be no relative motion between adjacent or neighbouring fluid layers. The velocity gradient, which is equal to the change of velocity between two adjacent fluid layers divided by the distance between the layers, will be zero or $\frac{du}{dy} = 0$. The shear stress which is equal to $\mu \frac{\partial u}{\partial y}$ will also be zero. Then the forces acting on the fluid particles will be :

1. due to pressure of fluid normal to the surface,
2. due to gravity (or self-weight of fluid particles).

▶ 3.2 TOTAL PRESSURE AND CENTRE OF PRESSURE

Total pressure is defined as the force exerted by a static fluid on a surface either plane or curved when the fluid comes in contact with the surfaces. This force always acts normal to the surface.

Centre of pressure is defined as the point of application of the total pressure on the surface. There are four cases of submerged surfaces on which the total pressure force and centre of pressure is to be determined. The submerged surfaces may be :

1. Vertical plane surface,
2. Horizontal plane surface,
3. Inclined plane surface, and
4. Curved surface.

▶ 3.3 VERTICAL PLANE SURFACE SUBMERGED IN LIQUID

Consider a plane vertical surface of arbitrary shape immersed in a liquid as shown in Fig. 3.1.

Let A = Total area of the surface

$\bar{h}$ = Distance of C.G. of the area from free surface of liquid

G = Centre of gravity of plane surface

P = Centre of pressure

h^* = Distance of centre of pressure from free surface of liquid.

(*a*) **Total Pressure (F)**. The total pressure on the surface may be determined by dividing the entire surface into a number of small parallel strips. The force on small strip is then calculated and the total pressure force on the whole area is calculated by integrating the force on small strip.

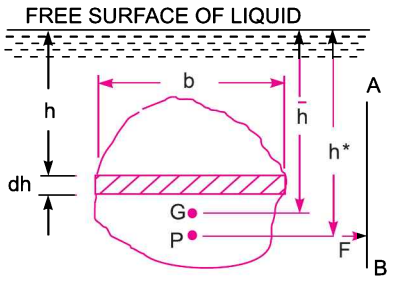


Fig. 3.1

Consider a strip of thickness dh and width b at a depth of h from free surface of liquid as shown in Fig. 3.1

Pressure intensity on the strip, $p = \rho g h$

(See equation 2.5)

Area of the strip, $dA = b \times dh$

Total pressure force on strip, $dF = p \times \text{Area}$

$= \rho g h \times b \times dh$

$\therefore$ Total pressure force on the whole surface,

$$F = \int dF = \int \rho g h \times b \times dh = \rho g \int b \times h \times dh$$

But $$\int b \times h \times dh = \int h \times dA$$

= Moment of surface area about the free surface of liquid

= Area of surface × Distance of C.G. from free surface

$= A \times \bar{h}$

$\therefore$ $$F = \rho g A \bar{h} \qquad ...(3.1)$$

For water the value of $\rho = 1000$ kg/m^3 and $g = 9.81$ m/s^2. The force will be in Newton.

(*b*) **Centre of Pressure (h*)**. Centre of pressure is calculated by using the "Principle of Moments", which states that the moment of the resultant force about an axis is equal to the sum of moments of the components about the same axis.

The resultant force F is acting at P, at a distance h^* from free surface of the liquid as shown in Fig. 3.1. Hence moment of the force F about free surface of the liquid = $F \times h^*$ ...(3.2)

Moment of force dF, acting on a strip about free surface of liquid

$= dF \times h$ $\{\because \quad dF = \rho g h \times b \times dh\}$

$= \rho g h \times b \times dh \times h$

Sum of moments of all such forces about free surface of liquid

$$= \int \rho g h \times b \times dh \times h = \rho g \int b \times h \times h dh$$

$$= \rho g \int b h^2 \, dh = \rho g \int h^2 \, dA \qquad (\because \quad b dh = dA)$$

But $$\int h^2 dA = \int b h^2 dh$$

= Moment of Inertia of the surface about free surface of liquid

$= I_0$

$\therefore$ Sum of moments about free surface

$$= \rho g I_0 \qquad ...(3.3)$$

Equating (3.2) and (3.3), we get

$$F \times h^* = \rho g I_0$$

But $$F = \rho g A \bar{h}$$

$\therefore$ $$\rho g A \bar{h} \times h^* = \rho g I_0$$

or $$h^* = \frac{\rho g I_0}{\rho g A \bar{h}} = \frac{I_0}{A \bar{h}} \quad \ldots(3.4)$$

By the theorem of parallel axis, we have

$$I_0 = I_G + A \times \overline{h^2}$$

where I_G = Moment of Inertia of area about an axis passing through the C.G. of the area and parallel to the free surface of liquid.

Substituting I_0 in equation (3.4), we get

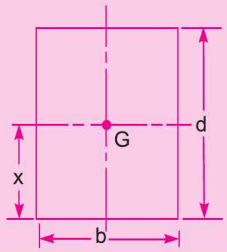
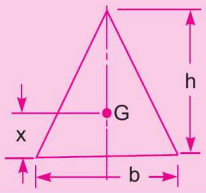
$$h^* = \frac{I_G + A\overline{h^2}}{A \bar{h}} = \frac{I_G}{A \bar{h}} + \bar{h} \quad \ldots(3.5)$$

In equation (3.5), $\bar{h}$ is the distance of C.G. of the area of the vertical surface from free surface of the liquid. Hence from equation (3.5), it is clear that :

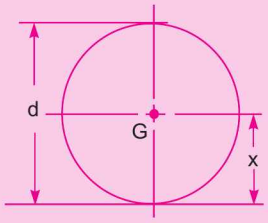
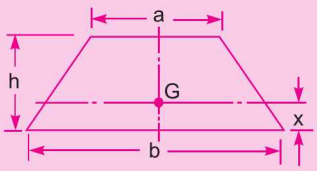
(*i*) Centre of pressure (*i.e.*, h^*) lies below the centre of gravity of the vertical surface.

(*ii*) The distance of centre of pressure from free surface of liquid is independent of the density of the liquid.

Table 3.1 The moments of inertia and other geometric properties of some important plane surfaces

Plane surface	*C.G. from the base*	*Area*	*Moment of inertia about an axis passing through C.G. and parallel to base (I_G)*	*Moment of inertia about base (I_0)*
1. Rectangle 	$x = \frac{d}{2}$	bd	$\frac{bd^3}{12}$	$\frac{bd^3}{3}$
2. Triangle 	$x = \frac{h}{3}$	$\frac{bh}{2}$	$\frac{bh^3}{36}$	$\frac{bh^3}{12}$

Contd...

Plane surface	*C.G. from the base*	*Area*	*Moment of inertia about an axis passing through C.G. and parallel to base (I_G)*	*Moment of inertia about base (I_0)*
3. Circle 	$x = \frac{d}{2}$	$\frac{\pi d^2}{4}$	$\frac{\pi d^4}{64}$	—
4. Trapezium 	$x = \left(\frac{2a+b}{a+b}\right)\frac{h}{3}$	$\frac{(a+b)}{2} \times h$	$\left(\frac{a^2+4ab+b^2}{36(a+b)}\right) \times h^3$	—

Problem 3.1 *A rectangular plane surface is 2 m wide and 3 m deep. It lies in vertical plane in water. Determine the total pressure and position of centre of pressure on the plane surface when its upper edge is horizontal and (a) coincides with water surface, (b) 2.5 m below the free water surface.*

Solution. Given :

Width of plane surface, $b = 2$ m

Depth of plane surface, $d = 3$ m

(*a*) **Upper edge coincides with water surface (Fig. 3.2).** Total pressure is given by equation (3.1) as

$$F = \rho g A \bar{h}$$

where $\rho = 1000$ kg/m^3, $g = 9.81$ m/s^2

$$A = 3 \times 2 = 6 \text{ m}^2, \ \bar{h} = \frac{1}{2}(3) = 1.5 \text{ m}.$$

$$\therefore \quad F = 1000 \times 9.81 \times 6 \times 1.5$$

$$= \mathbf{88290 \text{ N. Ans.}}$$

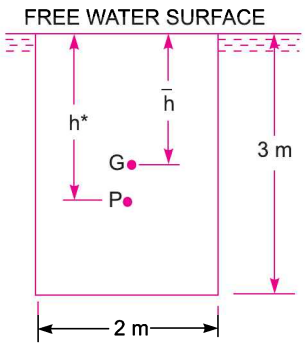


Fig. 3.2

Depth of centre of pressure is given by equation (3.5) as

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h}$$

where I_G = M.O.I. about C.G. of the area of surface

$$= \frac{bd^3}{12} = \frac{2 \times 3^3}{12} = 4.5 \text{ m}^4$$

$$\therefore \qquad h^* = \frac{4.5}{6 \times 1.5} + 1.5 = 0.5 + 1.5 = \mathbf{2.0\ m.\ Ans.}$$

(*b*) **Upper edge is 2.5 m below water surface (Fig. 3.3).** Total pressure (F) is given by (3.1)

$$F = \rho g A \bar{h}$$

where $\bar{h}$ = Distance of C.G. from free surface of water

$$= 2.5 + \frac{3}{2} = 4.0 \text{ m}$$

$$\therefore \qquad F = 1000 \times 9.81 \times 6 \times 4.0$$

$$= \mathbf{235440\ N.\ Ans.}$$

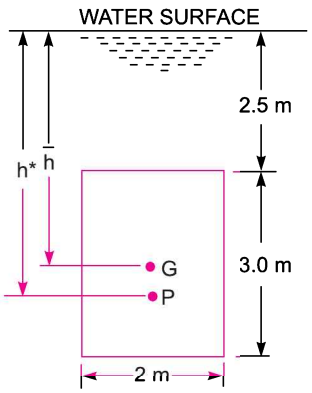


Fig. 3.3

Centre of pressure is given by $h^* = \frac{I_G}{A\bar{h}} + \bar{h}$

where $I_G = 4.5$, $A = 6.0$, $\bar{h} = 4.0$

$$\therefore \qquad h^* = \frac{4.5}{6.0 \times 4.0} + 4.0$$

$$= 0.1875 + 4.0 = 4.1875 = \mathbf{4.1875\ m.\ Ans.}$$

Problem 3.2 *Determine the total pressure on a circular plate of diameter 1.5 m which is placed vertically in water in such a way that the centre of the plate is 3 m below the free surface of water. Find the position of centre of pressure also.*

Solution. Given : Dia. of plate, $d = 1.5$ m

$$\therefore \text{ Area,} \qquad A = \frac{\pi}{4}(1.5)^2 = 1.767 \text{ m}^2$$

$$\bar{h} = 3.0 \text{ m}$$

Total pressure is given by equation (3.1),

$$F = \rho g A \bar{h}$$

$$= 1000 \times 9.81 \times 1.767 \times 3.0 \text{ N}$$

$$= \mathbf{52002.81\ N.\ Ans.}$$

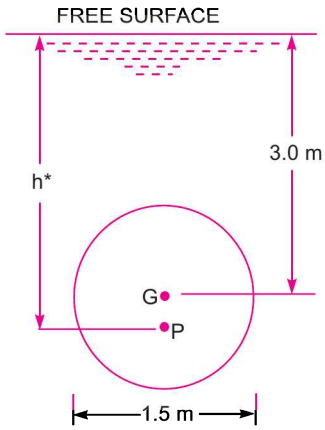


Fig. 3.4

Position of centre of pressure (h^*) is given by equation (3.5),

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h}$$

where $I_G = \frac{\pi d^4}{64} = \frac{\pi \times 1.5^4}{64} = 0.2485 \text{ m}^4$

$$\therefore \qquad h^* = \frac{0.2485}{1.767 \times 3.0} + 3.0 = 0.0468 + 3.0$$

$$= \mathbf{3.0468\ m.\ Ans.}$$

Problem 3.3 *A rectangular sluice gate is situated on the vertical wall of a lock. The vertical side of the sluice is 'd' metres in length and depth of centroid of the area is 'p' m below the water surface. Prove that the depth of pressure is equal to* $\left(p + \frac{d^2}{12p}\right)$.

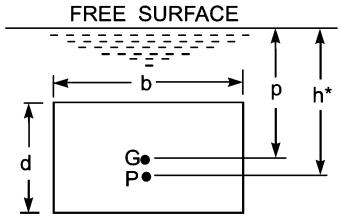


Fig. 3.5

Solution. Given :

Depth of vertical gate $= d$ m

Let the width of gate $= b$ m

$\therefore$ Area, $A = b \times d$ m^2

Depth of C.G. from free surface

$$\bar{h} = p \text{ m.}$$

Let h^* is the depth of centre of pressure from free surface, which is given by equation (3.5) as

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h}, \text{ where } I_G = \frac{bd^3}{12}$$

$\therefore$ $$h^* = \left(\frac{bd^3}{12} \Big/ b \times d \times p\right) + p = \mathbf{\frac{d^2}{12p} + p} \quad \text{or} \quad \mathbf{p + \frac{d^2}{12p}}\text{ . Ans.}$$

Problem 3.4 *A circular opening, 3 m diameter, in a vertical side of a tank is closed by a disc of 3 m diameter which can rotate about a horizontal diameter. Calculate :*

(i) the force on the disc, and

(ii) the torque required to maintain the disc in equilibrium in the vertical position when the head of water above the horizontal diameter is 4 m.

Solution. Given :

Dia. of opening, $d = 3$ m

$\therefore$ Area, $A = \frac{\pi}{4} \times 3^2 = 7.0685$ m^2

Depth of C.G., $\bar{h} = 4$ m

(*i*) Force on the disc is given by equation (3.1) as

$$F = \rho g A \bar{h} = 1000 \times 9.81 \times 7.0685 \times 4.0$$

$$= \mathbf{277368 \text{ N} = 277.368 \text{ kN. Ans.}}$$

(*ii*) To find the torque required to maintain the disc in equilibrium, first calculate the point of application of force acting on the disc, *i.e.*, centre of pressure of the force F. The depth of centre of pressure (h^*) is given by equation (3.5) as

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h} = \frac{\frac{\pi}{64} d^4}{\frac{\pi}{4} d^2 \times 4.0} + 4.0 \qquad \left\{\because I_G = \frac{\pi}{64} d^4\right\}$$

$$= \frac{d^2}{16 \times 4.0} + 4.0 = \frac{3^2}{16 \times 4.0} + 4.0 = 0.14 + 4.0 = 4.14 \text{ m}$$

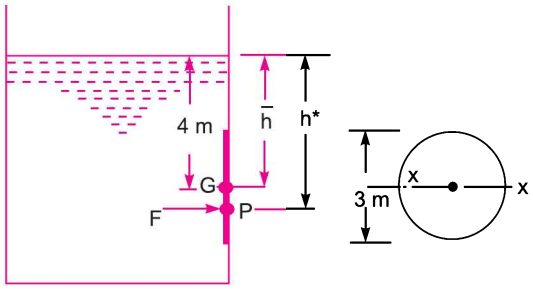


Fig. 3.6

The force F is acting at a distance of 4.14 m from free surface. Moment of this force about horizontal diameter X-X

$$= F \times (h^* - \bar{h}) = 277368\ (4.14 - 4.0) = \textbf{38831 Nm. Ans.}$$

Hence a torque of 38831 Nm must be applied on the disc in the clockwise direction.

Problem 3.5 *A pipe line which is 4 m in diameter contains a gate valve. The pressure at the centre of the pipe is 19.6 N/cm^2. If the pipe is filled with oil of sp. gr. 0.87, find the force exerted by the oil upon the gate and position of centre of pressure.*

Solution. Given :

Dia. of pipe, $d = 4$ m

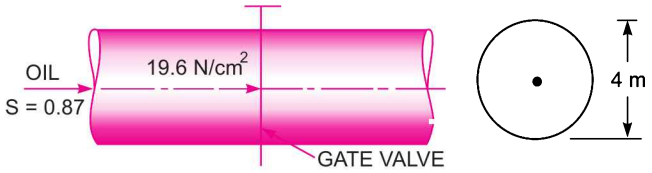


Fig. 3.7

$\therefore$ Area, $A = \dfrac{\pi}{4} \times 4^2 = 4\pi \text{ m}^2$

Sp. gr. of oil, $S = 0.87$

$\therefore$ Density of oil, $\rho_0 = 0.87 \times 1000 = 870 \text{ kg/m}^3$

$\therefore$ Weight density of oil, $w_0 = \rho_0 \times g = 870 \times 9.81 \text{ N/m}^3$

Pressure at the centre of pipe, $p = 19.6 \text{ N/cm}^2 = 19.6 \times 10^4 \text{ N/m}^2$

$\therefore$ Pressure head at the centre $= \dfrac{p}{w_0} = \dfrac{19.6 \times 10^4}{870 \times 9.81} = 22.988$ m

$\therefore$ The height of equivalent free oil surface from the centre of pipe = 22.988 m.

The depth of C.G. of the gate valve from free oil surface $\bar{h} = 22.988$ m.

(*i*) Now the force exerted by the oil on the gate is given by

$$F = \rho g A \bar{h}$$

where ρ = density of oil = 870 kg/m^3

$$F = 870 \times 9.81 \times 4\pi \times 22.988 = \textbf{2465500 N = 2.465 MN. Ans.}$$

(*ii*) Position of centre of pressure (h^*) is given by (3.5) as

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h} = \frac{\frac{\pi}{64} d^4}{\frac{\pi}{4} d^2 \times \bar{h}} + \bar{h} = \frac{d^2}{16\,\bar{h}} + \bar{h} = \frac{4^2}{16 \times 22.988} + 22.988$$

$$= 0.043 + 22.988 = \mathbf{23.031\ m.\ Ans.}$$

Or centre of pressure is below the centre of the pipe by a distance of 0.043 m. **Ans.**

Problem 3.6 *Determine the total pressure and centre of pressure on an isosceles triangular plate of base 4 m and altitude 4 m when it is immersed vertically in an oil of sp. gr. 0.9. The base of the plate coincides with the free surface of oil.*

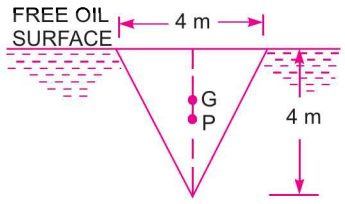


Fig. 3.8

Solution. Given :

Base of plate, $b = 4$ m

Height of plate, $h = 4$ m

$\therefore$ Area, $A = \frac{b \times h}{2} = \frac{4 \times 4}{2} = 8.0\ \text{m}^2$

Sp. gr. of oil, $S = 0.9$

$\therefore$ Density of oil, $\rho = 900\ \text{kg/m}^3$.

The distance of C.G. from free surface of oil,

$$\bar{h} = \frac{1}{3} \times h = \frac{1}{3} \times 4 = 1.33\ \text{m.}$$

Total pressure (F) is given by $F = \rho g A \bar{h}$

$$= 900 \times 9.81 \times 8.0 \times 1.33\ \text{N} = \mathbf{9597.6.\ N.\ Ans.}$$

Centre of pressure (h^*) from free surface of oil is given by

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h}$$

where I_G = M.O.I. of triangular section about its C.G.

$$= \frac{bh^3}{36} = \frac{4 \times 4^3}{36} = 7.11\ \text{m}^4$$

$\therefore$ $$h^* = \frac{7.11}{8.0 \times 1.33} + 1.33 = 0.6667 + 1.33 = \mathbf{1.99\ m.\ Ans.}$$

Problem 3.7 *A vertical sluice gate is used to cover an opening in a dam. The opening is 2 m wide and 1.2 m high. On the upstream of the gate, the liquid of sp. gr. 1.45, lies upto a height of 1.5 m above the top of the gate, whereas on the downstream side the water is available upto a height touching the top of the gate. Find the resultant force acting on the gate and position of centre of pressure. Find also the force acting horizontally at the top of the gate which is capable of opening it. Assume that the gate is hinged at the bottom.*

Solution. Given :

Width of gate, $b = 2$ m

Depth of gate, $d = 1.2$ m

$\therefore$ Area, $A = b \times d = 2 \times 1.2 = 2.4\ \text{m}^2$

Sp. gr. of liquid $= 1.45$

$\therefore$ Density of liquid, $\rho_1 = 1.45 \times 1000 = 1450$ kg/m^3

Let F_1 = Force exerted by the fluid of sp. gr. 1.45 on gate

F_2 = Force exerted by water on the gate.

The force F_1 is given by $F_1 = \rho_1 g \times A \times \bar{h}_1$

where $\rho_1 = 1.45 \times 1000 = 1450$ kg/m^2

$\bar{h}_1$ = Depth of C.G. of gate from free surface of liquid

$$= 1.5 + \frac{1.2}{2} = 2.1 \text{ m.}$$

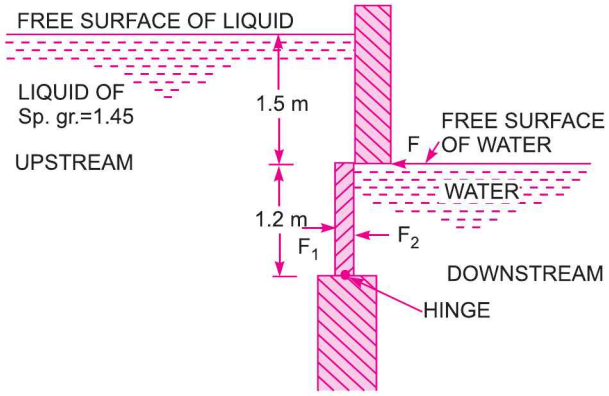


Fig. 3.9

$\therefore$ $F_1 = 1450 \times 9.81 \times 2.4 \times 2.1 = 71691$ N

Similarly, $F_2 = \rho_2 g . A \bar{h}_2$

where $\rho_2 = 1{,}000$ kg/m^3

$\bar{h}_2$ = Depth of C.G. of gate from free surface of water

$$= \frac{1}{2} \times 1.2 = 0.6 \text{ m}$$

$\therefore$ $F_2 = 1000 \times 9.81 \times 2.4 \times 0.6 = 14126$ N

(*i*) **Resultant force on the gate** = $F_1 - F_2 = 71691 - 14126 =$ **57565 N. Ans.**

(*ii*) **Position of centre of pressure of resultant force.** The force F_1 will be acting at a depth of h_1* from free surface of liquid, given by the relation

$$h_1^* = \frac{I_G}{A\bar{h}_1} + \bar{h}_1$$

where $I_G = \frac{bd^3}{12} = \frac{2 \times 1.2^3}{12} = 0.288$ m^4

$\therefore$ $h_1^* = \frac{.288}{2.4 \times 2.1} + 2.1 = 0.0571 + 2.1 = 2.1571$ m

$\therefore$ Distance of F_1 from hinge

$= (1.5 + 1.2) - h_1^* = 2.7 - 2.1571 = 0.5429$ m

The force F_2 will be acting at a depth of h_2* from free surface of water and is given by

$$h_2^* = \frac{I_G}{A\bar{h}_2} + \bar{h}_2$$

where $I_G = 0.288\ m^4$, $\bar{h}_2 = 0.6$ m, $A = 2.4\ m^2$,

$$h_2^* = \frac{0.288}{2.4 \times 0.6} + 0.6 = 0.2 + 0.6 = 0.8 \text{ m}$$

Distance of F_2 from hinge $= 1.2 - 0.8 = 0.4$ m

The resultant force 57565 N will be acting at a distance given by

$$= \frac{71691 \times .5429 - 14126 \times 0.4}{57565}$$

$$= \frac{38921 - 5650.4}{57565} \text{ m above hinge}$$

$$= \mathbf{0.578\ m\ above\ the\ hinge.\ Ans.}$$

(*iii*) **Force at the top of gate which is capable of opening the gate.** Let F is the force required on the top of the gate to open it as shown in Fig. 3.9. Taking the moments of F, F_1 and F_2 about the hinge, we get

$$F \times 1.2 + F_2 \times 0.4 = F_1 \times .5429$$

or

$$F = \frac{F_1 \times .5429 - F_2 \times 0.4}{1.2}$$

$$= \frac{71691 \times .5429 - 14126 \times 0.4}{1.2} = \frac{38921 - 5650.4}{1.2}$$

$$= \mathbf{27725.5\ N.\ Ans.}$$

Problem 3.8 *A caisson for closing the entrance to a dry dock is of trapezoidal form 16 m wide at the top and 10 m wide at the bottom and 6 m deep. Find the total pressure and centre of pressure on the caisson if the water on the outside is just level with the top and dock is empty.*

Solution. Given :

Width at top $= 16$ m

Width at bottom $= 10$ m

Depth, $d = 6$ m

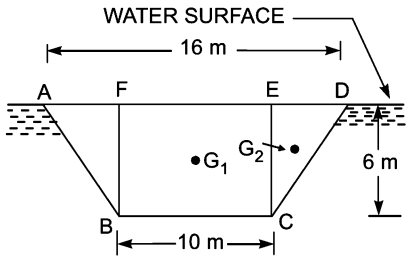


Fig. 3.10

Area of trapezoidal $ABCD$,

$$A = \frac{(BC + AD)}{2} \times d$$

$$= \frac{(10 + 16)}{2} \times 6 = 78\ m^2$$

Depth of C.G. of trapezoidal area $ABCD$ from free surface of water,

$$\bar{h} = \frac{10 \times 6 \times 3 + \frac{(16 - 10)}{2} \times 6 \times \frac{1}{3} \times 6}{78}$$

$$= \frac{180 + 36}{78} = 2.769 \text{ m from water surface.}$$

(*i*) **Total Pressure (F).** Total pressure, F is given by

$$F = \rho g A\bar{h} = 1000 \times 9.81 \times 78 \times 2.769 \text{ N}$$
$$= 2118783 \text{ N} = \textbf{2.118783 MN. Ans.}$$

(*ii*) **Centre of Pressure (h*).** Centre of pressure is given by equation (3.5) as

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h}$$

where I_G = M.O.I. of trapezoidal *ABCD* about its C.G.

Let I_{G_1} = M.O.I. of rectangle *FBCE* about its C.G.

I_{G_2} = M.O.I. of two Δs *ABF* and *ECD* about its C.G.

Then $$I_{G_1} = \frac{bd^3}{12} = \frac{10 \times 6^3}{12} = 180 \text{ m}^4$$

I_{G_1} is the M.O.I. of the rectangle about the axis passing through G_1.

∴ M.O.I. of the rectangle about the axis passing through the C.G. of the trapezoidal I_{G_1} + Area of rectangle × x_1^2

where x_1 is distance between the C.G. of rectangle and C.G. of trapezoidal

$$= (3.0 - 2.769) = 0.231 \text{ m}$$

∴ M.O.I. of *FBCE* passing through C.G. of trapezoidal

$$= 180 + 10 \times 6 \times (0.231)^2 = 180 + 3.20 = 183.20 \text{ m}^4$$

Now I_{G_2} = M.O.I. of Δ*ABD* in Fig. 3.11 about $G_2 = \frac{bd^3}{36}$

$$= \frac{(16 - 10) \times 6^3}{36} = 36 \text{ m}^4$$

The distance between the C.G. of triangle and C.G. of trapezoidal

$$= (2.769 - 2.0) = 0.769$$

∴ M.O.I. of the two Δs about an axis passing through C.G. of trapezoidal

$$= I_{G_2} + \text{Area of triangles} \times (.769)^2$$
$$= 36.0 + \frac{6 \times 6}{2} \times (.769)^2$$
$$= 36.0 + 10.64 = 46.64$$

∴ I_G = M.O.I. of trapezoidal about its C.G.
= M.O.I. of rectangle about the C.G. of trapezoidal
+ M.O.I. of triangles about the C.G. of the trapezoidal
= 183.20 + 46.64 = 229.84 m^4

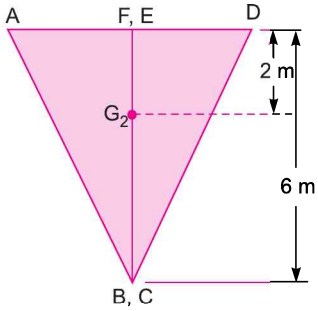


Fig. 3.11

∴ $$h^* = \frac{I_G}{A\bar{h}} + \bar{h}$$

where $A = 78$, $\bar{h} = 2.769$

$$h^* = \frac{229.84}{78 \times 2.769} + 2.769 = 1.064 + 2.769 = \textbf{3.833 m. Ans.}$$

Alternate Method

The distance of the C.G. of the trapezoidal channel from surface *AD* is given by (refer to Table 3.1 on page 71)

$$x = \frac{(2a+b)}{(a+b)} \times \frac{h}{3}$$

$$= \frac{(2\times 10+16)}{(10+16)} \times \frac{6}{3} \qquad (\because \quad a = 10,\ b = 16 \text{ and } h = 6)$$

$$= \frac{36}{26} \times 2 = 2.769 \text{ m}$$

This is also equal to the distance of the C.G. of the trapezoidal from free surface of water.

$$\bar{h} = 2.769 \text{ m}$$

$\therefore$ Total pressure, $F = \rho g A \bar{h}$ $(\because \quad A = 78)$

$= 1000 \times 9.81 \times 78 \times 2.769 \text{ N} =$ **2118783 N. Ans.**

Centre of Pressure, $(h^*) = \frac{I_G}{A\bar{h}} + \bar{h}$

Now I_G from Table 3.1 is given by,

$$I_G = \frac{\left(a^2 + 4ab + b^2\right)}{36(a+b)} \times h^3 = \frac{\left(10^2 + 4\times 10\times 16 + 16^2\right)}{36(10+16)} \times 6^3$$

$$= \frac{(100+640+256)}{36\times 26} \times 216 = 229.846 \text{ m}^4$$

$\therefore$ $$h^* = \frac{229.846}{78\times 2.769} + 2.769 \qquad (\because \quad A = 78 \text{ m}^2)$$

$= \mathbf{3.833\ m.\ Ans.}$

Problem 3.9 *A trapezoidal channel 2 m wide at the bottom and 1 m deep has side slopes 1 : 1. Determine :*

(i) the total pressure, and

(ii) the centre of pressure on the vertical gate closing the channel when it is full of water.

Solution. Given :

Width at bottom = 2 m

Depth, $d = 1$ m

Side slopes = 1 : 1

$\therefore$ Top width, $AD = 2 + 1 + 1 = 4$ m

Area of rectangle *FBEC*, $A_1 = 2 \times 1 = 2 \text{ m}^2$

Area of two triangles *ABF* and *ECD*, $A_2 = \frac{(4-2)}{2} \times 1 = 1 \text{ m}^2$

$\therefore$ Area of trapezoidal *ABCD*, $A = A_1 + A_2 = 2 + 1 = 3 \text{ m}^2$

Depth of C.G. of rectangle *FBEC* from water surface,

$$\bar{h}_1 = \frac{1}{2} = 0.5 \text{ m}$$

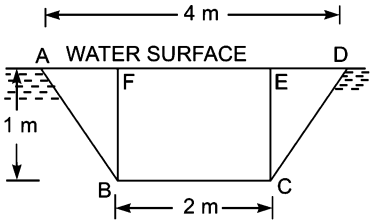


Fig. 3.12

Depth of C.G. of two triangles *ABF* and *ECD* from water surface,

$$\bar{h}_2 = \frac{1}{3} \times 1 = \frac{1}{3} \text{ m}$$

$\therefore$ Depth of C.G. of trapezoidal *ABCD* from free surface of water

$$\bar{h} = \frac{A_1 \times \bar{h}_1 + A_2 \times \bar{h}_2}{(A_1 + A_2)} = \frac{2 \times 0.5 + 1 \times 0.33333}{(2+1)} = .44444$$

(*i*) **Total Pressure (F).** Total pressure F is given by

$$F = \rho g A \bar{h}$$
$$= 1000 \times 9.81 \times 3.0 \times 0.44444 = \textbf{13079.9 N. Ans.}$$

(*ii*) **Centre of Pressure (h*).** M.O.I. of rectangle *FBCE* about its C.G.,

$$I_{G_1} = \frac{bd^3}{12} = \frac{2 \times 1^3}{12} = \frac{1}{6} \text{ m}^4$$

M.O.I. of *FBCE* about an axis passing through the C.G. of trapezoidal

or $$I_{G_1}{}^* = I_{G_1} + A_1 \times [\text{Distance between C.G. of rectangle and C.G. of trapezoidal}]^2$$

$$= \frac{1}{6} + 2 \times \left[\bar{h}_1 - \bar{h}\right]^2$$

$$= \frac{1}{6} + 2 \times [0.5 - .4444]^2 = .1666 + .006182 = 0.1727$$

M.O.I. of the two triangles *ABF* and *ECD* about their C.G.,

$$I_{G_2} = \frac{bd^3}{36} = \frac{(1+1) \times 1^3}{36} = \frac{2}{36} = \frac{1}{18} \text{ m}^4.$$

M.O.I. of the two triangles about the C.G. of trapezoidal,

$$I_{G_2}{}^* = I_{G_1} + A_2 \times [\text{Distance between C.G. of triangles and C.G. of trapezoidal}]^2$$

$$= \frac{1}{18} + 1 \times \left[\bar{h} - \bar{h}_2\right]^2 = \frac{1}{18} + 1 \times \left[.4444 - \frac{1}{3}\right]^2$$

$$= \frac{1}{18} + (.1111)^2 = 0.0555 + (.1111)^2$$
$$= .0555 + 0.01234 = 0.06789 \text{ m}^4$$

$\therefore$ M.O.I. of the trapezoidal about its C.G.

$$I_G = I_{G_1}{}^* + I_{G_2}{}^* = .1727 + .06789 = 0.24059 \text{ m}^4$$

Then centre of pressure (h^*) on the vertical trapezoidal,

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h} = \frac{0.24059}{3 \times .4444} + .4444 = 0.18046 + .4444 = 0.6248$$
$$\simeq \textbf{0.625 m. Ans.}$$

Alternate Method

The distance of the C.G. of the trapezoidal channel from surface AD is given by (refer to Table 3.1 on page 71).

$$x = \frac{(2a+b)}{(a+b)} \times \frac{h}{3} = \frac{(2\times 2+4)}{(2+4)} \times \frac{1}{3} \qquad (\because \quad a = 2,\ b = 4 \text{ and } h = 1)$$

$$= 0.444 \text{ m}$$

$\therefore \quad \bar{h} = x = 0.444$ m

$\therefore$ Total pressure, $F = \rho g A \bar{h} = 1000 \times 9.81 \times 3.0 \times .444 \qquad (\because \quad A = 3.0)$

$= \mathbf{13079\ N.\ Ans.}$

Centre of pressure, $h^* = \dfrac{I_G}{A\bar{h}} + \bar{h}$

where I_G from Table 3.1 is given by

$$I_G = \frac{(a^2+4ab+b^2)}{36(a+b)} \times h^3 = \frac{(2^2+4\times 2\times 4+4^2)}{36(2+4)} \times 1^3 = \frac{52}{36\times 6} = 0.2407 \text{ m}^4$$

$\therefore \quad h^* = \dfrac{0.2407}{3.0 \times .444} + .444 = \mathbf{0.625\ m.\ Ans.}$

Problem 3.10 *A square aperture in the vertical side of a tank has one diagonal vertical and is completely covered by a plane plate hinged along one of the upper sides of the aperture. The diagonals of the aperture are 2 m long and the tank contains a liquid of specific gravity 1.15. The centre of aperture is 1.5 m below the free surface. Calculate the thrust exerted on the plate by the liquid and position of its centre of pressure.*

Solution. Given : Diagonals of aperture, $AC = BD = 2$ m

$\therefore$ Area of square aperture, A = Area of ΔACB + Area of ΔACD

$$= \frac{AC \times BO}{2} + \frac{AC \times OD}{2} = \frac{2\times 1}{2} + \frac{2\times 1}{2} = 1 + 1 = 2.0 \text{ m}^2$$

Sp. gr. of liquid $= 1.15$

$\therefore$ Density of liquid, $\rho = 1.15 \times 1000 = 1150$ kg/m^3

Depth of centre of aperture from free surface,

$\bar{h} = 1.5$ m.

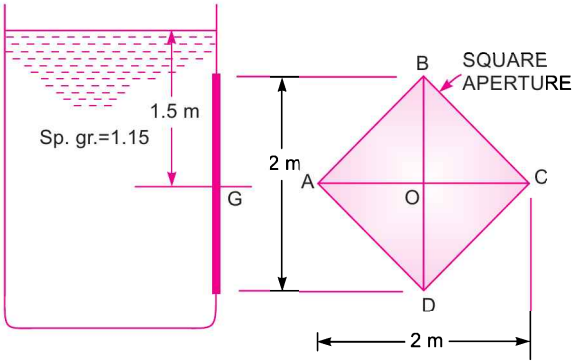


Fig. 3.13

(*i*) The thrust on the plate is given by

$$F = \rho g A \bar{h} = 1150 \times 9.81 \times 2 \times 1.5 = \textbf{33844.5. Ans.}$$

(*ii*) Centre of pressure (h^*) is given by

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h}$$

where I_G = M.O.I. of $ABCD$ about diagonal AC

= M.O.I. of triangle ABC about AC + M.O.I. of triangle ACD about AC

$$= \frac{AC \times OB^3}{12} + \frac{AC \times OD^3}{12} \qquad \left(\because \text{ M.O.I. of a triangle about its base} = \frac{bh^3}{12}\right)$$

$$= \frac{2 \times 1^3}{12} + \frac{2 \times 1^3}{12} = \frac{1}{6} + \frac{1}{6} = \frac{1}{3} \text{ m}^4$$

$$\therefore \qquad h^* = \frac{\frac{1}{3}}{2 \times 1.5} + 1.5 = \frac{1}{3 \times 2 \times 1.5} + 1.5 = \textbf{1.611 m. Ans.}$$

Problem 3.11 *A tank contains water upto a height of 0.5 m above the base. An immiscible liquid of sp. gr. 0.8 is filled on the top of water upto 1 m height. Calculate :*

(i) total pressure on one side of the tank,

(ii) the position of centre of pressure for one side of the tank, which is 2 m wide.

Solution. Given :

Depth of water = 0.5 m

Depth of liquid = 1 m

Sp. gr. of liquid = 0.8

Density of liquid, $\rho_1 = 0.8 \times 1000 = 800 \text{ kg/m}^3$

Density of water, $\rho_2 = 1000 \text{ kg/m}^3$

Width of tank = 2 m

(*i*) **Total pressure on one side** is calculated by drawing pressure diagram, which is shown in Fig. 3.14.

Intensity of pressure on top, $p_A = 0$

Intensity of pressure on D (or DE), $p_D = \rho_1 g.h_1$

$= 800 \times 9.81 \times 1.0 = 7848 \text{ N/m}^2$

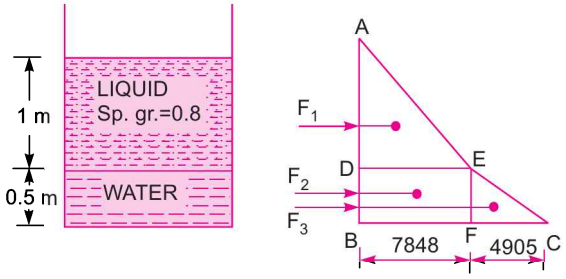


Fig. 3.14

Intensity of pressure on base (or BC), $p_B = \rho_1 g h_1 + \rho_2 g \times 0.5$

$= 7848 + 1000 \times 9.81 \times 0.5 = 7848 + 4905 = 12753 \text{ N/m}^2$

Now force F_1 = Area of $\Delta ADE \times$ Width of tank

$$= \frac{1}{2} \times AD \times DE \times 2.0 = \frac{1}{2} \times 1 \times 7848 \times 2.0 = 7848 \text{ N}$$

Force F_2 = Area of rectangle $DBFE$ × Width of tank

$= 0.5 \times 7848 \times 2 = 7848$ N

F_3 = Area of ΔEFC × Width of tank

$$= \frac{1}{2} \times EF \times FC \times 2.0 = \frac{1}{2} \times 0.5 \times 4905 \times 2.0 = 2452.5 \text{ N}$$

$\therefore$ Total pressure, $F = F_1 + F_2 + F_3$

$= 7848 + 7848 + 2452.5 =$ **18148.5 N. Ans.**

(*ii*) **Centre of Pressure (h*).** Taking the moments of all force about A, we get

$$F \times h^* = F_1 \times \frac{2}{3} AD + F_2 (AD + \frac{1}{2} BD) + F_3[AD + \frac{2}{3} BD]$$

$$18148.5 \times h^* = 7848 \times \frac{2}{3} \times 1 + 7848 \left(1.0 + \frac{0.5}{2}\right) + 2452.5 \left(1.0 + \frac{2}{3} \times .5\right)$$

$$= 5232 + 9810 + 3270 = 18312$$

$\therefore$ $$h^* = \frac{18312}{18148.5} =$$ **1.009 m from top. Ans.**

Problem 3.12 *A cubical tank has sides of 1.5 m. It contains water for the lower 0.6 m depth. The upper remaining part is filled with oil of specific gravity 0.9. Calculate for one vertical side of the tank:*

(a) total pressure, and

(b) position of centre of pressure.

Solution. Given :

Cubical tank of sides 1.5 m means the dimensions of the tank are 1.5 m × 1.5 m × 1.5 m.

Depth of water = 0.6 m

Depth of liquid = 1.5 – 0.6 = 0.9 m

Sp. gr. of liquid = 0.9

Density of liquid, $\rho_1 = 0.9 \times 1000 = 900$ kg/m^3

Density of water, $\rho_2 = 1000$ kg/m^3

(*a*) **Total pressure** on one vertical side is calculated by drawing pressure diagram, which is shown in Fig. 3.15.

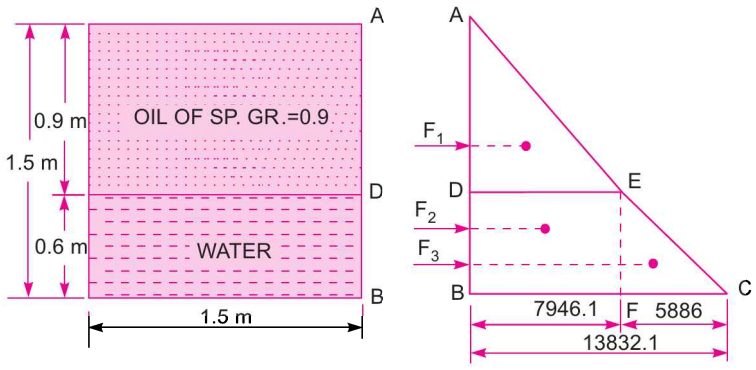


Fig. 3.15

Intensity of pressure at A, $p_A = 0$

Intensity of pressure at D, $p_D = \rho_1 g \times h = 900 \times 9.81 \times 0.9 = 7946.1$ N/m^2

Intensity of pressure at B, $p_B = \rho_1 g h_1 + \rho_2 g h_2 = 900 \times 9.81 \times 0.9 + 1000 \times 9.81 \times 0.6$

$= 7946.1 + 5886 = 13832.1$ N/m^2

Hence in pressure diagram :

$DE = 7946.1$ N/m^2, $BC = 13832.1$ N/m^2, $FC = 5886$ N/m^2

The pressure diagram is split into triangle ADE, rectangle $BDEF$ and triangle EFC. The total pressure force consists of the following components :

(*i*) Force F_1 = Area of triangle ADE × Width of tank

$= \left(\frac{1}{2} \times AD \times DE\right) \times 1.5$ ($\because$ Width = 1.5 m)

$= \left(\frac{1}{2} \times 0.9 \times 7946.1\right) \times 1.5$ N $= 5363.6$ N

This force will be acting at the C.G. of the triangle ADE, *i.e.*, at a distance of $\frac{2}{3} \times 0.9 = 0.6$ m below A

(*ii*) Force F_2 = Area of rectangle $BDEF$ × Width of tank

$= (BD \times DE) \times 1.5 = (0.6 \times 7946.1) \times 1.5 = 7151.5$

This force will be acting at the C.G. of the rectangle $BDEF$ *i.e.*, at a distance of $0.9 + \frac{0.6}{2} = 1.2$ m below A.

(*iii*) Force F_3 = Area of triangle EFC × Width of tank

$= \left(\frac{1}{2} \times EF \times FC\right) \times 1.5 = \left(\frac{1}{2} \times 0.6 \times 5886\right) \times 1.5 = 2648.7$ N

This force will be acting at the C.G. of the triangle EFC, *i.e.*, at a distance of $0.9 + \frac{2}{3} \times 0.6 = 1.30$ m below A.

$\therefore$ Total pressure force on one vertical face of the tank,

$F = F_1 + F_2 + F_3$

$= 5363.6 + 7151.5 + 2648.7 =$ **15163.8 N. Ans.**

(*b*) **Position of centre of pressure**

Let the total force F is acting at a depth of h^* from the free surface of liquid, *i.e.*, from A.

Taking the moments of all forces about A, we get

$$F \times h^* = F_1 \times 0.6 + F_2 \times 1.2 + F_3 \times 1.3$$

or
$$h^* = \frac{F_1 \times 0.6 + F_2 \times 1.2 + F_3 \times 1.3}{F}$$

$$= \frac{5363.6 \times 0.6 + 7151.5 \times 1.2 + 2648.7 \times 1.3}{15163.8}$$

$=$ **1.005 m from A. Ans.**

▶ 3.4 HORIZONTAL PLANE SURFACE SUBMERGED IN LIQUID

Consider a plane horizontal surface immersed in a static fluid. As every point of the surface is at the same depth from the free surface of the liquid, the pressure intensity will be equal on the entire surface and equal to, $p = \rho g h$, where h is depth of surface.

Let A = Total area of surface

Then total force, F, on the surface

$$= p \times \text{Area} = \rho g \times h \times A = \rho g A \bar{h}$$

where $\bar{h}$ = Depth of C.G. from free surface of liquid = h

also h^* = Depth of centre of pressure from free surface = h.

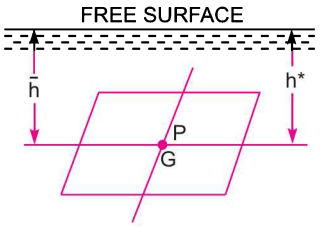


Fig. 3.16

Problem 3.13 *Fig. 3.17 shows a tank full of water. Find :*

(*i*) *Total pressure on the bottom of tank.*

(*ii*) *Weight of water in the tank.*

(*iii*) *Hydrostatic paradox between the results of (i) and (ii). Width of tank is 2 m.*

Solution. Given :

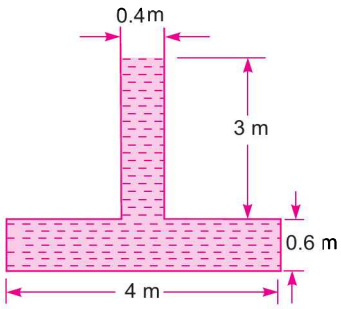


Fig. 3.17

Depth of water on bottom of tank

$$h_1 = 3 + 0.6 = 3.6 \text{ m}$$

Width of tank = 2 m

Length of tank at bottom = 4 m

$\therefore$ Area at the bottom, $A = 4 \times 2 = 8 \text{ m}^2$

(*i*) Total pressure F, on the bottom is

$$F = \rho g A \bar{h} = 1000 \times 9.81 \times 8 \times 3.6$$

$$= \mathbf{282528 \text{ N. Ans.}}$$

(*ii*) Weight of water in tank = $\rho g \times$ Volume of tank

$$= 1000 \times 9.81 \times [3 \times 0.4 \times 2 + 4 \times .6 \times 2]$$

$$= 1000 \times 9.81 \ [2.4 + 4.8] = \mathbf{70632 \text{ N. Ans.}}$$

(*iii*) From the results of (*i*) and (*ii*), it is observed that the total weight of water in the tank is much less than the total pressure at the bottom of the tank. This is known as Hydrostatic paradox.

▶ 3.5 INCLINED PLANE SURFACE SUBMERGED IN LIQUID

Consider a plane surface of arbitrary shape immersed in a liquid in such a way that the plane of the surface makes an angle θ with the free surface of the liquid as shown in Fig. 3.18.

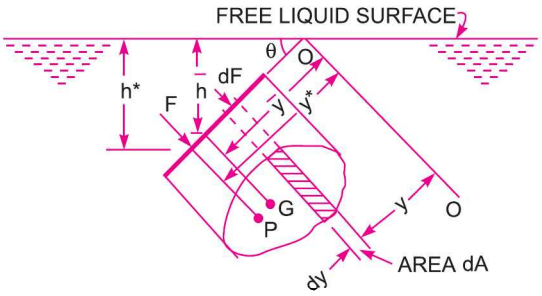


Fig. 3.18 *Inclined immersed surface.*

Let A = Total area of inclined surface

$\bar{h}$ = Depth of C.G. of inclined area from free surface

h^* = Distance of centre of pressure from free surface of liquid

θ = Angle made by the plane of the surface with free liquid surface.

Let the plane of the surface, if produced meet the free liquid surface at O. Then O-O is the axis perpendicular to the plane of the surface.

Let $\bar{y}$ = distance of the C.G. of the inclined surface from O-O

y^* = distance of the centre of pressure from O-O.

Consider a small strip of area dA at a depth 'h' from free surface and at a distance y from the axis O-O as shown in Fig. 3.18.

Pressure intensity on the strip, $p = \rho g h$

$\therefore$ Pressure force, dF, on the strip, $dF = p \times$ Area of strip $= \rho g h \times dA$

Total pressure force on the whole area, $F = \int dF = \int \rho g h dA$

But from Fig. 3.18, $\frac{h}{y} = \frac{\bar{h}}{\bar{y}} = \frac{h^*}{y^*} = \sin\theta$

$\therefore$ $h = y \sin\theta$

$\therefore$ $F = \int \rho g \times y \times \sin\theta \times dA = \rho g \sin\theta \int y dA$

But $\int y dA = A\bar{y}$

where $\bar{y}$ = Distance of C.G. from axis O-O

$\therefore$ $F = \rho g \sin\theta\ \bar{y} \times A$

$= \rho g A \bar{h}$ ($\because$ $\bar{h} = \bar{y} \sin\theta$) ...(3.6)

Centre of Pressure (h*)

Pressure force on the strip, $dF = \rho g h dA$

$= \rho g y \sin\theta\ dA$ [$h = y \sin\theta$]

Moment of the force, dF, about axis O-O

$= dF \times y = \rho g y \sin\theta\ dA \times y = \rho g \sin\theta\ y^2 dA$

Sum of moments of all such forces about O-O

$= \int \rho g \sin\theta\ y^2\ dA = \rho g \sin\theta \int y^2\ dA$

But $\int y^2\ dA$ = M.O.I. of the surface about O-$O = I_0$

$\therefore$ Sum of moments of all forces about O-$O = \rho g \sin\theta\ I_0$ ...(3.7)

Moment of the total force, F, about O-O is also given by

$= F \times y^*$ ...(3.8)

where y^* = Distance of centre of pressure from O-O.

Equating the two values given by equations (3.7) and (3.8)

$F \times y^* = \rho g \sin\theta\ I_0$

or $y^* = \frac{\rho g \sin\theta\ I_0}{F}$ ...(3.9)

Now $y^* = \frac{h^*}{\sin\theta}$, $F = \rho g A \bar{h}$

and I_0 by the theorem of parallel axis $= I_G + A\bar{y}^2$.

Substituting these values in equation (3.9), we get

$$\frac{h^*}{\sin\theta} = \frac{\rho g \sin\theta}{\rho g A\bar{h}}\left[I_G + A\bar{y}^2\right]$$

$$\therefore \quad h^* = \frac{\sin^2\theta}{A\bar{h}}\left[I_G + A\bar{y}^2\right]$$

But $$\frac{\bar{h}}{\bar{y}} = \sin\theta \quad \text{or} \quad \bar{y} = \frac{\bar{h}}{\sin\theta}$$

$$\therefore \quad h^* = \frac{\sin^2\theta}{A\bar{h}}\left[I_G + A \times \frac{\bar{h}^2}{\sin^2\theta}\right]$$

or $$h^* = \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h} \qquad \ldots(3.10)$$

If $\theta = 90°$, equation (3.10) becomes same as equation (3.5) which is applicable to vertically plane submerged surfaces.

In equation (3.10), I_G = M.O.I. of inclined surfaces about an axis passing through G and parallel to O-O.

Problem 3.14 (*a*) *A rectangular plane surface 2 m wide and 3 m deep lies in water in such a way that its plane makes an angle of 30° with the free surface of water. Determine the total pressure and position of centre of pressure when the upper edge is 1.5 m below the free water surface.*

Solution. Given :

Width of plane surface, $b = 2$ m

Depth, $d = 3$ m

Angle, $\theta = 30°$

Distance of upper edge from free water surface = 1.5 m

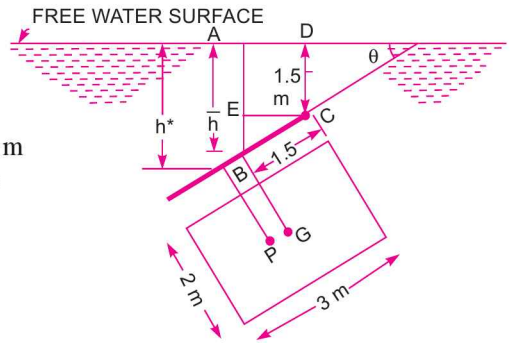


Fig. 3.19

(*i*) **Total pressure** force is given by equation (3.6) as

$$F = \rho g A\bar{h}$$

where $\rho = 1000$ kg/m^3

$A = b \times d = 3 \times 2 = 6$ m^2

$\therefore$ $\bar{h}$ = Depth of C.G. from free water surface

$= 1.5 + 1.5 \sin 30°$

$\{\because \quad \bar{h} = AE + EB = 1.5 + BC \sin 30° = 1.5 + 1.5 \sin 30°\}$

$= 1.5 + 1.5 \times \frac{1}{2} = 2.25$ m

$\therefore \quad F = 1000 \times 9.81 \times 6 \times 2.25 =$ **132435 N. Ans.**

(*ii*) **Centre of pressure (h*)**

Using equation (3.10), we have

$$h^* = \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h}, \quad \text{where } I_G = \frac{bd^3}{12} = \frac{2 \times 3^3}{12} = 4.5 \text{ m}^4$$

$$\therefore \quad h^* = \frac{4.5 \times \sin^2 30°}{6 \times 2.25} + 2.25 = \frac{4.5 \times \frac{1}{4}}{6 \times 2.25} + 2.25$$

$$= 0.0833 + 2.25 = \textbf{2.3333 m. Ans.}$$

Problem 3.14 (*b*) *A rectangular plane surface 3 m wide and 4 m deep lies in water in such a way that its plane makes an angle of 30° with the free surface of water. Determine the total pressure force and position of centre of pressure, when the upper edge is 2 m below the free surface.*

Solution. Given :

$$b = 3 \text{ m}, \; d = 4 \text{ m}, \; \theta = 30°$$

Distance of upper edge from free surface of water = 2 m

(*i*) **Total pressure force** is given by equation (3.6) as

$$F = \rho g A \bar{h}$$

where $\rho = 1000 \text{ kg/m}^3$,

$A = b \times d = 3 \times 4 = 12 \text{ m}^2$

and $\bar{h}$ = Depth of C.G. of plate from free water surface

$= 2 + BE = 2 + BC \sin \theta$

$= 2 + 2 \sin 30° = 2 + 2 \times \frac{1}{2} = 3 \text{ m}$

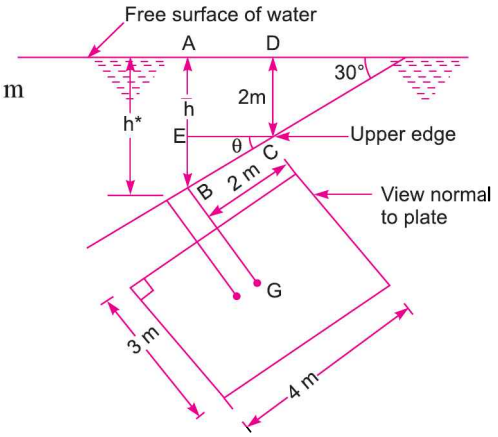


Fig. 3.19 (*a*)

$\therefore$ $F = 1000 \times 9.81 \times 12 \times 3 = 353167 \text{ N} = \mathbf{353.167 \text{ kN. Ans.}}$

(*ii*) **Centre of pressure (h*)**

Using equation (3.10), we have $h^* = \frac{I_G \sin^2 \theta}{A\bar{h}} + \bar{h}$

where $I_G = \frac{bd^3}{12} = \frac{3 \times 4^3}{12} = 16 \text{ m}^4$

$\therefore$ $h^* = \frac{16 \times \sin^2 30°}{12 \times 3} + 3 = \frac{16 \times \frac{1}{4}}{36} + 3 = \mathbf{3.111 \text{ m. Ans.}}$

Problem 3.15 (*a*) *A circular plate 3.0 m diameter is immersed in water in such a way that its greatest and least depth below the free surface are 4 m and 1.5 m respectively. Determine the total pressure on one face of the plate and position of the centre of pressure.*

Solution. Given :

Dia. of plate, $d = 3.0 \text{ m}$

$\therefore$ Area, $A = \frac{\pi}{4} d^2 = \frac{\pi}{4} (3.0)^2 = 7.0685 \text{ m}^2$

Distance $DC = 1.5 \text{ m}, \; BE = 4 \text{ m}$

Distance of C.G. from free surface

$= \bar{h} = CD + GC \sin \theta = 1.5 + 1.5 \sin \theta$

But $\sin \theta = \frac{AB}{BC} = \frac{BE - AE}{BC} = \frac{4.0 - DC}{3.0} = \frac{4.0 - 1.5}{3.0}$

$= \frac{2.5}{3.0} = 0.8333$

$\therefore$ $\bar{h} = 1.5 + 1.5 \times .8333 = 1.5 + 1.249 = 2.749 \text{ m}$

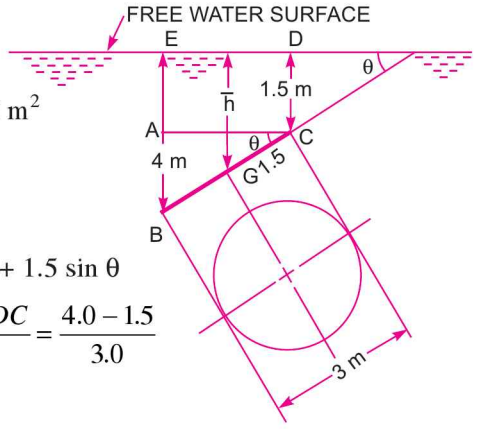


Fig. 3.20

(*i*) **Total pressure (F)**

$$F = \rho g A \bar{h}$$
$$= 1000 \times 9.81 \times 7.0685 \times 2.749 = \textbf{190621 N. Ans.}$$

(*ii*) **Centre of pressure (h*)**

Using equation (3.10), we have $h^* = \dfrac{I_G \sin^2\theta}{A\bar{h}} + \bar{h}$

where $I_G = \dfrac{\pi}{64} d^4 = \dfrac{\pi}{64} (3)^4 = 3.976 \text{ m}^4$

$$h^* = \frac{3.976 \times (.8333) \times .8333}{7.0685 \times 2.749} + 2.749 = 0.1420 + 2.749$$
$$= \textbf{2.891 m. Ans.}$$

Problem 3.15 (*b*) *If in the above problem, the given circular plate is having a concentric circular hole of diameter 1.5 m, then calculate the total pressure and position of the centre of pressure on one face of the plate.*

Solution. Given : [Refer to Fig. 3.20 (*a*)]

Dia. of plate, $d = 3.0$ m

$\therefore$ Area of solid plate $= \dfrac{\pi}{4} d^2 = \dfrac{\pi}{4} (3)^2 = 7.0685 \text{ m}^2$

Dia. of hole in the plate, $d_0 = 1.5$ m

$\therefore$ Area of hole $= \dfrac{\pi}{4} d_0^2 = \dfrac{\pi}{4} (1.5)^2 = 1.7671 \text{ m}^2$

$\therefore$ Area of the given plate, A = Area of solid plate – Area of hole

$= 7.0685 - 1.7671 = 5.3014 \text{ m}^2$

Distance $CD = 1.5$, $BE = 4$ m

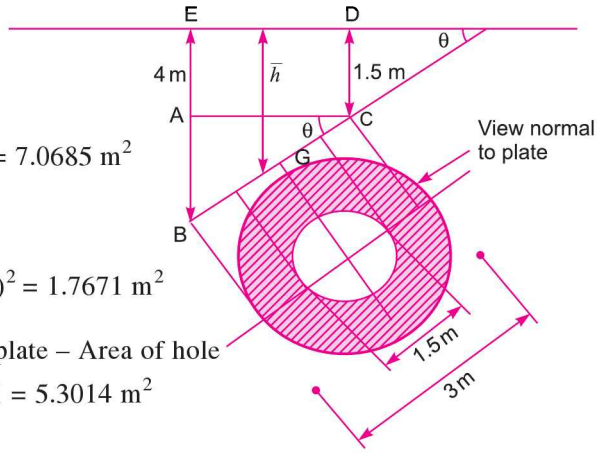


Fig. 3.20 (*a*)

Distance of C.G. from the free surface,

$$\bar{h} = CD + GC \sin\theta$$
$$= 1.5 + 1.5 \sin\theta$$

But $\sin\theta = \dfrac{AB}{BC} = \dfrac{BE - AE}{BC} = \dfrac{4 - 1.5}{3} = \dfrac{2.5}{3}$

$\therefore$ $\bar{h} = 1.5 + 1.5 \times \dfrac{2.5}{3} = 1.5 + 1.25 = 2.75$ m

(*i*) **Total pressure force (F)**

$$F = \rho g A \bar{h}$$
$$= 1000 \times 9.81 \times 5.3014 \times 2.75$$
$$= 143018 \text{ N} = \textbf{143.018 kN. Ans.}$$

(*ii*) **Position of centre of pressure (h*)**

Using equation (3.10), we have

$$h^* = \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h}$$

where $I_G = \frac{\pi}{64}[d^4 - d_0^4] = \frac{\pi}{64}[3^4 - 1.5^4]\ \text{m}^4$

$$A = \frac{\pi}{4}[d^2 - d_0^2] = \frac{\pi}{4}[3^2 - 1.5^2]\ \text{m}^2$$

$$\sin\theta = \frac{2.5}{3} \text{ and } \bar{h} = 2.75$$

$$\therefore \quad h^* = \frac{\frac{\pi}{64}\left[3^4 - 1.5^4\right] \times \left(\frac{2.5}{3}\right)^2}{\frac{\pi}{4}\left[3^2 - 1.5^2\right] \times 2.75} + 2.75$$

$$= \frac{\frac{1}{16}\left[3^2 + 1.5^2\right] \times \left(\frac{2.5}{3}\right)^2}{2.75} + 2.75 = \frac{1 \times 11.25 \times 6.25}{16 \times 2.75 \times 9} + 2.75$$

$$= 0.177 + 2.75 = \mathbf{2.927\ m.\ Ans.}$$

Problem 3.16 *A circular plate 3 metre diameter is submerged in water as shown in Fig. 3.21. Its greatest and least depths are below the surfaces being 2 metre and 1 metre respectively. Find : (i) the total pressure on front face of the plate, and (ii) the position of centre of pressure.*

Solution. Given :

Dia. of plate, $d = 3.0$ m

$\therefore$ Area, $A = \frac{\pi}{4}(3.0)^2 = 7.0685\ \text{m}^2$

Distance, $DC = 1$ m, $BE = 2$ m

In ΔABC, $\sin\theta = \frac{AB}{AC} = \frac{BE - AE}{BC} = \frac{BE - DC}{BC} = \frac{2.0 - 1.0}{3.0} = \frac{1}{3}$

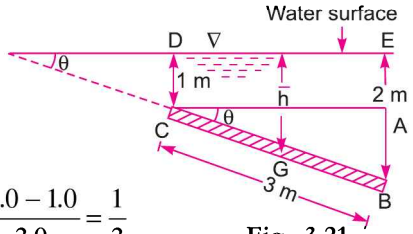


Fig. 3.21

The centre of gravity of the plate is at the middle of BC, *i.e.*, at a distance 1.5 m from C.

The distance of centre of gravity from the free surface of the water is given by

$$\bar{h} = CD + CG\sin\theta = 1.0 + 1.5 \times \tfrac{1}{3} \qquad (\because \sin\theta = \tfrac{1}{3})$$

$$= 1.5\ \text{m}.$$

(*i*) Total pressure on the front face of the plate is given by

$$F = \rho g A\bar{h}$$

$$= 1000 \times 9.81 \times 7.0685 \times 1.5 = \mathbf{104013\ N.\ Ans.}$$

(*ii*) Let the distance of the centre of pressure from the free surface of the water be h^*. Then using equation (3.10), we have

$$h^* = \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h}$$

where $I_G = \frac{\pi}{64} d^4 = \frac{\pi}{64} (3)^4$, $A = \frac{\pi}{4} d^2$, $\bar{h} = 1.5$ m and $\sin\theta = \frac{1}{3}$

Substituting the values, we get

$$h^* = \frac{\frac{\pi}{64} d^4 \times \left(\frac{1}{3}\right)^2}{\frac{\pi}{4} d^2 \times 1.5} + 1.5 = \frac{d^2}{16} \times \frac{1}{9 \times 1.5} + 1.5$$

$$= \frac{3^2}{16 \times 9 \times 1.5} + 1.5 = .0416 + 1.5 = \mathbf{1.5416\ m.\ Ans.}$$

Problem 3.17 *A rectangular gate 5 m ×2 m is hinged at its base and inclined at 60° to the horizontal as shown in Fig. 3.22. To keep the gate in a stable position, a counter weight of 5000 kgf is attached at the upper end of the gate as shown in figure. Find the depth of water at which the gate begins to fall. Neglect the weight of the gate and friction at the hinge and pulley.*

Solution. Given :

Length of gate = 5 m

Width of gate = 2 m

$\theta = 60°$

Weight, $W = 5000$ kgf

$= 5000 \times 9.81$ N

$= 49050$ N ($\because$ 1 kgf = 9.81 N)

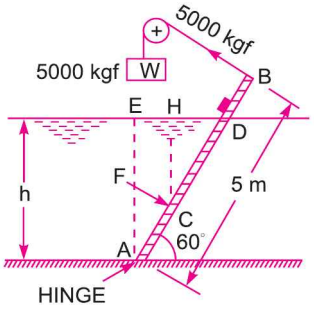


Fig. 3.22

As the pulley is frictionless, the force acting at $B = 49050$ N. First find the total force F acting on the gate AB for a given depth of water.

From figure, $AD = \frac{AE}{\sin\theta} = \frac{h}{\sin 60°} = \frac{5}{\sqrt{3}/2} = \frac{2h}{\sqrt{3}}$

$\therefore$ Area of gate immersed in water, $A = AD \times \text{Width} \times \frac{2h}{\sqrt{3}} \times 2 = \frac{4h}{\sqrt{3}}$ m^2

Also depth of the C.G. of the immersed area $= \bar{h} = \frac{h}{2} = 0.5\,h$

$\therefore$ Total force F is given by $F = \rho g A \bar{h} = 1000 \times 9.81 \times \frac{4h}{\sqrt{3}} \times \frac{h}{\sqrt{2}} = \frac{19620}{\sqrt{3}} h^2$N

The centre of pressure of the immersed surface, h^* is given by

$$h^* = \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h}$$

where I_G = M.O.I. of the immersed area

$$= \frac{b \times (AD)^3}{12} = \frac{2}{12} \times \left(\frac{2h}{\sqrt{3}}\right)^3 \qquad \left\{\because AD = \frac{2h}{\sqrt{3}}\right\}$$

$$= \frac{16h^3}{12 \times 3 \times \sqrt{3}} = \frac{4h^3}{9 \times \sqrt{3}} \text{ m}^4$$

$$\therefore \quad h^* = \frac{4h^3}{9\times\sqrt{3}} \times \frac{\left(\frac{\sqrt{3}}{2}\right)^2}{\frac{4h}{\sqrt{3}}\times\frac{h}{2}} + \frac{h}{2} = \frac{3h^3}{18h^2} + \frac{h}{2} = \frac{h}{6} + \frac{h}{2} = \frac{h+3h}{6} = \frac{2h}{3}$$

Now in the ΔCHD, $\quad CH = h^* = \frac{2h}{3}$, $\angle CDH = 60°$

$$\therefore \quad \frac{CH}{CD} = \sin 60°$$

$$\therefore \quad CD = \frac{CH}{\sin 60°} = \frac{h^*}{\sin 60°} = \frac{2h}{3\times\frac{\sqrt{3}}{2}} = \frac{4h}{3\times\sqrt{3}}$$

$$\therefore \quad AC = AD - CD = \frac{2h}{\sqrt{3}} - \frac{4h}{3\sqrt{3}} = \frac{6h-4h}{3\sqrt{3}} = \frac{2h}{3\sqrt{3}} \text{ m}$$

Taking the moments about hinge, we get

$$49050 \times 5.0 = F \times AC = \frac{19620}{\sqrt{3}} h^2 \times \frac{2h}{3\sqrt{3}}$$

or $$.245250 = \frac{39240\, h^3}{3\times 3}$$

$$\therefore \quad h^3 = \frac{9\times 245250}{39240} = 56.25$$

$$\therefore \quad h = (56.25)^{1/3} = \mathbf{3.83\ m.\ Ans.}$$

Problem 3.18 *An inclined rectangular sluice gate AB, 1.2 m by 5 m size as shown in Fig. 3.23 is installed to control the discharge of water. The end A is hinged. Determine the force normal to the gate applied at B to open it.*

Solution. Given :

$$A = \text{Area of gate} = 1.2 \times 5.0 = 6.0 \text{ m}^2$$

Depth of C.G. of the gate from free surface of the water $= \bar{h}$

$$= DG = BC - BE$$
$$= 5.0 - BG \sin 45°$$
$$= 5.0 - 0.6 \times \frac{1}{\sqrt{2}} = 4.576 \text{ m}$$

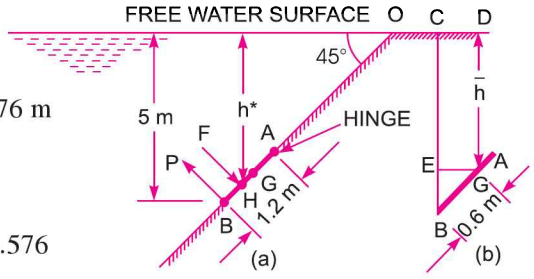


Fig. 3.23

The total pressure force (F) acting on the gate,

$$F = \rho g A \bar{h}$$
$$= 1000 \times 9.81 \times 6.0 \times 4.576$$
$$= 269343 \text{ N}$$

This force is acting at H, where the depth of H from free surface is given by

$$h^* = \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h}$$

where I_G = M.O.I. of gate = $\frac{bd^3}{12} = \frac{5.0 \times 1.2^3}{12} = 0.72$ m

$\therefore$ Depth of centre of pressure $h^* = \frac{0.72 \times \sin^2 45°}{6 \times 4.576} + 4.576 = .013 + 4.576 = 4.589$ m

But from Fig. 3.23 (*a*), $\frac{h^*}{OH} = \sin 45°$

$\therefore$ Distance, $OH = \frac{h^*}{\sin 45°} = \frac{4.589}{\frac{1}{\sqrt{2}}} = 4.589 \times \sqrt{2} = 6.489$ m

Distance, $BO = \frac{5}{\sin 45°} = 5 \times \sqrt{2} = 7.071$ m

Distance, $BH = BO - OH = 7.071 - 6.489 = 0.582$ m

$\therefore$ Distance $AH = AB - BH = 1.2 - 0.582 = 0.618$ m

Taking the moments about the hinge *A*

$$P \times AB = F \times (AH)$$

where *P* is the force normal to the gate applied at *B*

$\therefore$ $P \times 1.2 = 269343 \times 0.618$

$\therefore$ $P = \frac{269343 \times 0.618}{1.2} =$ **138708 N. Ans.**

Problem 3.19 *A gate supporting water is shown in Fig. 3.24. Find the height h of the water so that the gate tips about the hinge. Take the width of the gate as unity.*

Solution. Given : $\theta = 60°$

Distance, $AC = \frac{h}{\sin 60°} = \frac{2h}{\sqrt{3}}$

where h = Depth of water.

The gate will start tipping about hinge *B* if the resultant pressure force acts at *B*. If the resultant pressure force passes through a point which is lying from *B* to *C* anywhere on the gate, the gate will tip over the hinge. Hence limiting case is when the resultant force passes through *B*. But the resultant force passes through the centre of pressure. Hence for the given position, point *B* becomes the centre of pressure. Hence depth of centre of pressure,

$$h^* = (h - 3) \text{ m}$$

But h^* is also given by $= \frac{I_G \sin^2 \theta}{Ah} + \bar{h}$

Taking width of gate unity. Then

Area, $A = AC \times 1 = \frac{2h}{\sqrt{3}} \times 1$; $\bar{h} = \frac{h}{2}$

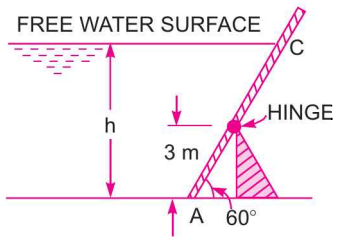


Fig. 3.24

$$I_G = \frac{bd^3}{12} = \frac{1 \times AC^3}{12} = \frac{1 \times \left(\frac{2h}{\sqrt{3}}\right)^3}{12} = \frac{8h^3}{12 \times 3 \times \sqrt{3}} = \frac{2h^3}{9 \times \sqrt{3}}$$

$$\therefore \qquad h^* = \frac{2h^3}{9\times\sqrt{3}} \times \frac{\sin^2 60°}{\frac{2h}{\sqrt{3}} \times \frac{h}{2}} + \frac{h}{2} = \frac{2h^3 \times \frac{3}{4}}{9h^2} + \frac{h}{2} = \frac{h}{6} + \frac{h}{2} = \frac{2h}{3}$$

Equating the two values of h^*,

$$h - 3 = \frac{2h}{3} \quad \text{or} \quad h - \frac{2h}{3} = 3 \quad \text{or} \quad \frac{h}{3} = 3$$

$$\therefore \qquad h = 3 \times 3 = 9 \text{ m}$$

$\therefore$ Height of water for tipping the gate = **9 m. Ans.**

Problem 3.20 *A rectangular sluice gate AB, 2 m wide and 3 m long is hinged at A as shown in Fig. 3.25. It is kept closed by a weight fixed to the gate. The total weight of the gate and weight fixed to the gate is 343350 N. Find the height of the water 'h' which will just cause the gate to open. The centre of gravity of the weight and gate is at G.*

Solution. Given :

Width of gate, $b = 2$ m ; Length of gate $L = 3$ m

$\therefore$ Area, $A = 2 \times 3 = 6 \text{ m}^2$

Weight of gate and $W = 343350$ N

Angle of inclination, $\theta = 45°$

Let h is the required height of water.

Depth of C.G. of the gate and weight = $\bar{h}$

From Fig. 3.25 (a),

$$\bar{h} = h - ED = h - (AD - AE)$$

$$= h - (AB \sin\theta - EG \tan\theta) \qquad \left\{\because \tan\theta = \frac{AE}{EG} \therefore AE = EG \tan\theta\right\}$$

$$= h - (3 \sin 45° - 0.6 \tan 45°)$$

$$= h - (2.121 - 0.6) = (h - 1.521) \text{ m}$$

The total pressure force, F is given by

$$F = \rho g A \bar{h} = 1000 \times 9.81 \times 6 \times (h - 1.521)$$

$$= 58860\ (h - 1.521) \text{ N.}$$

The total force F is acting at the centre of pressure as shown in Fig. 3.25 (b) at H. The depth of H from free surface is given by h^* which is equal to

$$h^* = \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h}, \text{ where } I_G = \frac{bd^3}{12} = \frac{2 \times 3^3}{12} = \frac{54}{12} = 4.5 \text{ m}^4$$

$$\therefore \qquad h^* = \frac{4.5 \times \sin^2 45°}{6 \times (h - 1.521)} + (h - 1.521) = \frac{0.375}{(h - 1.521)} + (h - 1.521) \text{ m}$$

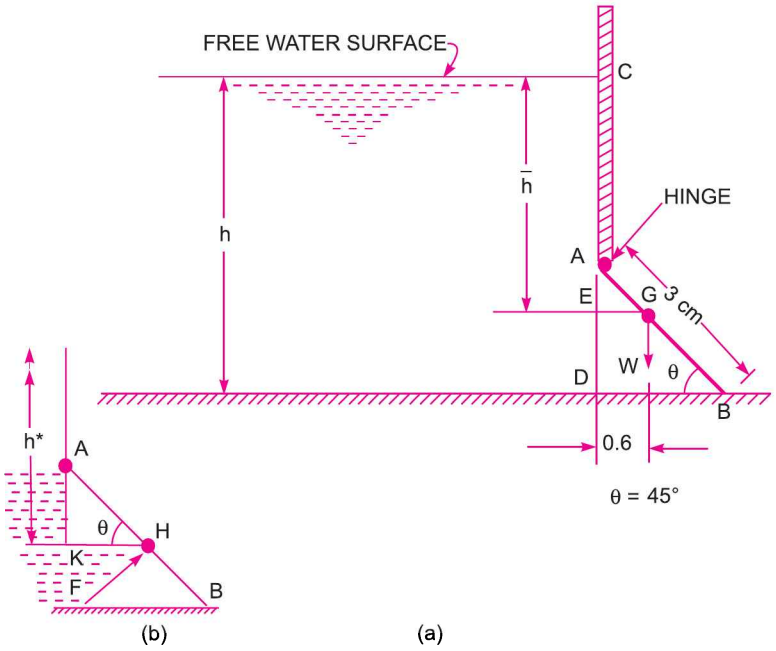


Fig. 3.25

Now taking moments about hinge A, we get

$$343350 \times EG = F \times AH$$

or
$$343350 \times 0.6 = F \times \frac{AK}{\sin 45°}$$

$$\left[\text{From } \Delta AKH, \text{Fig. 3.25 } (b)\ AK = AH \sin\theta = AH \sin 45° \therefore AH = \frac{AK}{\sin 45°}\right]$$

$$= \frac{58860\,(h - 1.521) \times AK}{\sin 45°}$$

$$\therefore \quad AK = \frac{343350 \times 0.6 \times \sin 45°}{58860\,(h - 1.521)} = \frac{0.3535 \times 7}{(h - 1.521)} \qquad ...(i)$$

But
$$AK = h^* - AC = \frac{.375}{(h - 1.521)} + (h - 1.521) - AC \qquad ...(ii)$$

But
$$AC = CD - AD = h - AB \sin 45^\circ = h - 3 \times \sin 45^\circ = h - 2.121$$

$\therefore$ Substituting this value in (ii), we get

$$AK = \frac{.375}{h - 1.521} + (h - 1.521) - (h - 2.121)$$

$$= \frac{.375}{h - 1.521} + 2.121 - 1.521 = \frac{.375}{h - 1.521} + 0.6 \qquad ...(iii)$$

Equating the two values of AK from (i) and (iii)

$$\frac{0.3535 \times 7}{h - 1.521} = \frac{0.375}{h - 1.521} + 0.6$$

or $0.3535 \times 7 = 0.375 + 0.6\,(h - 1.521) = 0.375 + 0.6\,h - 0.6 \times 1.521$

or $0.6h = 2.4745 - .375 + 0.6 \times 1.521 = 2.0995 + 0.9126 = 3.0121$

$$\therefore \quad h = \frac{3.0121}{0.6} = \textbf{5.02 m. Ans.}$$

Problem 3.21 *Find the total pressure and position of centre of pressure on a triangular plate of base 2 m and height 3 m which is immersed in water in such a way that the plane of the plate makes an angle of 60° with the free surface of the water. The base of the plate is parallel to water surface and at a depth of 2.5 m from water surface.*

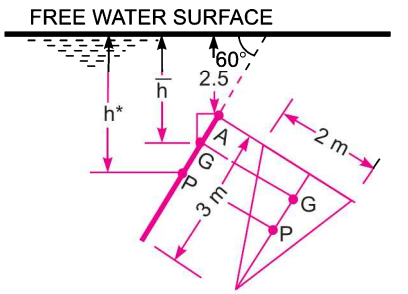


Fig. 3.26

Solution. Given :

Base of plate, $b = 2$ m

Height of plate, $h = 3$ m

$\therefore$ Area, $A = \dfrac{b \times h}{2} = \dfrac{2 \times 3}{2} = 3 \text{ m}^2$

Inclination, $\theta = 60°$

Depth of centre of gravity from free surface of water,

$$\bar{h} = 2.5 + AG \sin 60°$$

$$= 2.5 + \frac{1}{3} \times 3 \times \frac{\sqrt{3}}{2} \qquad \left\{\because AG = \frac{1}{3} \text{ of height of triangle}\right\}$$

$$= 2.5 + .866 \text{ m} = 3.366 \text{ m}$$

(*i*) **Total pressure force (F)**

$$F = \rho g A \bar{h} = 1000 \times 9.81 \times 3 \times 3.366 = \textbf{99061.38 N. Ans.}$$

(*ii*) **Centre of pressure (h*).** Depth of centre of pressure from free surface of water is given by

$$h^* = \frac{I_G \sin^2 \theta}{A\bar{h}} + \bar{h}$$

where $I_G = \dfrac{bh^3}{36} = \dfrac{2 \times 3^3}{36} = \dfrac{3}{2} = 1.5 \text{ m}^4$

$$\therefore \quad h^* = \frac{1.5 \times \sin^2 60°}{3 \times 3.366} + 3.366 = 0.111 + 3.366 = \textbf{3.477 m. Ans.}$$

▶ 3.6 CURVED SURFACE SUB-MERGED IN LIQUID

Consider a curved surface *AB*, sub-merged in a static fluid as shown in Fig. 3.27. Let *dA* is the area of a small strip at a depth of *h* from water surface.

Then pressure intensity on the area *dA* is $= \rho g h$

and pressure force, $dF = p \times \text{Area} = \rho g h \times dA$ ...(3.11)

This force *dF* acts normal to the surface.

Hence total pressure force on the curved surface should be

$$F = \int \rho g h dA \qquad \text{...(3.12)}$$

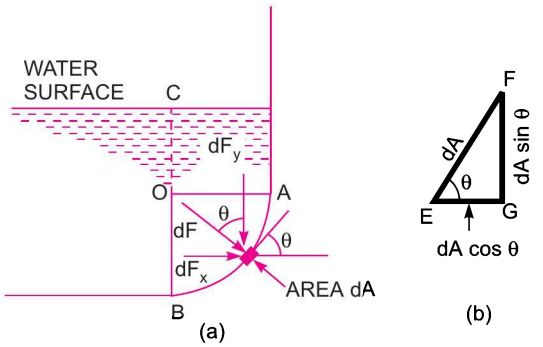


Fig. 3.27

But here as the direction of the forces on the small areas are not in the same direction, but varies from point to point. Hence integration of equation (3.11) for curved surface is impossible. The problem can, however, be solved by resolving the force dF in two components dF_x and dF_y in the x and y directions respectively. The total force in the x and y directions, *i.e.*, F_x and F_y are obtained by integrating dF_x and dF_y. Then total force on the curved surface is

$$F = \sqrt{F_x^2 + F_y^2} \qquad ...(3.13)$$

and inclination of resultant with horizontal is $\tan \phi = \dfrac{F_y}{F_x}$ ...(3.14)

Resolving the force dF given by equation (3.11) in x and y directions :

$$dF_x = dF \sin \theta = \rho g h dA \sin \theta \qquad \{\because dF = \rho g h dA\}$$

and $$dF_y = dF \cos \theta = \rho g h dA \cos \theta$$

Total forces in the x and y direction are :

$$F_x = \int dF_x = \int \rho g h dA \sin \theta = \rho g \int h dA \sin \theta \qquad ...(3.15)$$

and $$F_y = \int dF_y = \int \rho g h dA \cos \theta = \rho g \int h dA \cos \theta \qquad ...(3.16)$$

Fig. 3.27 (*b*) shows the enlarged area dA. From this figure, *i.e.*, ΔEFG,

$$EF = dA$$
$$FG = dA \sin \theta$$
$$EG = dA \cos \theta$$

Thus in equation (3.15), $dA \sin \theta = FG$ = Vertical projection of the area dA and hence the expression $\rho g \int h dA \sin \theta$ represents the total pressure force on the projected area of the curved surface on the vertical plane. Thus

F_x = Total pressure force on the projected area of the curved surface on vertical plane. ...(3.17)

Also $dA \cos \theta = EG$ = horizontal projection of dA and hence $hdA \cos \theta$ is the volume of the liquid contained in the elementary area dA upto free surface of the liquid. Thus $\int hdA \cos \theta$ is the total volume contained between the curved surface extended upto free surface.

Hence $\rho g \int hdA \cos \theta$ is the total weight supported by the curved surface. Thus

$$F_y = \rho g \int hdA \cos \theta$$

= weight of liquid supported by the curved surface upto free surface of liquid. ...(3.18)

In Fig. 3.28, the curved surface AB is not supporting any fluid. In such cases, F_y is equal to the weight of the imaginary liquid supported by AB upto free surface of liquid. The direction of F_y will be taken in upward direction.

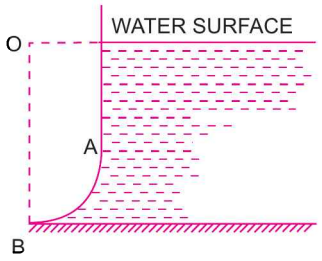


Fig. 3.28

Problem 3.22 *Compute the horizontal and vertical components of the total force acting on a curved surface AB, which is in the form of a quadrant of a circle of radius 2 m as shown in Fig. 3.29. Take the width of the gate as unity.*

Solution. Given :

Width of gate = 1.0 m

Radius of the gate = 2.0 m

$\therefore$ Distance $AO = OB = 2$ m

Horizontal force, F_x exerted by water on gate is given by equation (3.17) as

F_x = Total pressure force on the projected area of curved surface AB on vertical plane

= Total pressure force on OB

{projected area of curved surface on vertical plane = $OB \times 1$}

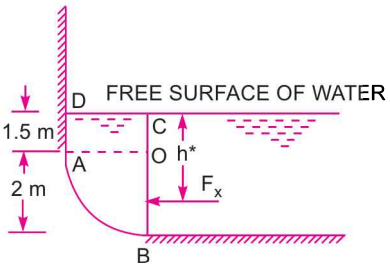


Fig. 3.29

$$= \rho g A \bar{h}$$

$$= 1000 \times 9.81 \times 2 \times 1 \times \left(1.5 + \frac{2}{2}\right)$$

{$\because$ Area of $OB = A = BO \times 1 = 2 \times 1 = 2$,

$\bar{h}$ = Depth of C.G. of OB from free surface = $1.5 + \frac{2}{2}$}

$$F_x = 9.81 \times 2000 \times 2.5 = \mathbf{49050\ N.\ Ans.}$$

The point of application of F_x is given by $h^* = \frac{I_G}{A\bar{h}} + \bar{h}$

where I_G = M.O.I. of OB about its C.G. = $\frac{bd^3}{12} = \frac{1 \times 2^3}{12} = \frac{2}{3}$ m^4

$$\therefore \quad h^* = \frac{\frac{2}{3}}{2 \times 2.5} + 2.5 = \frac{1}{7.5} + 2.5 \text{ m}$$

= 0.1333 + 2.5 = 2.633 m from free surface.

Vertical force, F_y, exerted by water is given by equation (3.18)

F_y = Weight of water supported by AB upto free surface

= Weight of portion $DABOC$

= Weight of $DAOC$ + Weight of water AOB

= ρg [Volume of $DAOC$ + Volume of AOB]

$$= 1000 \times 9.81 \left[AD \times AO \times 1 + \frac{\pi}{4}(AO)^2 \times 1 \right]$$

$$= 1000 \times 9.81 \left[1.5 \times 2.0 \times 1 + \frac{\pi}{4} \times 2^2 \times 1\right]$$

$$= 1000 \times 9.81 \ [3.0 + \pi]\text{N} = \textbf{60249.1 N. Ans.}$$

Problem 3.23 *Fig. 3.30 shows a gate having a quadrant shape of radius 2 m. Find the resultant force due to water per metre length of the gate. Find also the angle at which the total force will act.*

Solution. Given :

Radius of gate = 2 m

Width of gate = 1 m

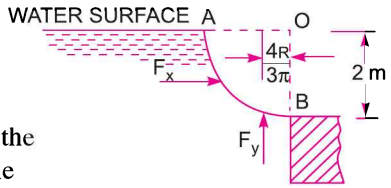


Fig. 3.30

Horizontal Force

F_x = Force on the projected area of the curved surface on vertical plane

= Force on $BO = \rho g A \bar{h}$

where A = Area of $BO = 2 \times 1 = 2\ \text{m}^2$, $\bar{h} = \frac{1}{2} \times 2 = 1$ m ;

$$F_x = 1000 \times 9.81 \times 2 \times 1 = 19620 \text{ N}$$

This will act at a depth of $\frac{2}{3} \times 2 = \frac{4}{3}$ m from free surface of liquid,

Vertical Force, F_y

F_y = Weight of water (imagined) supported by AB

= $\rho g \times$ Area of $AOB \times 1.0$

$$= 1000 \times 9.81 \times \frac{\pi}{4} (2)^4 \times 1.0 = 30819 \text{ N}$$

This will act at a distance of $\frac{4R}{3\pi} = \frac{4 \times 2.0}{3\pi} = 0.848$ m from OB.

∴ Resultant force, F is given by

$$F = \sqrt{F_x^2 + F_y^2}$$

$$= \sqrt{19620 + 30819} = \sqrt{384944400 + 949810761}$$

$$= \textbf{36534.4 N. Ans.}$$

The angle made by the resultant with horizontal is given by

$$\tan \theta = \frac{F_y}{F_x} = \frac{30819}{19620} = 1.5708$$

∴ $\theta = \tan^{-1} 1.5708 = \textbf{57° 31′. Ans.}$

Problem 3.24 *Find the magnitude and direction of the resultant force due to water acting on a roller gate of cylindrical form of 4.0 m diameter, when the gate is placed on the dam in such a way that water is just going to spill. Take the length of the gate as 8 m.*

Solution. Given :

Dia. of gate = 4 m

∴ Radius, R = 2 m

Length of gate, l = 8 m

Horizontal force, F_x acting on the gate is

$F_x = \rho g A \bar{h}$ = Force on projected area of curved surface ACB on vertical plane

= Force on vertical area AOB

where A = Area of $AOB = 4.0 \times 8.0 = 32.0\ m^2$

$\bar{h}$ = Depth of C.G. of $AOB = 4/2 = 2.0$ m

$\therefore$ $F_x = 1000 \times 9.81 \times 32.0 \times 2.0$

$= 627840$ N.

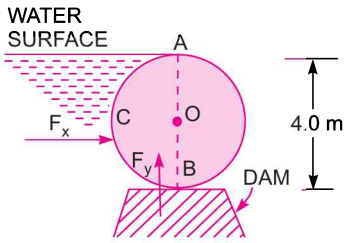


Fig. 3.31

Vertical force, F_y is given by

F_y = Weight of water enclosed or supported (actually or imaginary) by the curved surface ACB

$= \rho g \times$ Volume of portion ACB

$= \rho g \times$ Area of $ACB \times l$

$$= 1000 \times 9.81 \times \frac{\pi}{2}(R)^2 \times 8.0 = 9810 \times \frac{\pi}{2}(2)^2 \times 8.0 = 493104 \text{ N}$$

It will be acting in the upward direction.

$\therefore$ Resultant force, $F = \sqrt{F_x^2 + F_y^2} = \sqrt{627840 + 493104}$ = **798328 N. Ans.**

Direction of resultant force is given by $\tan \theta = \dfrac{F_y}{F_x} = \dfrac{493104}{627840} = 0.7853$

$\therefore$ θ = **31° 8′. Ans.**

Problem 3.25 *Find the horizontal and vertical component of water pressure acting on the face of a tainter gate of 90° sector of radius 4 m as shown in Fig. 3.32. Take width of gate unity.*

Solution. Given :

Radius of gate, $R = 4$ m

Horizontal component of force acting on the gate is

F_x = Force on area of gate projected on vertical plane

= Force on area ADB

$= \rho g A \bar{h}$

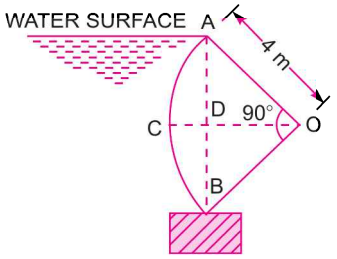


Fig. 3.32

where $A = AB \times$ Width of gate

$= 2 \times AD \times 1$ ($\because AB = 2AD$)

$= 2 \times 4 \times \sin 45° = 8 \times .707 = 5.656\ m^2$ {$\because AD = 4 \sin 45°$}

$$\bar{h} = \frac{AB}{2} = \frac{5.656}{2} = 2.828 \text{ m}$$

$\therefore$ $F_x = 1000 \times 9.81 \times 5.656 \times 2.828$ N = **156911 N. Ans.**

Vertical component

F_y = Weight of water supported or enclosed by the curved surface

= Weight of water in portion $ACBDA$

$= \rho g \times$ Area of $ACBDA \times$ Width of gate

$= 1000 \times 9.81 \times$ [Area of sector $ACBOA$ − Area of ΔABO] $\times 1$

$$= 9810 \times \left[\frac{\pi}{4}R^2 - \frac{AO \times BO}{2}\right] \qquad [\because \Delta AOB \text{ is a right angled}]$$

$$= 9810 \times \left[\frac{\pi}{4}4^2 - \frac{4 \times 4}{2}\right] = \mathbf{44796\ N.\ Ans.}$$

Problem 3.26 *Calculate the horizontal and vertical components of the water pressure exerted on a tainter gate of radius 8 m as shown in Fig. 3.33. Take width of gate unity.*

Solution. The horizontal component of water pressure is given by

$F_x = \rho g A\bar{h}$ = Force on the area projected on vertical plane

= Force on the vertical area of BD

where $A = BD \times$ Width of gate $= 4.0 \times 1 = 4.0$ m

$$\bar{h} = \frac{1}{2} \times 4 = 2 \text{ m}$$

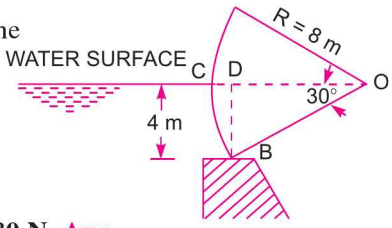


Fig. 3.33

$$F_x = 1000 \times 9.81 \times 4.0 \times 2.0 = \mathbf{78480\ N.\ Ans.}$$

Vertical component of the water pressure is given by

F_y = Weight of water supported or enclosed (imaginary) by curved surface CB

= Weight of water in the portion $CBDC$

= $\rho g \times$ [Area of portion $CBDC$] $\times$ Width of gate

= $\rho g \times$ [Area of sector CBO – Area of the triangle BOD] $\times 1$

$$= 1000 \times 9.81 \times \left[\frac{30}{360} \times \pi R^2 - \frac{BD \times DO}{2}\right]$$

$$= 9810 \times \left[\frac{1}{12}\pi \times 8^2 - \frac{4.0 \times 8.8 \cos 30°}{2}\right]$$

$\{\because DO = BO \cos 30° = 8 \times \cos 30°\}$

$= 9810 \times [16.755 - 13.856] = \mathbf{28439\ N.\ Ans.}$

Problem 3.27 *A cylindrical gate of 4 m diameter 2 m long has water on its both sides as shown in Fig. 3.34. Determine the magnitude, location and direction of the resultant force exerted by the water on the gate. Find also the least weight of the cylinder so that it may not be lifted away from the floor.*

Solution. Given :

Dia. of gate = 4 m

Radius = 2 m

(*i*) The forces acting on the left side of the cylinder are :

The horizontal component, F_{x_1}

where F_{x_1} = Force of water on area projected on vertical plane

= Force on area AOC

= $\rho g A\bar{h}$

= $1000 \times 9.81 \times 8 \times 2$

= 156960 N

where $A = AC \times$ Width $= 4 \times 2$

$= 8\ m^2$

$= \bar{h} = \frac{1}{2} \times 4 = 2$ m

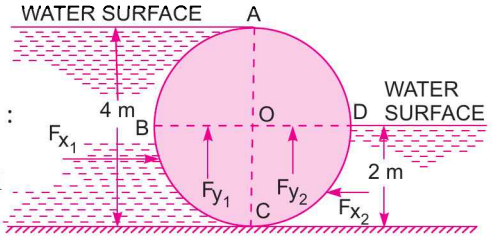


Fig. 3.34

F_{y_1} = weight of water enclosed by $ABCOA$

$$= 1000 \times 9.81 \times \left[\frac{\pi}{2}R^2\right] \times 2.0 = 9810 \times \frac{\pi}{2} \times 2^2 \times 2.0 = \textbf{123276 N.}$$

Right Side of the Cylinder

$F_{x_2} = \rho g A_2 \bar{h}_2$ = Force on vertical area CO

$$= 1000 \times 9.81 \times 2 \times 2 \times \frac{2}{2}\left\{ \quad A_2 = CO \times 1 = 2 \times 1 = 2 \text{ m}^2, \bar{h}_2 = \frac{2}{2} = 1.0 \right\}$$

$$= 39240 \text{ N}$$

F_{y_2} = Weight of water enclosed by $DOCD$

$$= \rho g \times \left[\frac{\pi}{4}R^2\right] \times \text{Width of gate}$$

$$= 1000 \times 9.81 \times \frac{\pi}{4} \times 2^2 \times 2 = 61638 \text{ N}$$

$\therefore$ Resultant force in the direction of x,

$$F_x = F_{x_1} - F_{x_2} = 156960 - 39240 = 117720 \text{ N}$$

Resultant force in the direction of y,

$$F_y = F_{y_1} + F_{y_2} = 123276 + 61638 = 184914 \text{ N}$$

(i) **Resultant force,** F is given as

$$F = \sqrt{F_x^2 + F_y^2} = \sqrt{(117720)^2 + (184914)^2} = \textbf{219206 N. Ans.}$$

(ii) **Direction of resultant force** is given by

$$\tan \theta = \frac{F_y}{F_x} = \frac{184914}{117720} = 1.5707$$

$\therefore$ $\theta = \textbf{57° 31′. Ans.}$

(iii) **Location of the resultant force**

Force, F_{x_1} acts at a distance of $\frac{2 \times 4}{3} = 2.67$ m from the top surface of water on left side, while F_{x_2} acts at a distance of $\frac{2}{3} \times 2 = 1.33$ m from free surface on the right side of the cylinder. The resultant force F_x in the direction of x will act at a distance of y from the bottom as

$$F_x \times y = F_{x_1}[4 - 2.67] - F_{x_2}[2 - 1.33]$$

or $$117720 \times y = 156960 \times 1.33 - 39240 \times .67 = 208756.8 - 26290.8 = 182466$$

$\therefore$ $$y = \frac{182466}{117720} = 1.55 \text{ m from the bottom.}$$

Force F_{y_1} acts at a distance $\frac{4R}{3\pi}$ from AOC or at a distance $\frac{4 \times 2.0}{3\pi} = 0.8488$ m from AOC towards left of AOC.

Also F_{y_2} acts at a distance $\frac{4R}{3\pi} = 0.8488$ m from AOC towards the right of AOC. The resultant force F_y will act at a distance x from AOC which is given by

$$F_y \times x = F_{y_1} \times .8488 - F_{y_2} \times .8488$$

or $$184914 \times x = 123276 \times .8488 - 61638 \times .8488 = .8488\ [123276 - 61638] = 52318.4$$

$$\therefore \quad x = \frac{52318.4}{184914} = 0.2829 \text{ m from AOC.}$$

(*iv*) **Least weight of cylinder.** The resultant force in the upward direction is

$$F_y = 184914 \text{ N}$$

Thus the weight of cylinder should not be less than the upward force F_y. Hence least weight of cylinder should be at least.

= 184914 N. Ans.

Problem 3.28 *Fig. 3.35 shows the cross-section of a tank full of water under pressure. The length of the tank is 2 m. An empty cylinder lies along the length of the tank on one of its corner as shown. Find the horizontal and vertical components of the force acting on the curved surface ABC of the cylinder.*

Solution. Radius, $R = 1$ m

Length of tank, $l = 2$ m

Pressure, $p = 0.2 \text{ kgf/cm}^2 = 0.2 \times 9.81 \text{ N/cm}^2$

$= 1.962 \text{ N/cm}^2 = 1.962 \times 10^4 \text{ N/m}^2$

$\therefore$ Pressure head, $h = \dfrac{p}{\rho g} = \dfrac{1.962 \times 10^4}{1000 \times 9.81} = 2$ m

$\therefore$ Free surface of water will be at a height of 2 m from the top of the tank.

$\therefore$ Fig. 3.36 shows the equivalent free surface of water.

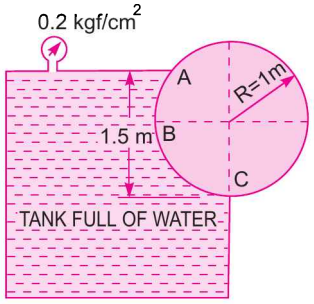


Fig. 3.35

(*i*) **Horizontal Component of Force**

$$F_x = \rho g A \bar{h}$$

where A = Area projected on vertical plane

$= 1.5 \times 2.0 = 3.0 \text{ m}^2$

$$\bar{h} = 2 + \frac{1.5}{2} = 2.75$$

$$\therefore \quad F_x = 1000 \times 9.81 \times 3.0 \times 2.75$$

= 80932.5 N. Ans.

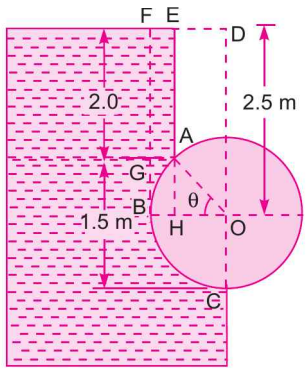


Fig. 3.36

(*ii*) **Vertical Component of Force**

F_y = Weight of water enclosed or supported actually or imaginary by curved surface ABC

= Weight of water in the portion *CODE ABC*

= Weight of water in *CODFBC* – Weight of water in *AEFB*

But weight of water in *CODFBC*

= Weight of water in [*COB* + *ODFBO*]

$$= \rho g \left[\frac{\pi R^2}{4} + BO \times OD\right] \times 2 = 1000 \times 9.81 \left[\frac{\pi}{4} \times 1^2 + 1.0 \times 2.5\right] \times 2$$

$= 64458.5$ N

Weight of water in *AEFB* = ρg [Area of *AEFB*] $\times$ 2.0

$= 1000 \times 9.81$ [Area of $(AEFG + AGBH - AHB)$] $\times 2.0$

In ΔAHO, $\sin\theta = \dfrac{AH}{AO} = \dfrac{0.5}{1.0} = 0.5 \quad \therefore \quad \theta = 30°$

$BH = BO - HO = 1.0 - AO \cos\theta = 1.0 - 1 \times \cos 30° = 0.134$

Area, ABH = Area ABO − Area AHO

$$= \pi R^2 \times \frac{30}{360} - \frac{AH \times HO}{2.0} = \frac{\pi R^2}{12} - \frac{0.5 \times .866}{2} = 0.0453$$

$\therefore$ Weight of water in $AEFB$

$= 9810 \times [AE \times AG + AG \times AH - 0.0453] \times 2.0$

$= 9810 \times [2.0 \times .134 + .134 \times .5 - .0453] \times 2.0$

$= 9810 \times [.268 + .067 - .0453] \times 2.0 = 5684$ N

$\therefore \quad F_y = 64458.5 - 5684 =$ **58774.5 N. Ans.**

Problem 3.29 *Find the magnitude and direction of the resultant water pressure acting on a curved face of a dam which is shaped according to the relation* $y = \dfrac{x^2}{9}$ *as shown in Fig. 3.37. The height of the water retained by the dam is 10 m. Consider the width of the dam as unity.*

Solution. Equation of curve AB is

$$y = \frac{x^2}{9} \quad \text{or} \quad x^2 = 9y$$

$\therefore \quad x = \sqrt{9y} = 3\sqrt{y}$

Height of water, $h = 10$ m

Width, $b = 1$ m

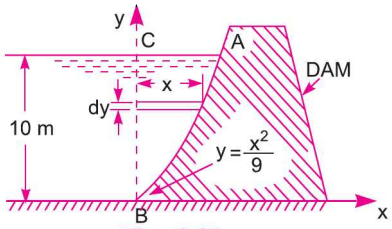


Fig. 3.37

The horizontal component, F_x is given by

F_x = Pressure due to water on the curved area projected on vertical plane

= Pressure on area BC

$= \rho g A \bar{h}$

where $A = BC \times 1 = 10 \times 1$ m^2, $\bar{h} = \frac{1}{2} \times 10 = 5$ m

$F_x = 1000 \times 9.81 \times 10 \times 5 = 490500$ N

Vertical component, F_y is given by

F_y = Weight of water supported by the curve AB

= Weight of water in the portion ABC

$= \rho g$[Area of ABC] $\times$ Width of dam

$$= \rho g \left[\int_0^{10} x \times dy\right] \times 1.0 \qquad \left\{\text{Area of strip} = xdy \quad \therefore \text{Area } ABC = \int_0^{10} xdy\right\}$$

$$= 1000 \times 9.81 \times \int_0^{10} 3\sqrt{y}\, dy \qquad \{\because x = 3\sqrt{y}\}$$

$$= 29430 \left[\frac{y^{3/2}}{3/2}\right]_0^{10} = 29430 \times \frac{2}{3}\left[y^{3/2}\right]_0^{10} = 19620\,[10^{3/2}]$$

$= 19620 \times 31.622 = 620439$ N

$\therefore$ Resultant water pressure on dam

$$F = \sqrt{F_x^2 + F_y^2} = \sqrt{(490500)^2 + (620439)^2}$$

$$= 790907 \text{ N} = \mathbf{790.907\ kN.\ Ans.}$$

Direction of the resultant is given by

$$\tan\theta = \frac{F_y}{F_x} = \frac{620439}{490500} = 1.265$$

$\therefore$ $\theta = \mathbf{51^\circ\ 40'.\ Ans.}$

Problem 3.30 *A dam has a parabolic shape* $y = y_0\left(\frac{x}{x_0}\right)^2$ *as shown in Fig. 3.38 below having* $x_0 = 6$ *m and* $y_0 = 9$ *m. The fluid is water with density = 1000 kg/m*^{3}*. Compute the horizontal, vertical and the resultant thrust exerted by water per metre length of the dam.*

Solution. Given :

Equation of the curve *OA* is

$$y = y_0\left(\frac{x}{x_0}\right)^2 = 9\left(\frac{x}{6}\right)^2 = 9 \times \frac{x^2}{36} = \frac{x^2}{4}$$

or $x^2 = 4y$

$\therefore$ $x = \sqrt{4y} = 2y^{1/2}$

Width of dam, $b = 1$ m.

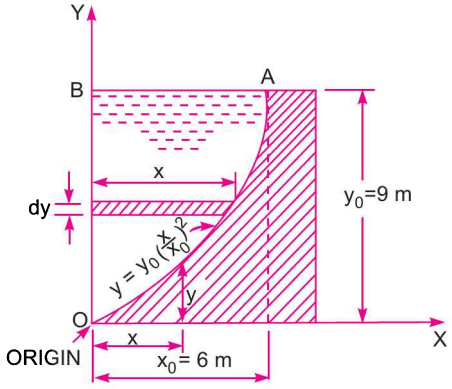


Fig. 3.38

(*i*) **Horizontal thrust exerted by water**

F_x = Force exerted by water on vertical surface *OB*, *i.e.*, the surface obtained by projecting the curved surface on vertical plane

$$= \rho g A \bar{h}$$

$$= 1000 \times 9.81 \times (9 \times 1) \times \frac{9}{2} = \mathbf{397305\ N.\ Ans.}$$

(*ii*) **Vertical thrust exerted by water**

F_y = Weight of water supported by curved surface *OA* upto free surface of water

= Weight of water in the portion *ABO*

= $\rho g \times$ Area of *OAB* $\times$ Width of dam

$$= 1000 \times 9.81 \times \left[\int_0^9 x \times dy\right] \times 1.0$$

$$= 1000 \times 9.81 \times \left[\int_0^9 2y^{1/2} \times dy\right] \times 1.0 \qquad (\because\ x = 2y^{1/2})$$

$$= 19620 \times \left[\frac{y^{3/2}}{(3/2)}\right]_0^9 = 19620 \times \frac{2}{3}\ [9^{3/2}]$$

$$= 19620 \times \frac{2}{3} \times 27 = \mathbf{353160\ N.\ Ans.}$$

(*iii*) **Resultant thrust exerted by water**

$$F = \sqrt{F_x^2 + F_y^2} = \sqrt{397305 + 353160} = \mathbf{531574\ N.\ Ans.}$$

Direction of resultant is given by

$$\tan\theta = \frac{F_y}{F_x} = \frac{353160}{397305} = 0.888$$

$$\theta = \tan^{-1} 0.888 = \mathbf{41.63°.\ Ans.}$$

Problem 3.31 *A cylinder 3 m in diameter and 4 m long retains water on one side. The cylinder is supported as shown in Fig. 3.39. Determine the horizontal reaction at A and the vertical reaction at B. The cylinder weighs 196.2 kN. Ignore friction.*

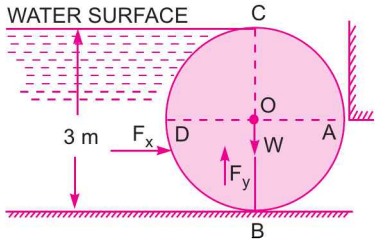


Fig. 3.39

Solution. Given :

Dia. of cylinder = 3 m

Length of cylinder = 4 m

Weight of cylinder, W = 196.2 kN = 196200 N

Horizontal force exerted by water

F_x = Force on vertical area BOC

$= \rho g A \bar{h}$

where $A = BOC \times l = 3 \times 4 = 12\ m^2$, $\bar{h} = \frac{1}{2} \times 3 = 1.5$ m

$F_x = 1000 \times 9.81 \times 12 \times 1.5 = 176580$ N

The vertical force exerted by water

F_y = Weight of water enclosed in $BDCOB$

$$= \rho g \times \left(\frac{\pi}{2} R^2\right) \times l = 1000 \times 9.81 \times \frac{\pi}{2} \times (1.5)^2 \times 4 = 138684\ \text{N}$$

Force F_y is acting in the upward direction.

For the equilibrium of cylinder

Horizontal reaction at $A = F_x = 176580$ N

Vertical reaction at B = Weight of cylinder – F_y

= 196200 – 138684 = **57516 N. Ans.**

▶ 3.7 TOTAL PRESSURE AND CENTRE OF PRESSURE ON LOCK GATES

Lock gates are the devices used for changing the water level in a canal or a river for navigation. Fig. 3.40 shows plan and elevation of a pair of lock gates. Let AB and BC be the two lock gates. Each gate is supported on two hinges fixed on their top and bottom at the ends A and C. In the closed position, the gates meet at B.

Let F = Resultant force due to water on the gate AB or BC acting are right angles to the gate

R = Reaction at the lower and upper hinge

P = Reaction at the common contact surface of the two gates and acting perpendicular to the contact surface.

Let the force P and F meet at O. Then the reaction R must pass through O as the gate AB is in the equilibrium under the action of three forces. Let θ is the inclination of the lock gate with the normal to the side of the lock.

In $\angle ABO$, $\angle OAB = \angle ABO = \theta$.

Resolving all forces along the gate AB and putting equal to zero, we get

$$R \cos\theta - P \cos\theta = 0 \text{ or } R = P \qquad ...(3.19)$$

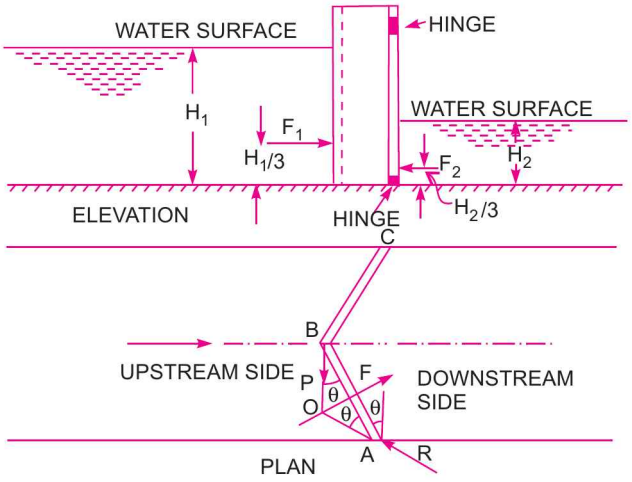


Fig. 3.40

Resolving forces normal to the gate AB

$$R \sin\theta + P \sin\theta - F = 0$$

or $$F = R \sin\theta + P \sin\theta = 2P \sin\theta \qquad \{\because R = P\}$$

$$\therefore \quad P = \frac{F}{2 \sin\theta} \qquad ...(3.20)$$

To calculate P and R

In equation (3.20), P can be calculated if F and θ are known. The value of θ is calculated from the angle between the lock gates. The angle between the two lock gate is equal to $180^\circ - 2\theta$. Hence θ can be calculated. The value of F is calculated as :

Let H_1 = Height of water on the upstream side

H_2 = Height of water on the downstream side

F_1 = Water pressure on the gate on upstream side

F_2 = Water pressure on the gate on downstream side of the gate

l = Width of gate

Now $$F_1 = \rho g A_1 \bar{h}_1$$

$$= \rho g \times H_1 \times l \times \frac{H_1}{2}$$

$$= \rho g l \frac{H_1^2}{2} \qquad \left[\because A_1 = H_1 \times l, \bar{h}_1 = \frac{H_1}{2}\right]$$

Similarly, $$F_2 = \rho g A_2 \bar{h}_2 = \rho g \times (H_2 \times l) \times \frac{H_2}{2} = \frac{\rho g l H_2^2}{2}$$

$\therefore$ Resultant force $$F = F_1 - F_2 = \frac{\rho g l H_1^2}{2} - \frac{\rho g l H_2^2}{2}$$

Substituting the value of θ and F in equation (3.20), the value of P and R can be calculated.

Reactions at the top and bottom hinges

Let R_t = Reaction of the top hinge

R_b = Reaction of the bottom hinge

Then $R = R_t + R_b$

The resultant water pressure F acts normal to the gate. Half of the value of F is resisted by the hinges of one lock gates and other half will be resisted by the hinges of other lock gate. Also F_1 acts at a distance of $\frac{H_1}{3}$ from bottom while F_2 acts at a distance of $\frac{H_2}{3}$ from bottom.

Taking moments about the lower hinge

$$R_t \times \sin\theta \times H = \frac{F_1}{2} \times \frac{H_1}{3} - \frac{F_2}{2} \times \frac{H_2}{3} \qquad ...(i)$$

where H = Distance between two hinges

Resolving forces horizontally

$$R_t \sin\theta + R_b \sin\theta = \frac{F_1}{2} - \frac{F_2}{2} \qquad ...(ii)$$

From equations (i) and (ii), we can find R_t and R_b.

Problem 3.32 *Each gate of a lock is 6 m high and is supported by two hinges placed on the top and bottom of the gate. When the gates are closed, they make an angle of 120°. The width of lock is 5 m. If the water levels are 4 m and 2 m on the upstream and downstream sides respectively, determine the magnitude of the forces on the hinges due to water pressure.*

Solution. Given :

Height of lock = 6 m

Width of lock = 5 m

Width of each lock gate = AB

or $$l = \frac{AD}{\cos 30°} = \frac{2.5}{\cos 30°}$$

$$= 2.887 \text{ m}$$

Angle between gates = 120°

$\therefore$ $$\theta = \frac{180° - 120°}{2} = \frac{60°}{2} = 30°$$

Height of water on upstream side

$H_1 = 4$ m

and $H_2 = 2$ m

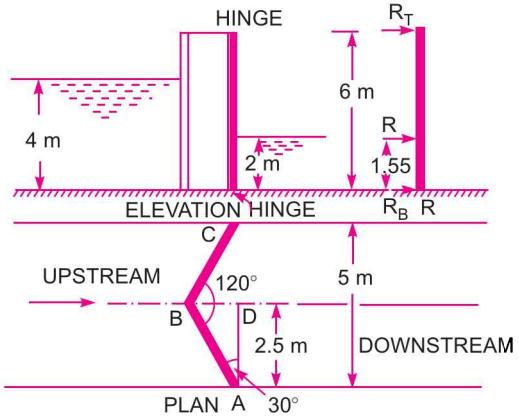


Fig. 3.41

$\therefore$ Total water pressure on upstream side

$$F_1 = \rho g A_1 \bar{h}_1, \text{ where } A_1 = H_1 \times l = 4.0 \times 2.887 \text{ m}^2$$

$$= 1000 \times 9.81 \times 4 \times 2.887 \times 2.0 \qquad \left\{\bar{h}_1 = \frac{H_1}{2} = \frac{4}{2} = 2.0 \text{ m}\right\}$$

$$= 226571 \text{ N}$$

Force F_1 will be acting at a distance of $\frac{H_1}{3} = \frac{4}{3} = 1.33$ m from bottom.

Similarly, total water pressure on the downstream side

$$F_2 = \rho g A_2 \bar{h}_2, \text{ where } A_2 = H_2 \times l = 2 \times 2.887 \text{ m}^2$$

$$= 1000 \times 9.81 \times 2 \times 2.887 \times 1.0 \qquad \bar{h}_2 = \frac{H_2}{2} = \frac{2}{2} = 1.0 \text{ m}$$

$= 56643$ N

F_2 will act at a distance of $\frac{H_2}{3} = \frac{2}{3} = 0.67$ m from bottom,

Resultant water pressure on each gate

$$F = F_1 - F_2 = 226571 - 56643 = 169928 \text{ N.}$$

Let x is height of F from the bottom, then taking moments of F_1, F_2 and F about the bottom, we have

$$F \times x = F_1 \times 1.33 - F_2 \times 0.67$$

or $$169928 \times x = 226571 \times 1.33 - 56643 \times 0.67$$

$$\therefore \quad x = \frac{226571 \times 1.33 - 56643 \times 0.67}{169928} = \frac{301339 - 37950}{169928} = 1.55 \text{ m}$$

From equation (3.20), $P = \frac{F}{2 \sin \theta} = \frac{169928}{2 \sin 30} = 169928$ N.

From equation (3.19), $R = P = 169928$ N.

If R_T and R_B are the reactions at the top and bottom hinges, then $R_T + R_B = R = 169928$ N.

Taking movements of hinge reactions R_T, R_B and R about the bottom hinges, we have

$$R_T \times 6.0 + R_B \times 0 = R \times 1.55$$

$$\therefore \quad R_T = \frac{169928 \times 1.55}{6.0} = 43898 \text{ N}$$

$$\therefore \quad R_B = R - R_T = 169928 - 43898 = \textbf{126030 N. Ans.}$$

Problem 3.33 *The end gates ABC of a lock are 9 m high and when closed include an angle of 120°. The width of the lock is 10 m. Each gate is supported by two hinges located at 1 m and 6 m above the bottom of the lock. The depths of water on the two sides are 8 m and 4 m respectively. Find:*

(i) Resultant water force on each gate,

(ii) Reaction between the gates AB and BC, and

(iii) Force on each hinge, considering the reaction of the gate acting in the same horizontal plane as resultant water pressure.

Solution. Given :

Height of gate $= 9$ m

Inclination of gate $= 120°$

$$\therefore \quad \theta = \frac{180° - 120°}{2} = 30°$$

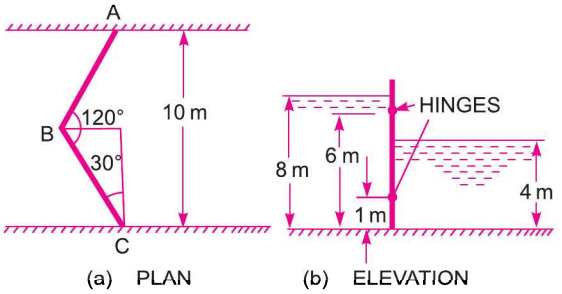


Fig. 3.42

Width of lock = 10 m

$\therefore$ Width of each lock $= \dfrac{5}{\cos 30^\circ}$ or $l = 5.773$ m

Depth of water on upstream side, $H_1 = 8$ m

Depth of water on downstream side, $H_2 = 4$ m

(*i*) **Water pressure on upstream side**

$$F_1 = \rho g A_1 \bar{h}_1$$

where $A_1 = l \times H_1 = 5.773 \times 8 = 46.184$ m, $\bar{h}_1 = \dfrac{H_1}{2} = \dfrac{8}{2} = 4.0$ m

$$F_1 = 1000 \times 9.81 \times 46.184 \times 4.0 = 1812260 \text{ N} = 1812.26 \text{ kN}$$

Water pressure on downstream side,

$$F_2 = \rho g A_2 \bar{h}_2$$

where $A_2 = l \times H_2 = 5.773 \times 4 = 23.092$ m, $\bar{h}_2 = \dfrac{4}{2} = 2.0$

$$F_2 = 1000 \times 9.81 \times 23.092 \times 2.0 = 453065 \text{ N} = 453.065 \text{ kN}$$

$\therefore$ Resultant water pressure

$$= F_1 - F_2 = 1812.26 - 453.065 = 1359.195 \text{ kN}$$

(*ii*) **Reaction between the gates AB and BC.** The reaction (P) between the gates AB and BC is given by equation (3.20) as

$$F = \frac{F}{2 \sin \theta} = \frac{1359.195}{2 \times \sin 30^\circ} = \textbf{1359.195 kN. Ans.}$$

(*iii*) **Force on each hinge.** If R_T and R_B are the reactions at the top and bottom hinges then

$$R_T + R_B = R$$

But from equation (3.19), $R = P = 1359.195$

$\therefore$ $R_T + R_B = 1359.195$

The force F_1 is acting at $\dfrac{H_1}{3} = \dfrac{8}{3} = 2.67$ m from bottom and F_2 at $\dfrac{H_2}{3} = \dfrac{4}{3} = 1.33$ m from bottom.

The resultant force F will act at a distance x from bottom is given by

$$F \times x = F_1 \times 2.67 - F_2 \times 1.33$$

or
$$x = \frac{F_1 \times 2.67 - F_2 \times 1.33}{F} = \frac{1812.26 \times 2.67 - 453.065 \times 1.33}{1359.195}$$

$$= \frac{4838.734 - 602.576}{1359.195} = 3.116 = 3.11 \text{ m}$$

Hence R is also acting at a distance 3.11 m from bottom.

Taking moments of R_T and R about the bottom hinge

$$R_T \times [6.0 - 1.0] = R \times (x - 1.0)$$

$\therefore$
$$R_T = \frac{R \times (x - 1.0)}{5.0} = \frac{1359.195 \times 2.11}{5.0} = 573.58 \text{ N}$$

$\therefore$
$$R_B = R - R_T = 1359.195 - 573.58$$

$$= \textbf{785.615 kN. Ans.}$$

▶ 3.8 PRESSURE DISTRIBUTION IN A LIQUID SUBJECTED TO CONSTANT HORIZONTAL/VERTICAL ACCELERATION

In chapters 2 and 3, the containers which contains liquids, are assumed to be at rest. Hence the liquids are also at rest. They are in static equilibrium with respect to containers. But if the container containing a liquid is made to move with a constant acceleration, the liquid particles initially will move relative to each other and after some time, there will not be any relative motion between the liquid particles and boundaries of the container. The liquid will take up a new position under the effect of acceleration imparted to its container. The liquid will come to rest in this new position relative to the container. The entire fluid mass moves as a single unit. Since the liquid after attaining a new position is in static condition relative to the container, the laws of hydrostatic can be applied to determine the liquid pressure. As there is no relative motion between the liquid particles, hence the shear stresses and shear forces between liquid particles will be zero. The pressure will be normal to the surface in contact with the liquid.

The following are the important cases under consideration :

(*i*) Liquid containers subject to constant horizontal acceleration.

(*ii*) Liquid containers subject to constant vertical acceleration.

3.8.1 Liquid Containers Subject to Constant Horizontal Acceleration. Fig. 3.43 (*a*) shows a tank containing a liquid upto a certain depth. The tank is stationary and free surface of liquid is horizontal. Let this tank is moving with a constant acceleration '*a*' in the horizontal direction towards right as shown in Fig. 3.43 (*b*). The initial free surface of liquid which was horizontal, now takes the shape as shown in Fig. 3.43 (*b*). Now *AB* represents the new free surface of the liquid. Thus the free surface of liquid due to horizontal acceleration will become a downward sloping inclined plane, with the liquid rising at the back end, the liquid falling at the front end. The equation for the free liquid surface can be derived by considering the equilibrium of a fluid element C lying on the free surface. The forces acting on the element *C* are :

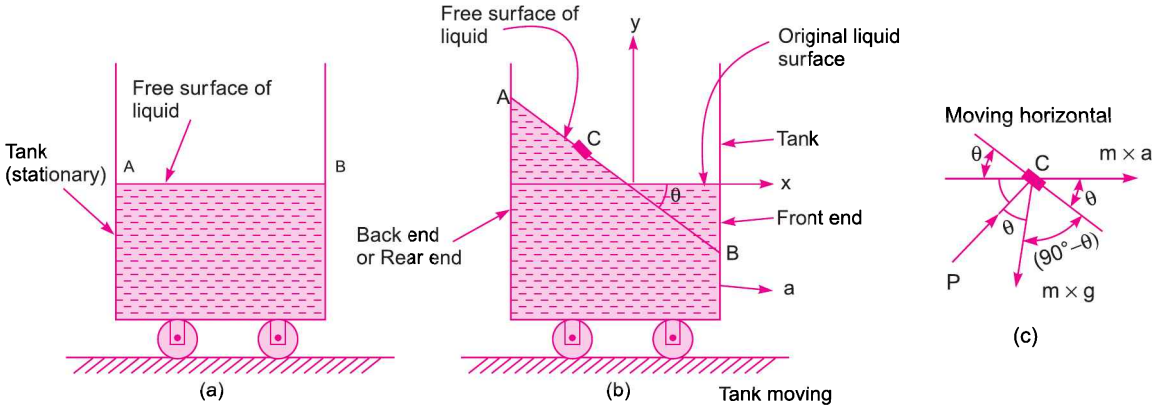


Fig. 3.43

(*i*) the pressure force *P* exerted by the surrounding fluid on the element *C*. This force is normal to the free surface.

(*ii*) the weight of the fluid element *i.e.*, $m \times g$ acting vertically downward.

(*iii*) accelerating force *i.e.*, $m \times a$ acting in horizontal direction.

Resolving the forces horizontally, we get

$$P \sin \theta + m \times a = 0$$

or $$P \sin \theta = - ma \qquad ...(i)$$

Resolving the forces vertically, we get

$$P \cos \theta - mg = 0$$

or $$P \cos \theta = m \times g \qquad ...(ii)$$

Dividing (i) by (ii), we get

$$\tan \theta = - \frac{a}{g} \quad \left(\text{or } \frac{a}{g} \text{ Numerically} \right) \qquad ...(3.20A)$$

The above equation, gives the slope of the free surface of the liquid which is contained in a tank which is subjected to horizontal constant acceleration. The term (a/g) is a constant and hence tan θ will be constant. The –ve sign shows that the free surface of liquid is sloping downwards. Hence the free surface is a straight plane inclined down at an angle θ along the direction of acceleration.

Now let us find the expression for the pressure at any point D in the liquid mass subjected to horizontal acceleration. Let the point D is at a depth of 'h' from the free surface. Consider an elementary prism DE of height 'h' and cross-sectional area dA as shown in Fig. 3.44.

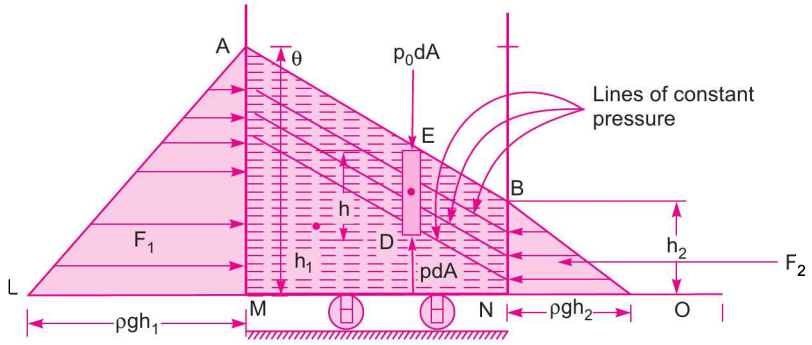


Fig. 3.44

Consider the equilibrium of the elementary prism DE.

The forces acting on this prism DE in the vertical direction are :

(i) the atmospheric pressure force $(p_0 \times dA)$ at the top end of the prism acting downwards,

(ii) the weight of the element $(\rho \times g \times h \times dA)$ at the C.G. of the element acting in the downward direction, and

(iii) the pressure force $(p \times dA)$ at the bottom end of the prism acting upwards.

Since there is no vertical acceleration given to the tank, hence net force acting vertically should be zero.

$$\therefore \qquad p \times dA - p_0 \times dA - \rho g h \, dA = 0$$

or $$p - p_0 - \rho g h = 0 \quad \text{or} \quad p = p_0 + \rho g h$$

or $$p - p_0 = \rho g h$$

or gauge pressure at point D is given by

$$p = \rho g h$$

or pressure head at point D, $$\frac{p}{\rho g} = h.$$

From the above equation, it is clear that pressure head at any point in a liquid subjected to a constant horizontal acceleration is equal to the height of the liquid column above that point. Therefore the pressure distribution in a liquid subjected to a constant horizontal acceleration is same as hydrostatic pressure distribution. The planes of constant pressure are therefore, parallel to the inclined surface as shown in Fig. 3.44. This figure also shows the variation of pressure on the rear and front end of the tank.

If h_1 = Depth of liquid at the rear end of the tank

h_2 = Depth of liquid at the front end of the tank

F_1 = Total pressure force exerted by liquid on the rear side of the tank

F_2 = Total pressure force exerted by liquid on the front side of the tank,

then F_1 = (Area of triangle AML) × Width

$$= (\tfrac{1}{2} \times LM \times AM \times b) = \tfrac{1}{2} \times \rho g h_1 \times h_1 \times b = \frac{\rho g . b . h_1^2}{2}$$

and F_2 = (Area of triangle BNO) × Width

$$= (\tfrac{1}{2} \times BN \times NO) = \tfrac{1}{2} \times h_2 \times \rho g h_2 \times b = \frac{\rho g . b . h_2^2}{2}$$

where b = Width of tank perpendicular to the plane of the paper.

The values of F_1 and F_2 can also be obtained as

[Refer to Fig. 3.44 (*a*)]

$$F_1 = \rho \times g \times A_1 \times \bar{h}_1, \text{ where } A_1 = h_1 \times b \text{ and } \bar{h}_1 = \frac{h_1}{2}$$

$$= \rho \times g \times (h_1 \times b) \times \frac{h_1}{2} = \frac{1}{2} \rho g . b . h_1^2$$

and $F_2 = \rho \times g \times A_2 \times \bar{h}_2$, where $A_2 = h_2 \times b$ and $\bar{h}_2 = \dfrac{h_2}{2}$

$$= \rho \times g \times (h_2 \times b) \times \frac{h_2}{2}$$

$$= \frac{1}{2} \rho g . b \times h_2^2.$$

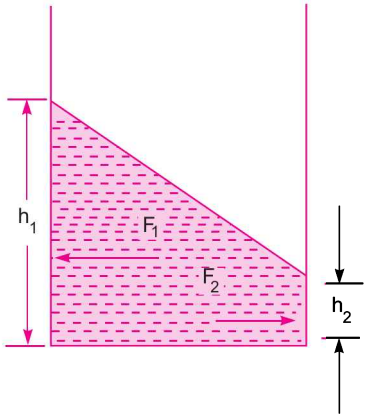


Fig. 3.44(*a*)

It can also be proved that the difference of these two forces (*i.e.*, $F_1 - F_2$) is equal to the force required to accelerate the mass of the liquid contained in the tank *i.e.*,

$$F_1 - F_2 = M \times a$$

where M = Total mass of the liquid contained in the tank

a = Horizontal constant acceleration.

Note : (*i*) If a tank completely filled with liquid and open at the top is subjected to a constant horizontal acceleration, then some of the liquid will spill out from the tank and new free surface with its slope given by equation $\tan \theta = -\dfrac{a}{g}$ will be developed.

(*ii*) If a tank partly filled with liquid and open at the top is subjected to a constant horizontal acceleration, spilling of the liquid may take place depending upon the magnitude of the acceleration.

(*iii*) If a tank completely filled with liquid and closed at the top is subjected to a constant horizontal acceleration, then the liquid would not spill out from the tank and also there will be no adjustment in the surface elevation of the liquid. But the equation $\tan \theta = -\dfrac{a}{g}$ is applicable for this case also.

(*iv*) The example for a tank with liquid subjected to a constant horizontal acceleration, is a fuel tank on an airplane during take off.

Problem 3.34 *A rectangular tank is moving horizontally in the direction of its length with a constant acceleration of 2.4 m/s^2. The length, width and depth of the tank are 6 m, 2.5 m and 2 m respectively. If the depth of water in the tank is 1 m and tank is open at the top then calculate :*

(*i*) *the angle of the water surface to the horizontal,*

(*ii*) *the maximum and minimum pressure intensities at the bottom,*

(*iii*) *the total force due to water acting on each end of the tank.*

Solution. Given :

Constant acceleration, $a = 2.4$ m/s^2.

Length = 6 m ; Width = 2.5 m and depth = 2 m.

Depth of water in tank, $h = 1$ m

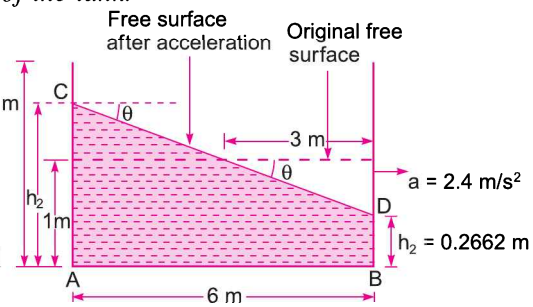


Fig. 3.45

(*i*) **The angle of the water surface to the horizontal**

Let θ = the angle of water surface to the horizontal

Using equation (3.20), we get

$$\tan\theta = -\frac{a}{g} = -\frac{2.4}{9.81} = -0.2446$$

(the –ve sign shows that the free surface of water is sloping downward as shown in Fig. 3.45)

$\therefore$ $\tan\theta = 0.2446$ (slope downward)

$\therefore$ $\theta = \tan^{-1} 0.2446 =$ **13.7446°** or **13° 44.6′. Ans.**

(*ii*) **The maximum and minimum pressure intensities at the bottom of the tank**

From the Fig. 3.45,

Depth of water at the front end,

$$h_1 = 1 - 3\tan\theta = 1 - 3 \times 0.2446 = 0.2662 \text{ m}$$

Depth of water at the rear end,

$$h_2 = 1 + 3\tan\theta = 1 + 3 \times 0.2446 = 1.7338 \text{ m}$$

The pressure intensity will be maximum at the bottom, where depth of water is maximum.

Now the maximum pressure intensity at the bottom will be at point A and it is given by,

$$p_{max} = \rho \times g \times h_2$$
$$= 1000 \times 9.81 \times 1.7338 \text{ N/m}^2 = \mathbf{17008.5 \text{ N/m}^2}. \textbf{ Ans.}$$

The minimum pressure intensity at the bottom will be at point B and it is given by

$$p_{min} = \rho \times g \times h_1$$
$$= 1000 \times 9.81 \times 0.2662 = \mathbf{2611.4 \text{ N/m}^2}. \textbf{ Ans.}$$

(*iii*) **The total force due to water acting on each end of the tank**

Let F_1 = total force acting on the front side (*i.e.*, on face BD)

F_2 = total force acting on the rear side (*i.e.*, on face AC)

Then $F_1 = \rho g A_1 \bar{h}_1$, where $A_1 = BD \times$ width of tank $= h_1 \times 2.5 = 0.2662 \times 2.5$

and $\bar{h}_1 = \frac{BD}{2} = \frac{h_1}{2} = \frac{0.2662}{2} = 0.1331$ m

$$= 1000 \times 9.81 \times (0.2662 \times 2.5) \times 0.1331$$
$$= \mathbf{868.95 \text{ N}}. \textbf{ Ans.}$$

and $F_2 = \rho.g.A_2.\bar{h}_2$, where $A_2 = AB \times$ width of tank $= h_2 \times 2.5 = 1.7338 \times 2.5$

$$\bar{h}_2 = \frac{AB}{2} = \frac{h_2}{2} = \frac{1.7338}{2} = 0.8669 \text{ m}$$

$$= 1000 \times 9.81 \times (1.7338 \times 2.5) \times 0.8669$$

$$= \mathbf{36861.8\ N.\ Ans.}$$

$\therefore$ Resultant force $= F_1 - F_2$

$$= 36861.8 \text{ N} - 868.95$$

$$= 35992.88 \text{ N}$$

Note. The difference of the forces acting on the two ends of the tank is equal to the force necessary to accelerate the liquid mass. This can be proved as shown below :

Consider the control volume of the liquid *i.e.*, control volume is *ACDBA* as shown in Fig. 3.46. The net force acting on the control volume in the horizontal direction must be equal to the product of mass of the liquid in control volume and acceleration of the liquid.

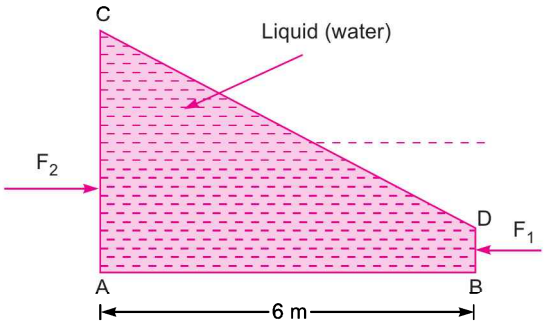


Fig. 3.46

$\therefore$ $(F_1 - F_2) = M \times a$

$$= (\rho \times \text{volume of control volume}) \times a$$

$$= (1000 \times \text{Area of } ABDCE \times \text{width}) \times 2.4$$

$$= \left[1000 \times \left(\frac{AC + BD}{2}\right) \times AB \times \text{width}\right] \times 2.4$$

$$\left[\because \text{Area of trapezium} = \left(\frac{AC + BD}{2}\right) \times AB\right]$$

$$= 1000 \times \left(\frac{1.7338 + 0.2662}{2}\right) \times 6 \times 2.5 \times 2.4$$

$$= 36000 \text{ N}$$

($\because AC = h_2 = 1.7338$ m, $BD = h_1 = 0.2662$ m, and $AB = 6$ m, width $= 2.5$ m)

The above force is nearly the same as the difference of the forces acting on the two ends of the tank. (*i.e.*, $35992.88 \simeq 36000$).

Problem 3.35 *The rectangular tank of the above problem contains water to a depth of 1.5 m. Find the horizontal acceleration which may be imparted to the tank in the direction of its length so that*

(*i*) *the spilling of water from the tank is just on the verge of taking place,*

(*ii*) *the front bottom corner of the tank is just exposed,*

(*iii*) *the bottom of the tank is exposed upto its mid-point.*

Also calculate the total forces exerted by the water on each end of the tank in each case. Also prove that the difference between these forces is equal to the force necessary to accelerate the mass of water tank.

Solution. Given :

Dimensions of the tank from previous problem,

$L = 6$ m, width (b) $= 2.5$ m and depth $= 2$ m

Depth of water in tank, $h = 1.5$ m

Horizontal acceleration imparted to the tank

(*i*) (*a*) *When the spilling of water from the tank is just on the verge of taking place*

Let a = required horizontal acceleration

When the spilling of water from the tank is just on the verge of taking place, the water would rise upto the rear top corner of the tank as shown in Fig. 3.47 (*a*)

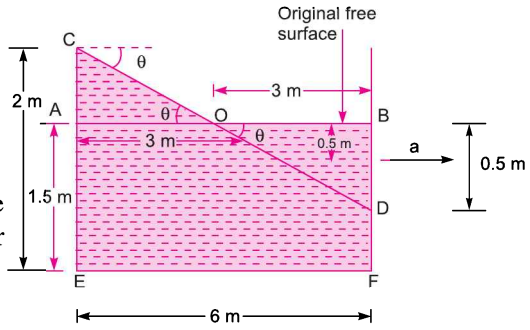


Fig. 3.47 (*a*) *Spilling of water is just on the verge of taking place.*

$$\therefore \qquad \tan\theta = \frac{AC}{AO} = \frac{(2-1.5)}{3} = \frac{0.5}{3} = 0.1667$$

But from equation (3.20) $\tan\theta = \dfrac{a}{g}$ (Numerically)

$$\therefore \qquad a = g \times \tan\theta = 9.81 \times 0.1667 = \mathbf{1.635\ m/s^2.}\ \textbf{Ans.}$$

(*b*) *Total forces exerted by water on each end of the tank*

The force exerted by water on the end CE of the tank is

$$F_1 = \rho g A_1 \bar{h}_1, \text{ where } A_1 = CE \times \text{width of the tank} = 2 \times 2.5$$

$$\bar{h}_1 = \frac{CE}{2} = \frac{2}{2} = 1 \text{ m}$$

$$= 1000 \times 9.81 \times (2 \times 2.5) \times 1$$

$$= \mathbf{49050\ N.}\ \textbf{Ans.}$$

The force exerted by water on the end FD of the tank is

$$F_2 = \rho g A_2 \times \bar{h}_2, \text{ where } A_2 = FD \times \text{width} = 1 \times 2.5$$

$$(\because AC = BD = 0.5 \text{ m}, \quad \therefore \quad FD = BF - BD = 1.5 - 0.5 = 1)$$

$$= 1000 \times 9.81 \times (1 \times 2.5) \times 0.5 \qquad \bar{h}_2 = \frac{FD}{2} = \frac{1}{2} = 0.5 \text{ m}$$

$$= \mathbf{12262.5\ N.}\ \textbf{Ans.}$$

(*c*) *Difference of the forces is equal to the force necessary to accelerate the mass of water in the tank*

Difference of the forces $= F_1 - F_2$

$$= 49050 - 12262.5 = 36787.5 \text{ N}$$

Volume of water in the tank before acceleration is imparted to it $= L \times b \times$ depth of water

$$= 6 \times 2.5 \times 1.5 = 22.5 \text{ m}^3.$$

The force necessary to accelerate the mass of water in the tank

$$= \text{Mass of water in tank} \times \text{Acceleration}$$

$$= (\rho \times \text{volume of water}) \times 1.635 \qquad (\because a = 1.635 \text{ m/s}^2)$$

$$= 1000 \times 22.5 \times 1.635 \text{ [There is no spilling of water and volume of water} = 22.5 \text{ m}^3\text{]}$$

$$= 36787.5 \text{ N}$$

Hence the difference between the forces on the two ends of the tank is equal to the force necessary to accelerate the mass of water in the tank.

Volume of water in the tank can also be calculated as volume $= \left(\frac{CE + FD}{2}\right) \times EF \times$ Width [Refer to Fig. 3.47 (*a*)]

$$= \left(\frac{2+1}{2}\right) \times 6 \times 2.5 = 22.5 \text{ m}^3.$$

(*ii*) (*a*) *Horizontal acceleration when the front bottom corner of the tank is just exposed*

Refer to Fig. 3.47 (*b*). In this case the free surface of water in the tank will be along *CD*.

Let a = required horizontal acceleration.

In this case, $\tan\theta = \frac{CE}{ED} = \frac{2}{6} = \frac{1}{3}$

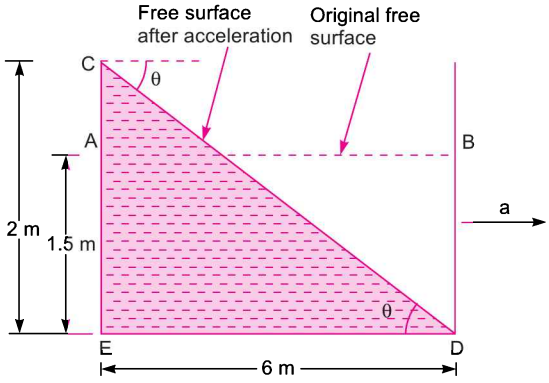


Fig. 3.47 (*b*)

But from equation (3.17),

$$\tan\theta = \frac{a}{g} \text{ (Numerically)}$$

$$\therefore \quad a = g \times \tan\theta = 9.81 \times \frac{1}{3} = \mathbf{3.27 \text{ m/s}^2.} \textbf{ Ans.}$$

(*b*) *Total forces exerted by water on each end of the tank*

The force exerted by water on the end *CE* of the tank is

$$F_1 = \rho g \times A_1 \times \bar{h}_1$$

where $A_1 = CE \times \text{width} = 2 \times 2.5 = 5 \text{ m}^2$

$$\bar{h}_1 = \frac{CE}{2} = \frac{2}{2} = 1 \text{ m} \qquad = 1000 \times 9.81 \times 5 \times 1$$

$$= \mathbf{49050 \text{ N.}} \textbf{ Ans.}$$

The force exerted by water on the end *BD* of the tank is zero as there is no water against the face *BD*

$$\therefore \quad F_2 = 0$$

$$\therefore \quad \text{Difference of forces} = 49050 - 0 = 49050 \text{ N}$$

(*c*) *Difference of forces is equal to the force necessary to accelerate the mass of water in the tank.*

Volume of water in the tank = Area of *CED* × Width of tank

$$= \left(\frac{CE \times ED}{2}\right) \times 2.5 \qquad (\because \text{ Width of tank} = 2.5 \text{ m})$$

$$= \frac{2 \times 6}{2} \times 2.5 = 15 \text{ m}^3$$

$\therefore$ Force necessary to accelerate the mass of water in the tank

= Mass of water in tank × Acceleration

= (1000 × Volume of water) × 3.27

= 1000 × 15 × 3.27 = 49050 N

Difference of two forces is also = 49050 N

Hence difference between the forces on the two ends of the tank is equal to the force necessary to accelerate the mass of water in the tank.

(iii) (*a*) *Horizontal acceleration when the bottom of the tank is exposed upto its mid-point*

Refer to Fig. 3.47 (*c*). In this case the free surface of water in the tank will be along *CD**, where *D** is the mid-point of *ED*.

Let a = required horizontal acceleration from Fig. 3.47 (*c*), it is clear that

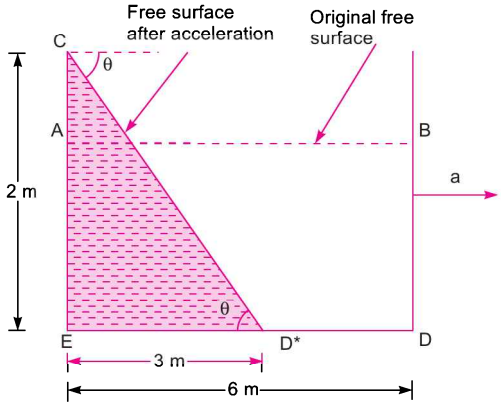


Fig. 3.47 (*c*)

$$\tan\theta = \frac{CE}{ED^*} = \frac{2}{3}$$

But from equation (3.20) numerically

$$\tan\theta = \frac{a}{g}$$

$$\therefore \quad a = g \times \tan\theta = 9.81 \times \frac{2}{3} = \mathbf{6.54\ m/s^2.\ Ans.}$$

(*b*) *Total forces exerted by water on each end of the tank*

The force exerted by water on the end *CE* of the tank is

$$F_1 = \rho \times g \times A_1 \times \bar{h}_1$$

where $A_1 = CE \times \text{Width} = 2 \times 2.5 = 5\ \text{m}^2$

$$\bar{h}_1 = \frac{CE}{2} = \frac{2}{2} = 1\ \text{m}$$

$$= 1000 \times 9.81 \times 5 \times 1$$

$$= \mathbf{49050\ N.\ Ans.}$$

The force exerted by water on the end *BD* is zero as there is no water against the face *BD*.

$$\therefore \quad F_2 = 0$$

$\therefore$ Difference of the forces $= F_1 - F_2 = 49050 - 0 = 49050$ N

(*c*) *Difference of the two forces is equal to the force necessary to accelerate the mass of water remaining in the tank*

Volume of water in the tank = Area *CED** × Width of tank

$$= \frac{CE \times ED^*}{2} \times 2.5 = \frac{2 \times 3}{2} \times 2.5 = 7.5\ \text{m}^3$$

Force necessary to accelerate the mass of water in the tank

$$= \text{Mass of water} \times \text{Acceleration}$$

$$= \rho \times \text{Volume of water} \times 6.54 \qquad (\because a = 6.54\ \text{m/s}^2)$$

$$= 1000 \times 7.5 \times 6.54$$

$$= 49050\ \text{N}$$

This is the same force as the difference of the two forces on the two ends of the tank.

Problem 3.36 *A rectangular tank of length 6 m, width 2.5 m and height 2 m is completely filled with water when at rest. The tank is open at the top. The tank is subjected to a horizontal constant linear acceleration of 2.4 m/s² in the direction of its length. Find the volume of water spilled from the tank.*

Solution. Given :

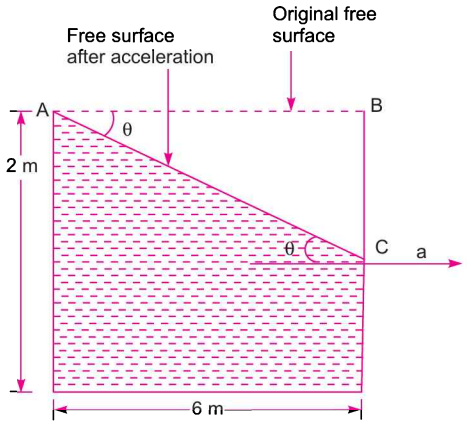
$L = 6$ m, $b = 2.5$ m and height, $H = 2$ m

Horizontal acceleration, $a = 2.4$ m/s^2.

The slope of the free surface of water after the tank is subjected to linear constant acceleration is given by equation (3.20) as

$$\tan\theta = \frac{a}{g} \text{ (Numerically)}$$

$$= \frac{2.4}{9.81} = 0.2446$$



From Fig. 3.48,

$$\tan\theta = \frac{BC}{AB}$$

$\therefore$ $BC = AB \times \tan\theta$

$= 6 \times 0.2446$

($\because$ AB = Length = 6 m ; $\tan\theta = 0.2446$) Fig. 3.48

$= 1.4676$ m

$\therefore$ Volume of water spilled = Area of $ABC \times$ Width of tank

$= \left(\frac{1}{2} \times AB \times BC\right) \times 2.5$ ($\because$ Width = 2.5 m)

$= \frac{1}{2} \times 6 \times 1.4676 \times 2.5$ ($\because$ $BC = 1.4676$ m)

$= 11.007$ m^3. Ans.

3.8.2 Liquid Container Subjected to Constant Vertical Acceleration. Fig. 3.49 shows a tank containing a liquid and the tank is moving vertically upward with a constant acceleration. The liquid in the tank will be subjected to the same vertical acceleration. To obtain the expression for the pressure at any point in the liquid mass subjected to vertical upward acceleration, consider a vertical elementary prism of liquid *CDFE*.

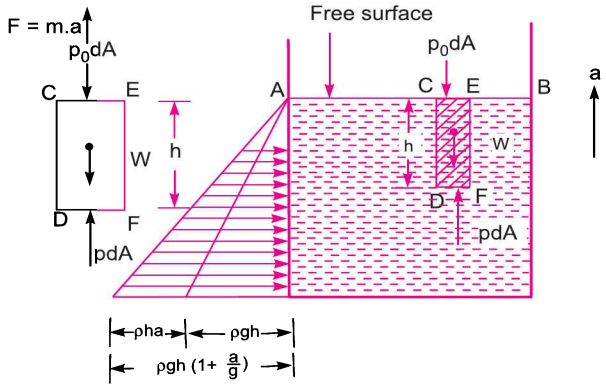


Fig. 3.49

Let dA = Cross-sectional area of prism

h = Height of prism

p_0 = Atmospheric pressure acting on the face *CE*

p = Pressure at a depth h acting on the face *DF*

The forces acting on the elementary prism are :

(*i*) Pressure force equal to $p_0 \times dA$ acting on the face *CE* vertically downward

(*ii*) Pressure force equal to $p \times dA$ acting on the face *DF* vertically upward

(*iii*) Weight of the prism equal to $\rho \times g \times dA \times h$ acting through C.G. of the element vertically downward.

According to Newton's second law of motion, the net force acting on the element must be equal to mass multiplied by acceleration in the same direction.

$\therefore$ Net force in vertically upward direction = Mass × acceleration

$$p \times dA - p_0 \times dA - \rho g dA \,.\, h = (\rho \times dA \times h) \times a \qquad (\because \text{Mass} = \rho \times dA \times h)$$

or $$p - p_0 - \rho g h = \rho h \times a \qquad \text{(Cancelling } dA \text{ from both sides)}$$

or $$p - p_0 = \rho g h + \rho h a$$

$$= \rho g h \left[1 + \frac{a}{g}\right] \qquad ...(3.21)$$

But $(p - p_0)$ is the gauge pressure. Hence gauge pressure at any point in the liquid mass subjected to a constant vertical upward acceleration, is given by

$$p_g = \rho g h \left[1 + \frac{a}{g}\right] \qquad ...(3.22)$$

$$= \rho g h + \rho h a \qquad ...(3.22A)$$

where $p_g = p - p_0$ = gauge pressure

In equation (3.22) ρ, g and a are constant. Hence variation of guage pressure is linear. Also when $h = 0$, $p_g = 0$. This means $p - p_0 = 0$ or $p = p_0$. Hence when $h = 0$, the pressure is equal to atmospheric pressure. Hence free surface of liquid subjected to constant vertical acceleration will be horizontal.

From equation (3.22*A*) it is also clear that the pressure at any point in the liquid mass is greater than the hydrostatic pressure (hydrostatic pressure is = $\rho g h$) by an amount of $\rho \times h \times a$.

Fig. 3.49 shows the variation of pressure for the liquid mass subjected to a constant vertical upward acceleration.

If the tank containing liquid is moving vertically downward with a constant acceleration, then the gauge pressure at any point in the liquid at a depth of h from the free surface will be given by

$$(p - p_0) = \rho g h \left[1 - \frac{a}{g}\right] = \rho g h - \rho h a \qquad ...(3.23)$$

The above equation shows that the pressure at any point in the liquid mass is less than the hydrostatic pressure by an amount of $\rho h a$. Fig. 3.50 shows the variation of pressure for the liquid mass subjected to a constant vertical downward acceleration.

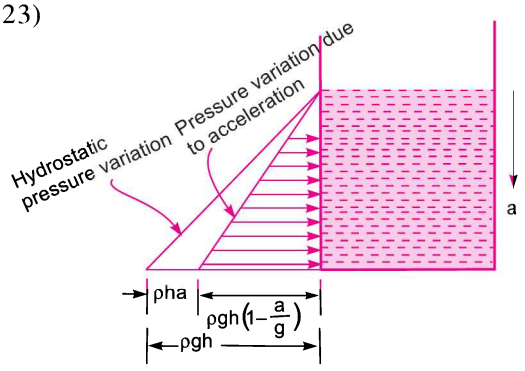


Fig. 3.50

If the tank containing liquid is moving downward with a constant acceleration equal to g (*i.e.,* when $a = g$), then equation reduces to $p - p_0 = 0$ or $p = p_0$. This means the pressure at any point in the liquid is equal to surrounding atmospheric pressure. There will be no force on the walls or on the base of the tank.

Note. If a tank containing a liquid is subjected to a constant acceleration in the inclined direction, then the acceleration may be resolved along the horizontal direction and vertical direction. Then each of these cases may be separately analysed in accordance with the above procedure.

Problem 3.37 *A tank containing water upto a depth of 500 mm is moving vertically upward with a constant acceleration of 2.45 m/s^2. Find the force exerted by water on the side of the tank. Also calculate the force on the side of the tank when the width of tank is 2 m and*

(i) tank is moving vertically downward with a constant acceleration of 2.45 m/s^2, and

(ii) the tank is not moving at all.

Solution. Given :

Depth of water, $h = 500 \text{ mm} = 0.5 \text{ m}$

Vertical acceleration, $a = 2.45 \text{ m/s}^2$

Width of tank, $b = 2 \text{ m}$

To find the force exerted by water on the side of the tank when moving vertically upward, let us first find the pressure at the bottom of the tank.

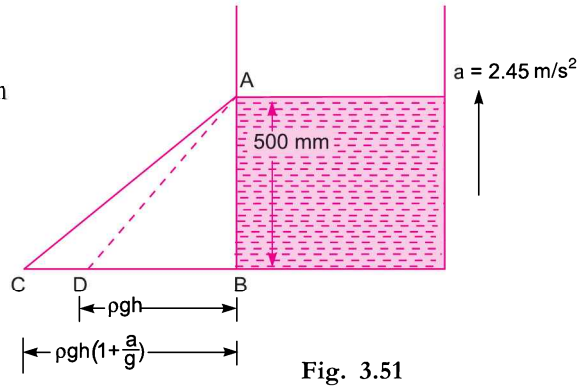


Fig. 3.51

The gauge pressure at the bottom (*i.e.*, at point *B*) for this case is given by equation as

$$p_B = \rho g h \left(1 + \frac{a}{g}\right)$$

$$= 1000 \times 9.81 \times 0.5 \left(1 + \frac{2.45}{9.81}\right) = 6131.25 \text{ N/m}^2$$

This pressure is represented by line *BC*.

Now the force on the side *AB* = Area of triangle *ABC* × Width of tank

$$= \left(\tfrac{1}{2} \times AB \times BC\right) \times b$$

$$= \left(\tfrac{1}{2} \times 0.5 \times 6131.25\right) \times 2 \qquad (\because BC = 6131.25 \text{ and } b = 2 \text{ m})$$

= 3065.6 N. Ans.

(*i*) **Force on the side of the tank, when tank is moving vertically downward.**

The pressure variation is shown in Fig. 3.52. For this case, the pressure at the bottom of the tank (*i.e.*, at point *B*) is given by equation (3.23) as

$$p_B = \rho g h \left(1 - \frac{a}{g}\right)$$

$$= 1000 \times 9.81 \times 0.5 \left(1 - \frac{2.45}{9.81}\right)$$

$$= 3678.75 \text{ N/m}^2$$

This pressure is represented by line *BC*.

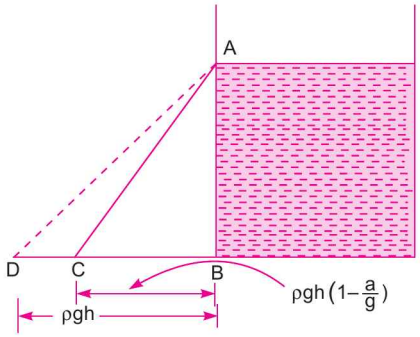


Fig. 3.52

Now the force on the side *AB* = Area of triangle *ABC* × Width

$$= \left(\tfrac{1}{2} \times AB \times BC\right) \times b$$

$$= \left(\tfrac{1}{2} \times 0.5 \times 3678.75\right) \times 2 \qquad (\because BC = 3678.75, b = 2)$$

= 1839.37 N. Ans.

(*ii*) **Force on the side of the tank, when tank is stationary.**

The pressure at point B is given by,

$$p_B = \rho g h = 1000 \times 9.81 \times 0.5 = 4905 \text{ N/m}^2$$

This pressure is represented by line BD in Fig. 3.52

Force on the side AB = Area of triangle $ABD \times$ Width

$$= \left(\frac{1}{2} \times AB \times BD\right) \times b$$

$$= \left(\frac{1}{2} \times 0.5 \times 4905\right) \times 2 \qquad (\because BD = 4905)$$

= 2452.5 N. Ans.

For this case, the force on AB can also be obtained as

$$F_{AB} = \rho g A.\bar{h}$$

where $A = AB \times \text{Width} = 0.5 \times 2 = 1 \text{ m}^2$

$$\bar{h} = \frac{AB}{2} = \frac{0.5}{2} = 0.25 \text{ m} = 1000 \times 9.81 \times 1 \times 0.25$$

= 2452.5 N. Ans.

Problem 3.38 *A tank contains water upto a depth of 1.5 m. The length and width of the tank are 4 m and 2 m respectively. The tank is moving up an inclined plane with a constant acceleration of 4 m/s^2. The inclination of the plane with the horizontal is 30° as shown in Fig. 3.53. Find,*

(i) the angle made by the free surface of water with the horizontal.

(ii) the pressure at the bottom of the tank at the front and rear ends.

Solution. Given :

Depth of water, $h = 1.5$ m ; Length, $L = 4$ m and Width, $b = 2$ m

Constant acceleration along the inclined plane,

$$a = 4 \text{ m/s}^2$$

Inclination of plane, $\alpha = 30°$

Let θ = Angle made by the free surface of water after the acceleration is imparted to the tank

p_A = Pressure at the bottom of the tank at the front end and p_D = Pressure at the bottom of the tank at the rear end.

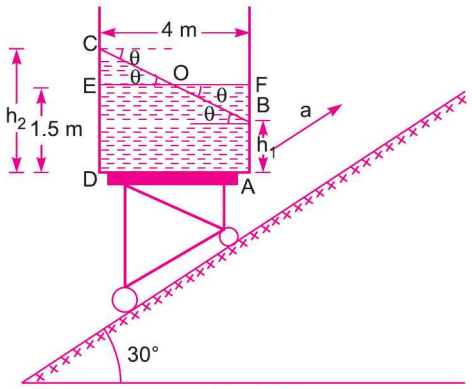


Fig. 3.53

This problem can be done by resolving the given acceleration along the horizontal direction and vertical direction. Then each of these cases may be separately analysed according to the set procedure.

Horizontal and vertical components of the acceleration are :

$$a_x = a \cos \alpha = 4 \cos 30° = 3.464 \text{ m/s}^2$$

$$a_y = a \sin \alpha = 4 \sin 30° = 2 \text{ m/s}^2$$

When the tank is stationary on the inclined plane, free surface of liquid will be along EF as shown in Fig. 3.53. But when the tank is moving upward along the inclined plane the free surface of liquid will be along BC. When the tank containing a liquid is moving up an inclined plane with a constant acceleration, the angle made by the free surface of the liquid with the horizontal is given by

$$\tan\theta = \frac{a_x}{a_y + g} = \frac{3.464}{2 + 9.81} = 0.2933$$

$\therefore$ $\theta = \tan^{-1} 0.2933 = \mathbf{16.346°}$ **or** $\mathbf{16° 20.8'}$. **Ans.**

Now let us first find the depth of liquid at the front and rear end of the tank.

Depth of liquid at front end = $h_1 = AB$

Depth of liquid at rear end = $h_2 = CD$

From Fig. 3.53, in triangle COE, $\tan\theta = \frac{CE}{EO}$

or $CE = EO \tan\theta = 2 \times 0.2933$ ($\because EO = 2$ m, $\tan\theta = 0.2933$)

$= 0.5866$ m

$\therefore$ $CD = h_2 = ED + CE = 1.5 + 0.5866 = 2.0866$ m

Similarly $h_1 = AB = AF - BF$

$= 1.5 - 0.5866$ ($\because AF = 1.5$, $BF = CE = 0.5866$)

$= 0.9134$ m

The pressure at the bottom of tank at the rear end is given by,

$$p_D = \rho g h_2 \left(1 + \frac{a_y}{g}\right)$$

$$= 1000 \times 9.81 \times 2.0866 \left(1 + \frac{2}{9.81}\right) = \mathbf{24642.7\ N/m^2}. \textbf{ Ans.}$$

The pressure at the bottom of tank at the front end is given by

$$p_A = \rho g h_1 \left(1 + \frac{a_y}{g}\right)$$

$$= 1000 \times 9.81 \times 0.9134 \left(1 + \frac{2}{9.81}\right) = \mathbf{10787.2\ N/m^2}. \textbf{ Ans.}$$

HIGHLIGHTS

1. When the fluid is at rest, the shear stress is zero.
2. The force exerted by a static fluid on a vertical, horizontal or an inclined plane immersed surface,

$$F = \rho g A \bar{h}$$

where ρ = Density of the liquid,
A = Area of the immersed surface, and
$\bar{h}$ = Depth of the centre of gravity of the immersed surface from free surface of the liquid.

3. Centre of pressure is defined as the point of application of the resultant pressure.
4. The depth of centre of pressure of an immersed surface from free surface of the liquid,

$$h^* = \frac{I_G}{A\bar{h}} + \bar{h}$$ for vertically immersed surface.

$$= \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h}$$ for inclined immersed surface.

5. The centre of pressure for a plane vertical surface lies at a depth of two-third the height of the immersed surface.
6. The total force on a curved surface is given by $F = \sqrt{F_x^2 + F_y^2}$

   where F_x = Horizontal force on curved surface and is equal to total pressure force on the projected area of the curved surface on the vertical plane,

   $= \rho g A \bar{h}$

   and F_y = Vertical force on sub-merged curved surface and is equal to the weight of liquid actually or imaginary supported by the curved surface.
7. The inclination of the resultant force on curved surface with horizontal, $\tan \theta = \frac{F_y}{F_x}$.
8. The resultant force on a sluice gate, $F = F_1 - F_2$

   where F_1 = Pressure force on the upstream side of the sluice gate and

   F_2 = Pressure force on the downstream side of the sluice gate.
9. For a lock gate, the reaction between the two gates is equal to the reaction at the hinge, $R = P$.

   Also the reaction between the two gates, $P = \frac{F}{2 \sin \theta}$

   where F = Resultant water pressure on the lock gate = $F_1 - F_2$

   and θ = Inclination of the gate with the normal to the side of the lock.

EXERCISE

(A) THEORETICAL PROBLEMS

1. What do you understand by 'Total Pressure' and 'Centre of Pressure' ?
2. Derive an expression for the force exerted on a sub-merged vertical plane surface by the static liquid and locate the position of centre of pressure.
3. Prove that the centre of pressure of a completely sub-merged plane surface is always below the centre of gravity of the sub-merged surface or at most coincide with the centre of gravity when the plane surface is horizontal.
4. Prove that the total pressure exerted by a static liquid on an inclined plane sub-merged surface is the same as the force exerted on a vertical plane surface as long as the depth of the centre of gravity of the surface is unaltered.
5. Derive an expression for the depth of centre of pressure from free surface of liquid of an inclined plane surface sub-merged in the liquid.
6. (*a*) How would you determine the horizontal and vertical components of the resultant pressure on a sub-merged curved surface ?

   (*b*) Explain the procedure of finding hydrostatic forces on curved surfaces.

   (Delhi University, Dec. 2002)
7. Explain how you would find the resultant pressure on a curved surface immersed in a liquid.
8. Why the resultant pressure on a curved sub-merged surface is determined by first finding horizontal and vertical forces on the curved surface ? Why is the same method not adopted for a plane inclined surface sub-merged in a liquid ?

9. Describe briefly with sketches the various methods used for measuring pressure exerted by fluids.
10. Prove that the vertical component of the resultant pressure on a sub-merged curved surface is equal to the weight of the liquid supported by the curved surface.
11. What is the difference between sluice gate and lock gate ?
12. Prove that the reaction between the gates of a lock is equal to the reaction at the hinge.
13. Derive an expression for the reaction between the gates as $P = \frac{F}{2 \sin \theta}$

    where F = Resultant water pressure on lock gate, θ = inclination of the gate with normal to the side of the lock.
14. When will centre of pressure and centre of gravity of an immersed plane surface coincide ?
15. Find an expression for the force exerted and centre of pressure for a completely sub-merged inclined plane surface. Can the same method be applied for finding the resultant force on a curved surface immersed in the liquid ? If not, why ?
16. What do you understand by the hydrostatic equation ? With the help of this equation derive the expressions for the total thrust on a sub-merged plane area and the buoyant force acting on a sub-merged body.

(B) NUMERICAL PROBLEMS

1. Determine the total pressure and depth of centre of pressure on a plane rectangular surface of 1 m wide and 3 m deep when its upper edge is horizontal and (*a*) coincides with water surface (*b*) 2 m below the free water surface. [**Ans.** (*a*) 44145 N, 2.0 m, (*b*) 103005 N, 3.714 m]
2. Determine the total pressure on a circular plate of diameter 1.5 m which is placed vertically in water in such a way that centre of plate is 2 m below the free surface of water. Find the position of centre of pressure also. [**Ans.** 34668.54 N, 2.07 m]
3. A rectangular sluice gate is situated on the vertical wall of a lock. The vertical side of the sluice is 6 m in length and depth of centroid of area is 8 m below the water surface. Prove that the depth of centre of pressure is given by 8.475 m.
4. A circular opening, 3 m diameter, in a vertical side of a tank is closed by a disc of 3 m diameter which can rotate about a horizontal diameter. Calculate : (*i*) the force on the disc, and (*ii*) the torque required to maintain the disc in equilibrium in the vertical position when the head of water above the horizontal diameter is 6 m. [**Ans.** (*i*) 416.05 kN, (*ii*) 39005 Nm]
5. The pressure at the centre of a pipe of diameter 3 m is 29.43 N/cm^2. The pipe contains oil of sp. gr. 0.87 and is filled with a gate valve. Find the force exerted by the oil on the gate and position of centre of pressure. [**Ans.** 2.08 MN, .016 m below centre of pipe]
6. Determine the total pressure and centre of pressure on an isosceles triangular plate of base 5 m and altitude 5 m when the plate is immersed vertically in an oil of sp. gr. 0.8. The base of the plate is 1 m below the free surface of water. [**Ans.** 261927 N, 3.19 m]
7. The opening in a dam is 3 m wide and 2 m high. A vertical sluice gate is used to cover the opening. On the upstream of the gate, the liquid of sp. gr. 1.5, lies upto a height of 2.0 m above the top of the gate, whereas on the downstream side, the water is available upto a height of the top of the gate. Find the resultant force acting on the gate and position of centre of pressure. Assume that the gate is higher at the bottom. [**Ans.** 206010 N, 0.964 m above the hinge]

8. A caisson for closing the entrance to a dry dock is of trapezoidal form 16 m wide at the top and 12 m wide at the bottom and 8 m deep. Find the total pressure and centre of pressure on the caisson if the water on the outside is 1 m below the top level of the caisson and dock is empty.
[**Ans.** 3.164 MN, 4.56 m below water surface]

9. A sliding gate 2 m wide and 1.5 m high lies in a vertical plane and has a co-efficient of friction of 0.2 between itself and guides. If the gate weighs one tonne, find the vertical force required to raise the gate if its upper edge is at a depth of 4 m from free surface of water. [**Ans.** 37768.5 N]

10. A tank contains water upto a height of 1 m above the base. An immiscible liquid of sp. gr. 0.8 is filled on the top of water upto 1.5 m height. Calculate : (*i*) total pressure on one side of the tank, (*ii*) the position of centre of pressure for one side of the tank, which is 3 m wide. [**Ans.** 76518 N, 1.686 m from top]

11. A rectangular tank 4 m long, 1.5 m wide contains water upto a height of 2 m. Calculate the force due to water pressure on the base of the tank. Find also the depth of centre of pressure from free surface.
[**Ans.** 117720 N, 2 m from free surface]

12. A rectangular plane surface 1 m wide and 3 m deep lies in water in such a way that its plane makes an angle of 30° with the free surface of water. Determine the total pressure and position of centre of pressure when the upper edge of the plate is 2 m below the free water surface. [**Ans.** 80932.5 N, 2.318 m]

13. A circular plate 3.0 m diameter is immersed in water in such a way that the plane of the plate makes an angle of 60° with the free surface of water. Determine the total pressure and position of centre of pressure when the upper edge of the plate is 2 m below the free water surface.
[**Ans.** 228.69 kN, 3.427 m from free surface]

14. A rectangular gate 6 m × 2 m is hinged at its base and inclined at 60° to the horizontal as shown in Fig. 3.54. To keep the gate in a stable position, a counter weight of 29430 N is attached at the upper end of the gate. Find the depth of water at which the gate begins to fall. Neglect the weight of the gate and also friction at the hinge and pulley. [**Ans.** 3.43 m]

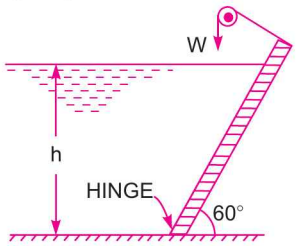


Fig. 3.54

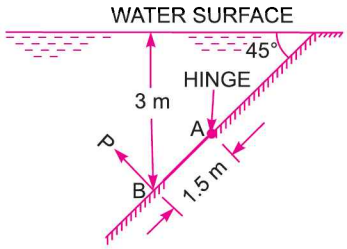


Fig. 3.55

15. An inclined rectangular gate of width 5 m and depth 1.5 m is installed to control the discharge of water as shown in Fig. 3.55. The end *A* is hinged. Determine the force normal to the gate applied at *B* to open it.
[**Ans.** 97435.8 N]

16. A gate supporting water is shown in Fig. 3.56. Find the height '*h*' of the water so that the gate begins to tip about the hinge. Take the width of the gate as unity. [**Ans.** $3 \times \sqrt{3}$ m]

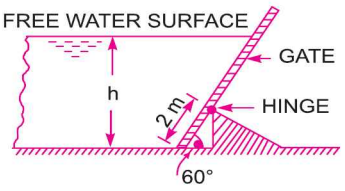


Fig. 3.56

17. Find the total pressure and depth of centre of pressure on a triangular plate of base 3 m and height 3 m which is immersed in water in such a way that plane of the plate makes an angle of 60° with the free surface. The base of the plate is parallel to water surface and at a depth of 2 m from water surface. [**Ans.** 126.52 kN, 2.996 m]

18. Find the horizontal and vertical components of the total force acting on a curved surface AB, which is in the form of a quadrant of a circle of radius 2 m as shown in Fig. 3.57. Take the width of the gate 2 m. [**Ans.** $F_x = 117.72$ kN, $F_y = 140.114$ kN]

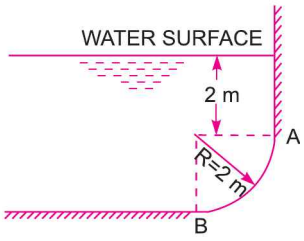


Fig. 3.57

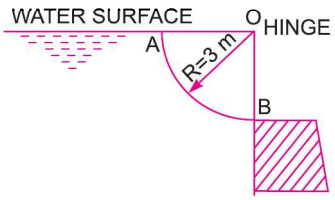


Fig. 3.58

19. Fig. 3.58 shows a gate having a quadrant shape of radius of 3 m. Find the resultant force due to water per metre length of the gate. Find also the angle at which the total force will act. [**Ans.** 82.201 kN, $\theta = 57° 31'$]

20. A roller gate is shown in Fig. 3.59. It is cylindrical form of 6.0 m diameter. It is placed on the dam. Find the magnitude and direction of the resultant force due to water acting on the gate when the water is just going to spill. The length of the gate is given 10 m. [**Ans.** 2.245 MN, $\theta = 38° 8'$]

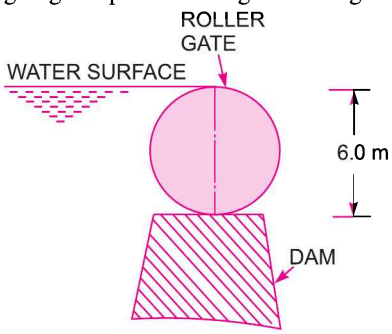


Fig. 3.59

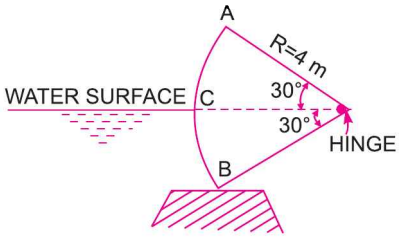


Fig. 3.60

21. Find the horizontal and vertical components of the water pressure exerted on a tainter gate of radius 4 m as shown in Fig. 3.60. Consider width of the gate unity. [**Ans.** $F_x = 19.62$ kN, $F_y = 7102.44$ N]

22. Find the magnitude and direction of the resultant water pressure acting on a curved face of a dam which is shaped according to the relation $y = \frac{x^2}{6}$ as shown in Fig. 3.61. The height of water retained by the dam is 12 m. Take the width of dam as unity. [**Ans.** 970.74 kN, $\theta = 43° 19'$]

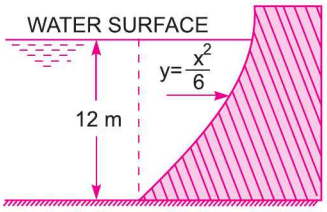


Fig. 3.61

23. Each gate of a lock is 5 m high and is supported by two hinges placed on the top and bottom of the gate. When the gates are closed, they make an angle of 120°. The width of the lock is 4 m. If the depths of water on the two sides of the gates are 4 m and 3 m respectively, determine : (*i*) the magnitude of resultant pressure on each gate, and (*ii*) magnitude of the hinge reactions. [**Ans.** (*i*) 79.279 kN, (*ii*) $R_T = 27.924$ kN, $R_B = 51.355$ kN]

24. The end gates *ABC* of a lock are 8 m high and when closed make an angle of 120°. The width of lock is 10 m. Each gate is supported by two hinges located at 1 m and 5 m above the bottom of the lock. The depth of water on the upstream and downstream sides of the lock are 6 m and 4 m respectively. Find :
(*i*) Resultant water force on each gate.

(*ii*) Reaction between the gates *AB* and *BC*, and

(*iii*) Force on each hinge, considering the reaction of the gate acting in the same horizontal plane as resultant water pressure. [**Ans.** 566.33 kN, (*ii*) 566.33 kN, and (*iii*) $R_T = 173.64$ kN, $R_B = 392.69$ kN]

25. A hollow circular plate of 2 m external and 1 m internal diameter is immersed vertically in water such that the centre of plate is 4 m deep from water surface. Find the total pressure and depth of centre of pressure. [**Ans.** 92.508 kN, 4.078 m]

26. A rectangular opening 2 m wide and 1 m deep in the vertical side of a tank is closed by a sluice gate of the same size. The gate can turn about the horizontal centroidal axis. Determine : (*i*) the total pressure on the sluice gate and (*ii*) the torque on the sluice gate. The head of water above the upper edge of the gate is 1.5 m. [**Ans.** (*i*) 39.24 kN, (*ii*) 1635 Nm]

27. Determine the total force and location of centre of pressure on one face of the plate shown in Fig. 3.62 immersed in a liquid of specific gravity 0.9. [**Ans.** 62.4 kN, 3.04 m]

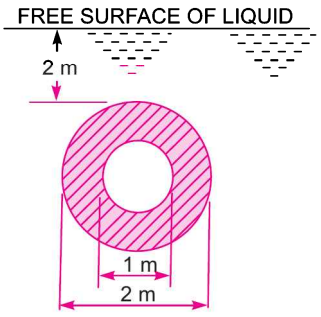


Fig. 3.62

28. A circular opening, 3 m diameter, in the vertical side of water tank is closed by a disc of 3 m diameter which can rotate about a horizontal diameter ? Calculate: (*i*) the force on the disc, and (*ii*) the torque required to maintain the disc in equilibrium in the vertical position when the head of water above the horizontal diameter is 4 m. [**Ans.** (*i*) 270 kN, and (*ii*) 38 kN m]

29. A penstock made up by a pipe of 2 m diameter contains a circular disc of same diameter to act as a valve which controls the discharge passing through it. It can rotate about a horizontal diameter. If the head of water above its centre is 20 m, find the total force acting on the disc and the torque required to maintain it in the vertical position.

30. A circular drum 1.8 m diameter and 1.2 m height is submerged with its axis vertical and its upper end at a depth of 1.8 m below water level. Determine :

(*i*) total pressure on top, bottom and curved surfaces of the drum,

(*ii*) resultant pressure on the whole surface, and

(*iii*) depth of centre of pressure on curved surface.

31. A circular plate of diameter 3 m is immersed in water in such a way that its least and greatest depth from the free surface of water are 1 m and 3 m respectively. For the front side of the plate, find (*i*) total force exerted by water and (*ii*) the position of centre of pressure. [**Ans.** (*i*) 138684 N ; (*ii*) 2.125 m]

32. A tank contains water upto a height of 10 m. One of the sides of the tank is inclined. The angle between free surface of water and inclined side is 60°. The width of the tank is 5 m. Find : (*i*) the force exerted by water on inclined side and (*ii*) position of centre of pressure. [**Ans.** (*i*) 283.1901 kN, (*ii*) 6.67 m]

33. A circular plate of 3 m diameter is under water with its plane making an angle of 30° with the water surface. If the top edge of the plate is 1 m below the water surface, find the force on one side of the plate and its location. *(J.N.T.U., Hyderabad S 2002)*

[**Hint.** $d = 3$ m, $\theta = 30°$, height of top edge = 1 m, $\bar{h} = 1 + 1.5 \times \sin 30°$

$= 1.75$

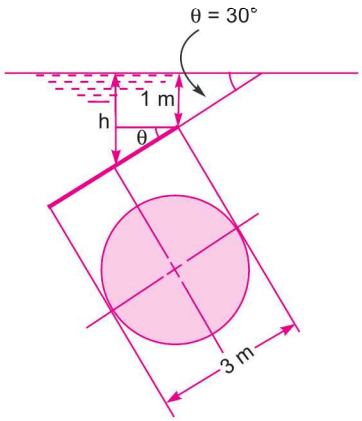


Fig. 3.63

$$F = \rho g A \bar{h} = 1000 \times 9.81 \times \left(\frac{\pi}{4} \times 3^2\right) \times 1.75 = \mathbf{121.35\ kN.}$$

$$h^* = \frac{I_G \sin^2\theta}{A\bar{h}} + \bar{h} = \frac{\frac{\pi}{64}\left(3^4\right)^4 \times \frac{1}{4}}{\frac{\pi}{4}\left(3^2\right) \times 1.75} + 1.75 = 0.08 + 1.75 = \mathbf{1.83\ m.}]$$

4 CHAPTER

BUOYANCY AND FLOATATION

▶ 4.1 INTRODUCTION

In this chapter, the equilibrium of the floating and sub-merged bodies will be considered. Thus the chapter will include : 1. Buoyancy, 2. Centre of buoyancy, 3. Metacentre, 4. Metacentric height, 5. Analytical method for determining metacentric height, 6. Conditions of equilibrium of a floating and sub-merged body, and 7. Experimental method for metacentric height.

▶ 4.2 BUOYANCY

When a body is immersed in a fluid, an upward force is exerted by the fluid on the body. This upward force is equal to the weight of the fluid displaced by the body and is called the force of buoyancy or simply buoyancy.

▶ 4.3 CENTRE OF BUOYANCY

It is defined as the point, through which the force of buoyancy is supposed to act. As the force of buoyancy is a vertical force and is equal to the weight of the fluid displaced by the body, the centre of buoyancy will be the centre of gravity of the fluid displaced.

Problem 4.1 *Find the volume of the water displaced and position of centre of buoyancy for a wooden block of width 2.5 m and of depth 1.5 m, when it floats horizontally in water. The density of wooden block is 650 kg/m^3 and its length 6.0 m.*

Solution. Given :

Width = 2.5 m

Depth = 1.5 m

Length = 6.0 m

Volume of the block = $2.5 \times 1.5 \times 6.0 = 22.50$ m^3

Density of wood, $\rho = 650$ kg/m^3

$\therefore$ Weight of block = $\rho \times g \times$ Volume

$= 650 \times 9.81 \times 22.50$ N $= 143471$ N

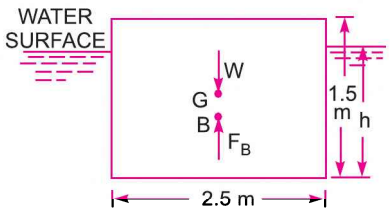


Fig. 4.1

For equilibrium the weight of water displaced = Weight of wooden block

$= 143471$ N

$\therefore$ Volume of water displaced

$$= \frac{\text{Weight of water displaced}}{\text{Weight density of water}} = \frac{143471}{1000 \times 9.81} = \mathbf{14.625\ m^3.\ Ans.}$$

($\because$ Weight density of water $= 1000 \times 9.81$ N/m^3)

Position of Centre of Buoyancy. Volume of wooden block in water

= Volume of water displaced

or $2.5 \times h \times 6.0 = 14.625$ m^3, where h is depth of wooden block in water

$$\therefore \quad h = \frac{14.625}{2.5 \times 6.0} = 0.975 \text{ m}$$

$$\therefore \quad \text{Centre of Buoyancy} = \frac{0.975}{2} = \mathbf{0.4875\ m\ from\ base.\ Ans.}$$

Problem 4.2 *A wooden log of 0.6 m diameter and 5 m length is floating in river water. Find the depth of the wooden log in water when the sp. gravity of the log is 0.7.*

Solution. Given :

Dia. of log $= 0.6$ m

Length, $L = 5$ m

Sp. gr., $S = 0.7$

$\therefore$ Density of log $= 0.7 \times 1000 = 700$ kg/m^3

$\therefore$ Weight density of log, $w = \rho \times g$

$= 700 \times 9.81$ N/m^3

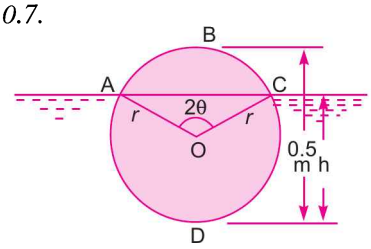


Fig. 4.2

Find depth of immersion or h

Weight of wooden log = Weight density $\times$ Volume of log

$$= 700 \times 9.81 \times \frac{\pi}{4} (D)^2 \times L$$

$$= 700 \times 9.81 \times \frac{\pi}{4} (.6)^2 \times 5 \text{ N} = 989.6 \times 9.81 \text{ N}$$

For equilibrium,

Weight of wooden log = Weight of water displaced

= Weight density of water $\times$ Volume of water displaced

$$\therefore \quad \text{Volume of water displaced} = \frac{989.6 \times 9.81}{1000 \times 9.81} = 0.9896 \text{ m}^3$$

($\because$ Weight density of water $= 1000 \times 9.81$ N/m^3)

Let h is the depth of immersion

$\therefore$ Volume of log inside water = Area of $ADCA \times$ Length

= Area of $ADCA \times 5.0$

But volume of log inside water = Volume of water displaced $= 0.9896$ m^3

$\therefore$ $0.9896 = \text{Area of } ADCA \times 5.0$

$\therefore$ Area of $ADCA = \dfrac{0.9896}{5.0} = 0.1979 \text{ m}^2$

But area of $ADCA$ = Area of curved surface $ADCOA$ + Area of ΔAOC

$$= \pi r^2 \left[\frac{360^\circ - 2\theta}{360^\circ}\right] + \frac{1}{2} r \cos\theta \times 2r \sin\theta$$

$$= \pi r^2 \left[1 - \frac{\theta}{180^\circ}\right] + r^2 \cos\theta \sin\theta$$

$\therefore$ $$0.1979 = \pi (.3)^2 \left[1 - \frac{\theta}{180^\circ}\right] + (.3)^2 \cos\theta \sin\theta$$

$$0.1979 = .2827 - .00157\,\theta + 0.9 \cos\theta \sin\theta$$

or $$.00157\,\theta - .09 \cos\theta \sin\theta = .2827 - .1979 = 0.0848$$

$$\theta - \frac{.09}{.00157} \cos\theta \sin\theta = \frac{.0848}{.00157}$$

or $$\theta - 57.32 \cos\theta \sin\theta = 54.01.$$

or $$\theta - 57.32 \cos\theta \sin\theta - 54.01 = 0$$

For $\theta = 60^\circ$, $60 - 57.32 \times 0.5 \times .866 - 54.01 = 60 - 24.81 - 54.01 = -18.82$

For $\theta = 70^\circ$, $70 - 57.32 \times .342 \times 0.9396 - 54.01 = 70 - 18.4 - 54.01 = -2.41$

For $\theta = 72^\circ$, $72 - 57.32 \times .309 \times .951 - 54.01 = 72 - 16.84 - 54.01 = +1.14$

For $\theta = 71^\circ$, $71 - 57.32 \times .325 \times .9455 - 54.01 = 71 - 17.61 - 54.01 = -0.376$

$\therefore$ $\theta = 71.5^\circ$, $71.5 - 57.32 \times .3173 \times .948 - 54.01 = 71.5 - 17.24 - 54.01 = +.248$

Then $h = r + r \cos 71.5^\circ$

$= 0.3 + 0.3 \times 0.3173 =$ **0.395 m. Ans.**

Problem 4.3 *A stone weighs 392.4 N in air and 196.2 N in water. Compute the volume of stone and its specific gravity.*

Solution. Given :

Weight of stone in air = 392.4 N

Weight of stone in water = 196.2 N

For equilibrium,

Weight in air – Weight of stone in water = Weight of water displaced

or $$392.4 - 196.2 = 196.2 = 1000 \times 9.81 \times \text{Volume of water displaced}$$

$\therefore$ Volume of water displaced

$$= \frac{196.2}{1000 \times 9.81} = \frac{1}{50} \text{ m}^3 = \frac{1}{50} \times 10^6 \text{ cm}^3 = \mathbf{2 \times 10^4 \text{ cm}^3}. \textbf{ Ans.}$$

= Volume of stone

$\therefore$ Volume of stone = $\mathbf{2 \times 10^4 \text{ cm}^3}$**. Ans.**

Specific Gravity of Stone

Mass of stone $= \dfrac{\text{Weight in air}}{g} = \dfrac{392.4}{9.81} = 40 \text{ kg}$

Density of stone $= \dfrac{\text{Mass in air}}{\text{Volume}} = \dfrac{40.0 \text{ kg}}{\frac{1}{50} \text{m}^3} = 40 \times 50 = 2000 \dfrac{\text{kg}}{\text{m}^3}$

$\therefore$ Sp. gr. of stone $= \dfrac{\text{Density of stone}}{\text{Density of water}} = \dfrac{2000}{1000} =$ **2.0. Ans.**

Problem 4.4 *A body of dimensions 1.5 m × 1.0 m × 2 m, weighs 1962 N in water. Find its weight in air. What will be its specific gravity ?*

Solution. Given :

Volume of body $= 1.50 \times 1.0 \times 2.0 = 3.0 \text{ m}^3$

Weight of body in water $= 1962$ N

Volume of the water displaced = Volume of the body $= 3.0 \text{ m}^3$

$\therefore$ Weight of water displaced $= 1000 \times 9.81 \times 3.0 = 29430$ N

For the equilibrium of the body

Weight of body in air – Weight of water displaced = Weight in water

$\therefore \quad W_{air} - 29430 = 1962$

$W_{air} = 29430 + 1962 = 31392$ N

Mass of body $= \dfrac{\text{Weight in air}}{g} = \dfrac{31392}{9.81} = 3200 \text{ kg}$

Density of the body $= \dfrac{\text{Mass}}{\text{Volume}} = \dfrac{3200}{3.0} = 1066.67$

$\therefore$ Sp. gravity of the body $= \dfrac{1066.67}{1000} =$ **1.067. Ans.**

Problem 4.5 *Find the density of a metallic body which floats at the interface of mercury of sp. gr. 13.6 and water such that 40% of its volume is sub-merged in mercury and 60% in water.*

Solution. Let the volume of the body $= V \text{ m}^3$

Then volume of body sub-merged in mercury

$$= \frac{40}{100} V = 0.4 \ V \text{ m}^3$$

Volume of body sub-merged in water

$$= \frac{60}{100} \times V = 0.6 \ V \text{ m}^3$$

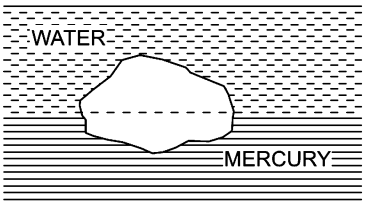


Fig. 4.3

For the equilibrium of the body

Total buoyant force (upward force) = Weight of the body

But total buoyant force = Force of buoyancy due to water + Force of buoyancy due to mercury

Force of buoyancy due to water = Weight of water displaced by body

= Density of water × g × Volume of water displaced

= 1000 × g × Volume of body in water

$= 1000 \times g \times 0.6 \times V$ N

and Force of buoyancy due to mercury = Weight of mercury displaced by body

$= g \times$ Density of mercury $\times$ Volume of mercury displaced

$= g \times 13.6 \times 1000 \times$ Volume of body in mercury

$= g \times 13.6 \times 1000 \times 0.4\ V$ N

Weight of the body = Density $\times\ g \times$ Volume of body $= \rho \times g \times V$

where ρ is the density of the body

$\therefore$ For equilibrium, we have

Total buoyant force = Weight of the body

$1000 \times g \times 0.6 \times V + 13.6 \times 1000 \times g \times .4\ V = \rho \times g \times V$

or $\rho = 600 + 13600 \times .4 = 600 + 54400 = 6040.00$ kg/m^3

$\therefore$ Density of the body = **6040.00 kg/m^3. Ans.**

Problem 4.6 *A float valve regulates the flow of oil of sp. gr. 0.8 into a cistern. The spherical float is 15 cm in diameter. AOB is a weightless link carrying the float at one end, and a valve at the other end which closes the pipe through which oil flows into the cistern. The link is mounted in a frictionless hinge at O and the angle AOB is 135°. The length of OA is 20 cm, and the distance between the centre of the float and the hinge is 50 cm. When the flow is stopped AO will be vertical. The valve is to be pressed on to the seat with a force of 9.81 N to completely stop the flow of oil into the cistern. It was observed that the flow of oil is stopped when the free surface of oil in the cistern is 35 cm below the hinge. Determine the weight of the float.*

Solution. Given :

Sp. gr. of oil $= 0.8$

$\therefore$ Density of oil, $\rho_0 = 0.8 \times 1000$

$= 800$ kg/m^3

Dia. of float, $D = 15$ cm

$\angle AOB = 135°$

$OA = 20$ cm

Force, $P = 9.81$ N

$OB = 50$ cm

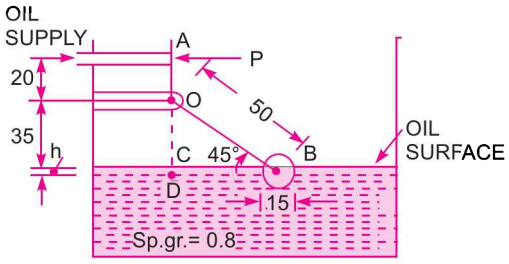


Fig. 4.4

Find the weight of the float. Let it is equal to W.

When the flow of oil is stopped, the centre of float is shown in Fig. 4.4

The level of oil is also shown. The centre of float is below the level of oil, by a depth 'h'.

From ΔBOD, $\sin 45° = \dfrac{OD}{OB} = \dfrac{OC + CD}{OB} = \dfrac{35 + h}{50}$

$\therefore$ $50 \times \sin 45° = 35 + h$

or $h = 50 \times \dfrac{1}{\sqrt{2}} - 35 = 35.355 - 35 = 0.355$ cm $= .00355$ m.

The weight of float is acting through B, but the upward buoyant force is acting through the centre of weight of oil displaced.

Volume of oil displaced $= \dfrac{2}{3}\pi r^3 + h \times \pi r^2$ $\left\{ r = \dfrac{D}{2} = \dfrac{15}{2} = 7.5 \text{ cm} \right\}$

$$= \tfrac{2}{3} \times \pi \times (.075)^3 + .00355 \times \pi \times (.075)^2 = 0.000945 \text{ m}^3$$

$\therefore$ Buoyant force = Weight of oil displaced

$= \rho_0 \times g \times$ Volume of oil

$= 800 \times 9.81 \times .000945 = 7.416$ N

The buoyant force and weight of the float passes through the same vertical line, passing through B. Let the weight of float is W. Then net vertical force on float

= Buoyant force – Weight of float = $(7.416 - W)$

Taking moments about the hinge O, we get

$$P \times 20 = (7.416 - W) \times BD = (7.416 - W) \times 50 \times \cos 45°$$

or $$9.81 \times 20 = (7.416 - W) \times 35.355$$

$\therefore$ $$W = 7.416 - \frac{20 \times 9.81}{35.355} = 7.416 - 5.55 = \textbf{1.866 N. Ans.}$$

▶ 4.4 META-CENTRE

It is defined as the point about which a body starts oscillating when the body is tilted by a small angle. The meta-centre may also be defined as the point at which the line of action of the force of buoyancy will meet the normal axis of the body when the body is given a small angular displacement.

Consider a body floating in a liquid as shown in Fig. 4.5 (*a*). Let the body is in equilibrium and G is the centre of gravity and B the centre of buoyancy. For equilibrium, both the points lie on the normal axis, which is vertical.

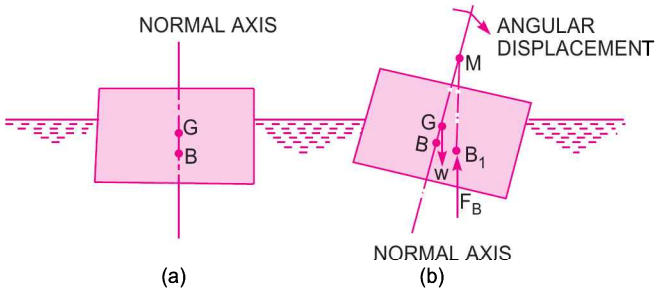


Fig. 4.5 *Meta-centre*

Let the body is given a small angular displacement in the clockwise direction as shown in Fig. 4.5 (*b*). The centre of buoyancy, which is the centre of gravity of the displaced liquid or centre of gravity of the portion of the body sub-merged in liquid, will now be shifted towards right from the normal axis. Let it is at B_1 as shown in Fig. 4.5 (*b*). The line of action of the force of buoyancy in this new position, will intersect the normal axis of the body at some point say M. This point M is called **Meta-centre.**

▶ 4.5 META-CENTRIC HEIGHT

The distance MG, *i.e.*, the distance between the meta-centre of a floating body and the centre of gravity of the body is called meta-centric height.

▶ 4.6 ANALYTICAL METHOD FOR META-CENTRE HEIGHT

Fig. 4.6 (*a*) shows the position of a floating body in equilibrium. The location of centre of gravity and centre of buoyancy in this position is at G and B. The floating body is given a small angular displacement in the clockwise direction. This is shown in Fig. 4.6 (*b*). The new centre of buoyancy is at B_1. The vertical line through B_1 cuts the normal axis at M. Hence M is the meta-centre and GM is meta-centric height.

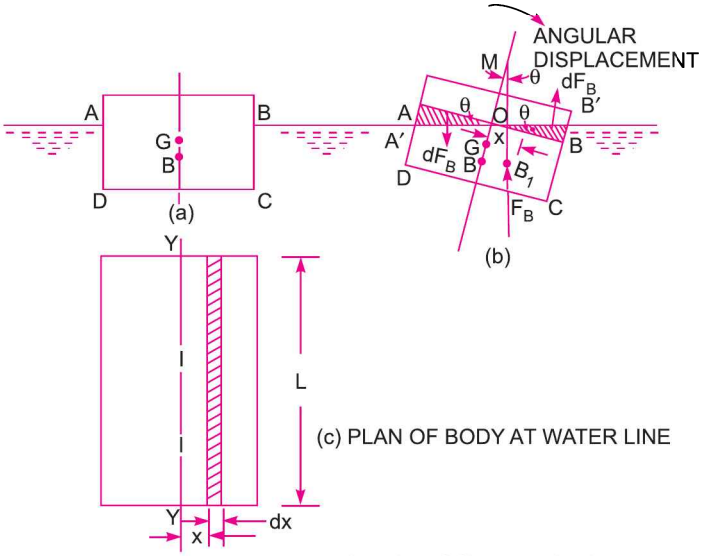


Fig. 4.6 *Meta-centre height of floating body.*

The angular displacement of the body in the clockwise direction causes the wedge-shaped prism BOB' on the right of the axis to go inside the water while the identical wedge-shaped prism represented by AOA' emerges out of the water on the left of the axis. These wedges represent a gain in buoyant force on the right side and a corresponding loss of buoyant force on the left side. The gain is represented by a vertical force dF_B acting through the $C.G.$ of the prism BOB' while the loss is represented by an equal and opposite force dF_B acting vertically downward through the centroid of AOA'. The couple due to these buoyant forces dF_B tends to rotate the ship in the counterclockwise direction. Also the moment caused by the displacement of the centre of buoyancy from B to B_1 is also in the counterclockwise direction. Thus these two couples must be equal.

Couple Due to Wedges. Consider towards the right of the axis a small strip of thickness dx at a distance x from O as shown in Fig. 4.5 (*b*). The height of strip $x \times \angle BOB' = x \times \theta$.

$\{\because \angle BOB' = \angle AOA' = BMB_1' = \theta\}$

$\therefore$ Area of strip = Height × Thickness = $x \times \theta \times dx$

If L is the length of the floating body, then

Volume of strip = Area × L

$= x \times \theta \times L \times dx$

$\therefore$ Weight of strip = $\rho g \times$ Volume = $\rho g x\, \theta L\, dx$

Similarly, if a small strip of thickness dx at a distance x from O towards the left of the axis is considered, the weight of strip will be $\rho g x \theta\, L\, dx$. The two weights are acting in the opposite direction and hence constitute a couple.

Moment of this couple = Weight of each strip × Distance between these two weights

$= \rho g x\ \theta L\ dx\ [x + x]$

$= \rho g x\ \theta L\ dx \times 2x = 2\rho g x^2\ \theta L\ dx$

$\therefore$ Moment of the couple for the whole wedge

$= \int 2\rho g x^2\ \theta L\ dx$ ...(4.1)

Moment of couple due to shifting of centre of buoyancy from B to B_1

$= F_B \times BB_1$

$= F_B \times BM \times \theta$ $\{\because BB_1 = BM \times \theta$ if θ is very small$\}$

$= W \times BM \times \theta$ $\{\because F_B = W\}$ ...(4.2)

But these two couples are the same. Hence equating equations (4.1) and (4.2), we get

$$W \times BM \times \theta = \int 2\rho g x^2\ \theta\ Ldx$$

$$W \times BM \times \theta = 2\rho g \theta \int x^2 Ldx$$

$$W \times BM = 2\rho g \int x^2 Ldx$$

Now Ldx = Elemental area on the water line shown in Fig. 4.6 (c) and = dA

$\therefore$ $W \times BM = 2\rho g \int x^2 dA.$

But from Fig. 4.5 (c) it is clear that $2 \int x^2\ dA$ is the second moment of area of the plan of the body at water surface about the axis Y-Y. Therefore

$W \times BM = \rho g I$ {where $I = 2 \int x^2\ dA$}

$\therefore$ $$BM = \frac{\rho g I}{W}$$

But W = Weight of the body

= Weight of the fluid displaced by the body

= $\rho g \times$ Volume of the fluid displaced by the body

= $\rho g \times$ Volume of the body sub-merged in water

= $\rho g \times \forall$

$\therefore$ $$BM = \frac{\rho g \times I}{\rho g \times \forall} = \frac{I}{\forall}$$ ...(4.3)

$$GM = BM - BG = \frac{I}{\forall} - BG$$

$\therefore$ Meta-centric height $= GM = \frac{I}{\forall} - BG.$ ...(4.4)

Problem 4.7 *A rectangular pontoon is 5 m long, 3 m wide and 1.20 m high. The depth of immersion of the pontoon is 0.80 m in sea water. If the centre of gravity is 0.6 m above the bottom of the pontoon, determine the meta-centric height. The density for sea water = 1025 kg/m^3.*

Solution. Given :

Dimension of pontoon = 5 m × 3 m × 1.20 m

Depth of immersion = 0.8 m

Distance $AG = 0.6$ m

Distance $AB = \frac{1}{2} \times$ Depth of immersion

$= \frac{1}{2} \times .8 = 0.4$ m

Density for sea water $= 1025$ kg/m^3

Meta-centre height GM, given by equation (4.4) is

$$GM = \frac{I}{\forall} - BG$$

where I = M.O. Inertia of the plan of the pontoon about Y-Y axis

$$= \frac{1}{12} \times 5 \times 3^3 \text{ m}^4 = \frac{45}{4} \text{ m}^4$$

$\forall$ = Volume of the body sub-merged in water

$= 3 \times 0.8 \times 5.0 = 12.0$ m^3

$BG = AG - AB = 0.6 - 0.4 = 0.2$ m

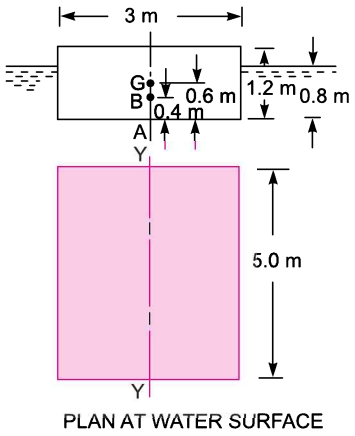


Fig. 4.7

$$\therefore \quad GM = \frac{45}{4} \times \frac{1}{12.0} - 0.2 = \frac{45}{48} - 0.2 = 0.9375 - 0.2 = \textbf{0.7375 m. Ans.}$$

Problem 4.8 *A uniform body of size 3 m long × 2 m wide × 1 m deep floats in water. What is the weight of the body if depth of immersion is 0.8 m ? Determine the meta-centric height also.*

Solution. Given :

Dimension of body $= 3 \times 2 \times 1$

Depth of immersion $= 0.8$ m

Find (*i*)Weight of body, W

(*ii*) Meta-centric height, GM

(*i*) **Weight of Body, W**

= Weight of water displaced

= $\rho g \times$ Volume of water displaced

= $1000 \times 9.81 \times$ Volume of body in water

= $1000 \times 9.81 \times 3 \times 2 \times 0.8$ N

= **47088 N. Ans.**

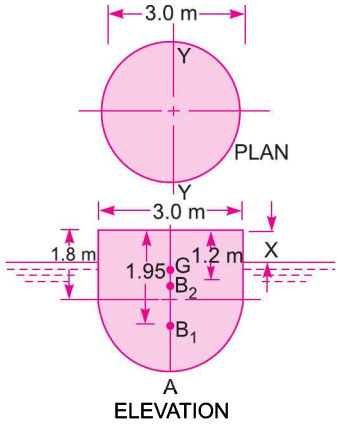


Fig. 4.8

(*ii*) **Meta-centric Height, GM**

Using equation (4.4), we get

$$GM = \frac{I}{\forall} - BG$$

where I = M.O.I about Y-Y axis of the plan of the body

$$= \frac{1}{12} \times 3 \times 2^3 = \frac{3 \times 2^3}{12} = 2.0 \text{ m}^4$$

$\forall$ = Volume of body in water

$= 3 \times 2 \times 0.8 = 4.8$ m^3

$$BG = AG - AB = \frac{1.0}{2} - \frac{0.8}{2} = 0.5 - 0.4 = 0.1$$

$$\therefore \quad GM = \frac{2.0}{4.8} - 0.1 = 0.4167 - 0.1 = \textbf{0.3167 m. Ans.}$$

Problem 4.9 *A block of wood of specific gravity 0.7 floats in water. Determine the meta-centric height of the block if its size is 2 m × 1 m × 0.8 m.*

Solution. Given :

Dimension of block = $2 \times 1 \times 0.8$

Let depth of immersion = h m

Sp. gr. of wood = 0.7

Weight of wooden piece = Weight density of wood* × Volume

$= 0.7 \times 1000 \times 9.81 \times 2 \times 1 \times 0.8$ N

Weight of water displaced = Weight density of water × Volume of the wood sub-merged in water

$= 1000 \times 9.81 \times 2 \times 1 \times h$ N

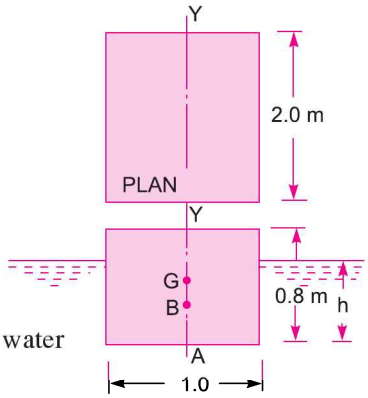


Fig. 4.9

For equilibrium,

Weight of wooden piece = Weight of water displaced

$\therefore \quad 700 \times 9.81 \times 2 \times 1 \times 0.8 = 1000 \times 9.81 \times 2 \times 1 \times h$

$$\therefore \quad h = \frac{700 \times 9.81 \times 2 \times 1 \times 0.8}{1000 \times 9.81 \times 2 \times 1} = 0.7 \times 0.8 = 0.56 \text{ m}$$

$\therefore$ Distance of centre of Buoyancy from bottom, *i.e.*,

$$AB = \frac{h}{2} = \frac{0.56}{2} = 0.28 \text{ m}$$

and $AG = 0.8/2.0 = 0.4$ m

$\therefore \quad BG = AG - AB = 0.4 - 0.28 = 0.12$ m

The meta-centric height is given by equation (4.4) or

$$GM = \frac{I}{\forall} - BG$$

where $I = \frac{1}{12} \times 2 \times 1.0^3 = \frac{1}{6}$ m^4

$\forall$ = Volume of wood in water

$= 2 \times 1 \times h = 2 \times 1 \times .56 = 1.12$ m^3

$$\therefore \quad GM = \frac{1}{6} \times \frac{1}{1.12} - 0.12 = 0.1488 - 0.12 = \mathbf{0.0288 \text{ m. Ans.}}$$

Problem 4.10 *A solid cylinder of diameter 4.0 m has a height of 3 metres. Find the meta-centric height of the cylinder when it is floating in water with its axis vertical. The sp. gr. of the cylinder = 0.6.*

Solution. Given :

Dia. of cylinder, $D = 4.0$ m

Height of cylinder, $h = 3.0$ m

---

* Weight density of wood = $\rho \times g$, where ρ = density of wood

$= 0.7 \times 1000 = 700$ kg/m^3. Hence w for wood $= 700 \times 9.81$ N/m^3.

Sp. gr. of cylinder = 0.6

Depth of immersion of cylinder

$$= 0.6 \times 3.0 = 1.8 \text{ m}$$

$$\therefore \quad AB = \frac{1.8}{2} = 0.9 \text{ m}$$

and $$AG = \frac{3}{2} = 1.5 \text{ m}$$

$$\therefore \quad BG = AG - AB$$

$$= 1.5 - 0.9 = 0.6 \text{ m}$$

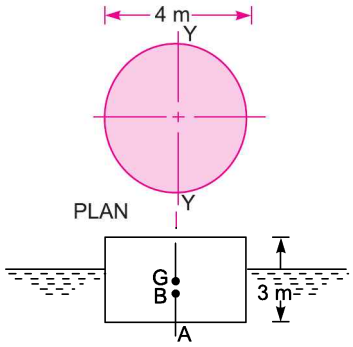


Fig. 4.10

Now the meta-centric height GM is given by equation (4.4)

$$GM = \frac{I}{\forall} - BG$$

But I = M.O.I. about Y-Y axis of the plan of the body

$$= \frac{\pi}{64} D^4 = \frac{\pi}{64} \times (4.0)^4$$

and $\forall$ = Volume of cylinder in water

$$= \frac{\pi}{4} D^2 \times \text{Depth of immersion}$$

$$= \frac{\pi}{4} (4)^2 \times 1.8 \text{ m}^3$$

$$\therefore \quad GM = \frac{\frac{\pi}{64} \times (4.0)^4}{\frac{\pi}{4} \times (4.0)^2 \times 1.8} - 0.6$$

$$= \frac{1}{16} \times \frac{4.0^2}{1.8} - 0.6 = \frac{1}{1.8} - 0.6 = 0.55 - 0.6 = \mathbf{-0.05 \text{ m. Ans.}}$$

– ve sign means that meta-centre, (M) is below the centre of gravity (G).

Problem 4.11 *A body has the cylindrical upper portion of 3 m diameter and 1.8 m deep. The lower portion is a curved one, which displaces a volume of 0.6 m^3 of water. The centre of buoyancy of the curved portion is at a distance of 1.95 m below the top of the cylinder. The centre of gravity of the whole body is 1.20 m below the top of the cylinder. The total displacement of water is 3.9 tonnes. Find the meta-centric height of the body.*

Solution. Given :

Dia. of body = 3.0 m

Depth of body = 1.8 m

Volume displaced by curved portion

$$= 0.6 \text{ m}^3 \text{ of water.}$$

Let B_1 is the centre of buoyancy of the curved surface and G is the centre of gravity of the whole body.

Then $CB_1 = 1.95$ m

$CG = 1.20$ m

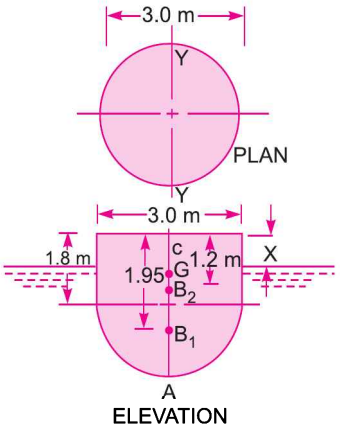


Fig. 4.11

Total weight of water displaced by body = 3.9 tonnes

$= 3.9 \times 1000 = 3900$ kgf

$= 3900 \times 9.81$ N $= 38259$ N

Find **meta-centric** height of the body.

Let the height of the body above the water surface x m. Total weight of water displaced by body

= Weight density of water × [Volume of water displaced]

= 1000 × 9.81 × [Volume of the body in water]

= 9810 [Volume of cylindrical part in water + Volume of curved portion]

$$= 9810 \left[\frac{\pi}{4} \times D^2 \times \text{Depth of cylindrical part in water} + \text{Volume displaced by curved portion}\right]$$

or $$38259 = 9810 \left[\frac{\pi}{4}(3)^2 \times (1.8 - x) + 0.6\right]$$

$\therefore$ $$\frac{\pi}{4}(3)^2 \times (1.8 - x) + 0.6 = \frac{38259}{9810} = 3.9$$

$\therefore$ $$\frac{\pi}{4} \times 3^2 \times (1.8 - x) = 3.9 - 0.6 = 3.3$$

or $$1.8 - x = \frac{3.3 \times 4}{\pi \times 3 \times 3} = 0.4668$$

$\therefore$ $$x = 1.8 - .4668 = 1.33 \text{ m}$$

Let B_2 is the centre of buoyancy of cylindrical part and B is the centre of buoyancy of the whole body.

Then depth of cylindrical part in water $= 1.8 - x = 0.467$ m

$\therefore$ $$CB_2 = x + \frac{.467}{2} = 1.33 + .2335 = 1.5635 \text{ m.}$$

The distance of the centre of buoyancy of the whole body from the top of the cylindrical part is given as

CB = (Volume of curved portion × CB_1 + Volume of cylindrical part in water × CB_2) ÷ (Total volume of water displaced)

$$= \frac{0.6 \times 1.95 + 3.3 \times 1.5635}{(0.6 + 3.3)} = \frac{1.17 + 5.159}{3.9} = 1.623 \text{ m.}$$

Then $BG = CB - CG = 1.623 - 1.20 = .423$ m.

Meta-centric height, GM, is given by

$$GM = \frac{I}{\forall} - BG$$

where I = M.O.I. of the plan of the body at water surface about Y-Y axis

$$= \frac{\pi}{64} \times D^4 = \frac{\pi}{64} \times 3^4 \text{ m}^4$$

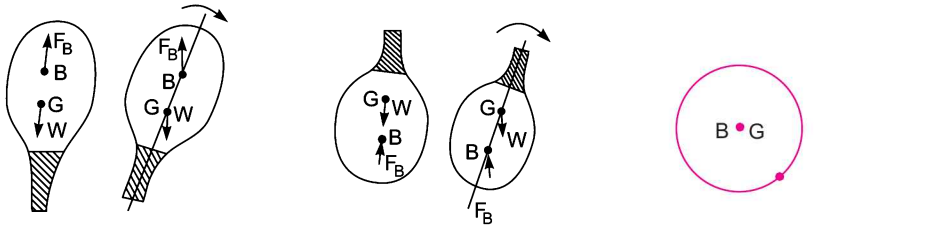
$\forall$ = Volume of the body in water = 3.9 m^3

$$\therefore \qquad GM = \frac{\pi}{64} \times \frac{3^4}{3.9} - .423 = 1.019 - .423 = \textbf{0.596 m. Ans.}$$

▶ 4.7 CONDITIONS OF EQUILIBRIUM OF A FLOATING AND SUB-MERGED BODIES

A sub-merged or a floating body is said to be stable if it comes back to its original position after a slight disturbance. The relative position of the centre of gravity (G) and centre of buoyancy (B_1) of a body determines the stability of a sub-merged body.

4.7.1 Stability of a Sub-merged Body. The position of centre of gravity and centre of buoyancy in case of a completely sub-merged body are fixed. Consider a balloon, which is completely sub-merged in air. Let the lower portion of the balloon contains heavier material, so that its centre of gravity is lower than its centre of buoyancy as shown in Fig. 4.12 (a). Let the weight of the balloon is W. The weight W is acting through G, vertically in the downward direction, while the buoyant force F_B is acting vertically up, through B. For the equilibrium of the balloon $W = F_B$. If the balloon is given an angular displacement in the clockwise direction as shown in Fig. 4.12 (a), then W and F_B constitute a couple acting in the anti-clockwise direction and brings the balloon in the original position. Thus the balloon in the position, shown by Fig. 4.12 (a) is in stable equilibrium.



(a) STABLE EQUILIBRIUM (b) UNSTABLE EQUILIBRIUM (c) NEUTRAL EQUILIBRIUM

Fig. 4.12 *Stabilities of sub-merged bodies.*

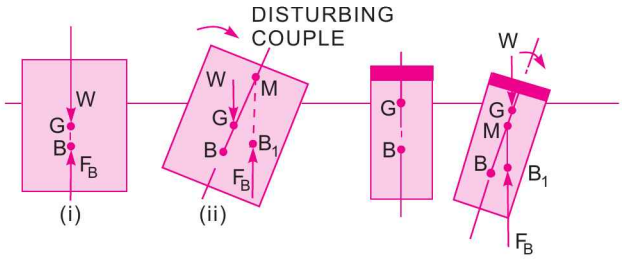
(a) **Stable Equilibrium.** When $W = F_B$ and point B is above G, the body is said to be in stable equilibrium.

(b) **Unstable Equilibrium.** If $W = F_B$, but the centre of buoyancy (B) is below centre of gravity (G), the body is in unstable equilibrium as shown in Fig. 4.12 (b). A slight displacement to the body, in the clockwise direction, gives the couple due to W and F_B also in the clockwise direction. Thus the body does not return to its original position and hence the body is in unstable equilibrium.

(c) **Neutral Equilibrium.** If $F_B = W$ and B and G are at the same point, as shown in Fig. 4.12 (c), the body is said to be in neutral equilibrium.

4.7.2 Stability of Floating Body. The stability of a floating body is determined from the position of Meta-centre (M). In case of floating body, the weight of the body is equal to the weight of liquid displaced.

(*a*) **Stable Equilibrium.** If the point *M* is above *G*, the floating body will be in stable equilibrium as shown in Fig. 4.13 (*a*). If a slight angular displacement is given to the floating body in the clockwise direction, the centre of buoyancy shifts from *B* to B_1 such that the vertical line through B_1 cuts at *M*. Then the buoyant force F_B through B_1 and weight *W* through *G* constitute a couple acting in the anti-clockwise direction and thus bringing the floating body in the original position.



(a) Stable equilibrium *M* is above *G* (b) Unstable equilibrium *M* is below *G*.

Fig. 4.13 *Stability of floating bodies.*

(*b*) **Unstable Equilibrium.** If the point *M* is below *G*, the floating body will be in unstable equilibrium as shown in Fig. 4.13 (*b*). The disturbing couple is acting in the clockwise direction. The couple due to buoyant force F_B and *W* is also acting in the clockwise direction and thus overturning the floating body.

(*c*) **Neutral Equilibrium.** If the point *M* is at the centre of gravity of the body, the floating body will be in neutral equilibrium.

Problem 4.12 *A solid cylinder of diameter 4.0 m has a height of 4.0 m. Find the meta-centric height of the cylinder if the specific gravity of the material of cylinder = 0.6 and it is floating in water with its axis vertical. State whether the equilibrium is stable or unstable.*

Solution. Given : $D = 4$ m

Height, $h = 4$ m

Sp. gr. $= 0.6$

Depth of cylinder in water = Sp. gr. $\times h$

$= 0.6 \times 4.0 = 2.4$ m

$\therefore$ Distance of centre of buoyancy (*B*) from *A*

or $$AB = \frac{2.4}{2} = 1.2 \text{ m}$$

Distance of centre of gravity (*G*) from *A*

or $$AG = \frac{h}{2} = \frac{4.0}{2} = 2.0 \text{ m}$$

$\therefore$ $$BG = AG - AB = 2.0 - 1.2 = 0.8 \text{ m}$$

Now the meta-centric height *GM* is given by

$$GM = \frac{I}{\forall} - BG$$

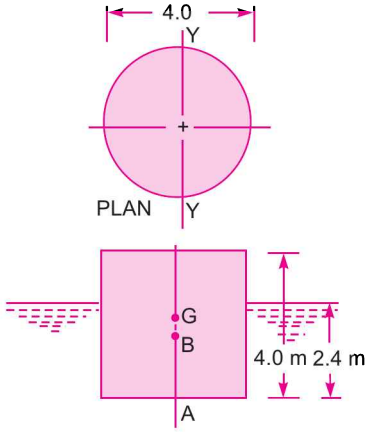


Fig. 4.14

where I = M.O.I. of the plan of the body about Y-Y axis

$$= \frac{\pi}{64} D^4 = \frac{\pi}{64} \times (4.0)^4$$

$\forall$ = Volume of cylinder in water

$$= \frac{\pi}{4.0} \times D^2 \times \text{Depth of cylinder in water} = \frac{\pi}{4} \times 4^2 \times 2.4 \text{ m}^3$$

$$\therefore \quad \frac{I}{\forall} = \frac{\frac{\pi}{64} \times 4^4}{\frac{\pi}{4} \times 4^2 \times 2.4} = \frac{1}{16} \times \frac{4^2}{2.4} = \frac{1}{2.4} = 0.4167 \text{ m}$$

$$\therefore \quad GM = \frac{I}{\forall} - BG = 0.4167 - 0.8 = \mathbf{-0.3833 \text{ m. Ans.}}$$

–ve sign means that the meta-centre (M) is below the centre of gravity (G). Thus the cylinder is in unstable equilibrium. **Ans.**

Problem 4.13 *A solid cylinder of 10 cm diameter and 40 cm long, consists of two parts made of different materials. The first part at the base is 1.0 cm long and of specific gravity = 6.0. The other part of the cylinder is made of the material having specific gravity 0.6. State, if it can float vertically in water.*

Solution. Given :

	$D = 10$ cm
Length,	$L = 40$ cm
Length of 1st part,	$l_1 = 1.0$ cm
Sp. gr.,	$S_1 = 6.0$
Density of 1st part,	$\rho_1 = 6 \times 1000 = 6000$ kg/m^3
Length of 2nd part,	$l_2 = 40 - 1.0 = 39.0$ cm
Sp. gr.,	$S_2 = 0.6$
Density of 2nd part,	$\rho_2 = 0.6 \times 1000 = 600$ kg/m^3

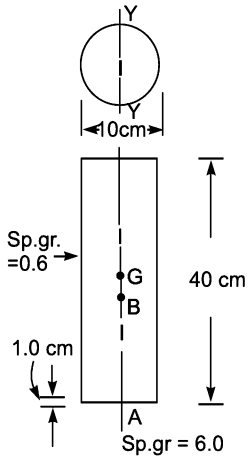


Fig. 4.15

The cylinder will float vertically in water if its meta-centric height GM is positive. To find meta-centric height, find the location of centre of gravity (G) and centre of buoyancy (B) of the combined solid cylinder. The distance of the centre of gravity of the solid cylinder from A is given as

AG = [(Weight of 1st part × Distance of C.G. of 1st part from A)
+ (Weight of 2nd part of cylinder
× Distance of C.G. of 2nd part from A)]
÷ [Weight of 1st part + weight of 2nd part]

$$= \frac{\left(\frac{\pi}{4} D^2 \times 1.0 \times 6.0 \times 0.5\right) + \left(\frac{\pi}{4} D^2 \times 39.0 \times 0.6 \times (1.0 \times 39/2)\right)}{\left(\frac{\pi}{4} D^2 \times 1.0 \times 6.0 + \frac{\pi}{4} D^2 \times 39 \times 0.6\right)}$$

$$= \frac{1.0 \times 6.0 \times 0.5 + 39.0 \times .6 \times (20.5)}{1.0 \times 6.0 + 39.0 \times 0.6}$$

Cancel $\frac{\pi}{4} D^2$ in the Numerator and Denominator $= \frac{3.0 + 479.7}{6.0 + 23.4} = \frac{482.7}{29.4} = 16.42.$

To find the centre of buoyancy of the combined two parts or of the cylinder, determine the depth of immersion of the cylinder. Let the depth of immersion of the cylinder is h. Then

Weight of the cylinder = Weight of water displaced

$$\frac{\pi}{4} \times (.1)^2 \times \frac{39.0}{100} \times 600 \times 9.81 + \frac{\pi}{4} (.1)^2 \times \frac{1.0}{100} \times 6000 \times 9.81 = \frac{\pi}{4} (.1)^2 \times \frac{h}{100} \times 1000 \times 9.81$$

[$\because$ h is in cm]

or cancelling $\frac{\pi}{4} (.1)^2 \times \frac{1000 \times 9.81}{100}$ throughout, we get

$$39.0 \times 0.6 + 1.0 \times 6.0 = h \quad \text{or} \quad h = 23.4 + 6.0 = 29.4$$

$\therefore$ The distance of the centre of the buoyancy B, of the cylinder from A is

$$AB = h/2 = \frac{29.4}{2} = 14.7$$

$\therefore$ $$BG = AG - AB = 16.42 - 14.70 = 1.72 \text{ cm.}$$

Meta-centric height GM is given by

$$GM = \frac{I}{\forall} - BG$$

where I = M.O.I. of plan of the body about Y-Y axis

$$= \frac{\pi}{64} D^4 = \frac{\pi}{.64} (10)^4 \text{ cm}^4$$

$\forall$ = Volume of cylinder in water

$$= \frac{\pi}{4} D^2 \times h = \frac{\pi}{4} (10)^2 \times 29.4 \text{ m}^3$$

$\therefore$ $$\frac{I}{\forall} = \frac{\pi}{64} (10)^4 \Big/ \frac{\pi}{4}(10)^2 \times 29.4 = \frac{1}{16} \times \frac{10^2}{29.4} = \frac{100}{19 \times 29.4} = 0.212$$

$\therefore$ $$GM = 0.212 - 1.72 = -1.508 \text{ cm}$$

As GM is – ve, it means that the Meta-centre M is below the centre of gravity (G). Thus the cylinder is in unstable equilibrium and so it cannot float vertically in water. **Ans.**

Problem 4.14 *A rectangular pontoon 10.0 m long, 7 m broad and 2.5 m deep weighs 686.7 kN. It carries on its upper deck an empty boiler of 5.0 m diameter weighing 588.6 kN. The centre of gravity of the boiler and the pontoon are at their respective centres along a vertical line. Find the meta-centric height. Weight density of sea water is 10.104 kN/m^3.*

Solution. Given : Dimension of pontoon = $10 \times 7 \times 2.5$

Weight of pontoon, $W_1 = 686.7$ kN

Dia. of boiler, $D = 5.0$ m

Weight of boiler, $W_2 = 588.6$ kN

w for sea water $= 10.104$ kN/m^3

To find the meta-centric height, first determine the common centre of gravity G and common centre of buoyancy B of the boiler and pontoon. Let G_1 and G_2 are the centre of gravities of pontoon and boiler respectively. Then

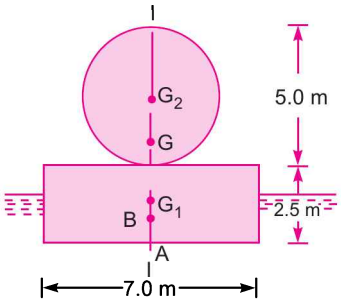


Fig. 4.16

$$AG_1 = \frac{2.5}{2} = 1.25 \text{ m}$$

$$AG_2 = 2.5 + \frac{5.0}{2} = 2.5 + 2.5 = 5.0 \text{ m}$$

The distance of common centre of gravity G from A is given as

$$AG = \frac{W_1 \times AG_1 + W_2 \times AG_2}{W_1 + W_2}$$

$$= \frac{686.7 \times 1.25 + 588.6 \times 5.0}{(686.7 + 588.6)} = 2.98 \text{ m.}$$

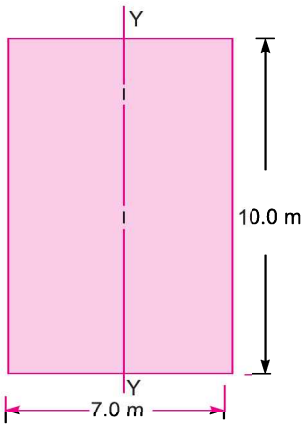


Fig. 4.17 *Plan of the body at water-line*

Let h is the depth of immersion. Then

Total weight of pontoon and boiler = Weight of sea water displaced

or $(686.7 + 588.6) = w \times$ Volume of the pontoon in water

$= 10.104 \times L \times b \times$ Depth of immersion

$\therefore \quad 1275.3 = 10.104 \times 10 \times 7 \times h$

$$h = \frac{1275.3}{10 \times 7 \times 10.104} = 1.803 \text{ m}$$

$\therefore$ The distance of the common centre of buoyancy B from A is

$$AB = \frac{h}{2} = \frac{1.803}{2} = .9015 \text{ m}$$

$\therefore \quad BG = AG - AB = 2.98 - .9015 = 2.0785 \text{ m} \approx 2.078 \text{ m}$

Meta-centric height is given by $GM = \frac{I}{\forall} - BG$

where I = M.O.I. of the plan of the body at the water level along Y-Y

$$= \frac{1}{12} \times 10.0 \times 7^3 = \frac{10 \times 49 \times 7}{12} \text{ m}^4$$

$\forall$ = Volume of the body in water

$= L \times b \times h = 10.0 \times 7 \times 1.857$

$\therefore \quad \frac{I}{\forall} = \frac{10 \times 49 \times 7}{12 \times 10 \times 7 \times 1.857} = \frac{49}{12 \times 1.857} = 2.198 \text{ m}$

$\therefore \quad GM = \frac{I}{\forall} - BG = 2.198 - 2.078 = 0.12 \text{ m.}$

$\therefore$ Meta-centric height of both the pontoon and boiler = **0.12 m. Ans.**

Problem 4.15 *A wooden cylinder of sp. gr. = 0.6 and circular in cross-section is required to float in oil (sp. gr. = 0.90). Find the L/D ratio for the cylinder to float with its longitudinal axis vertical in oil, where L is the height of cylinder and D is its diameter.*

Solution. Given :

Dia. of cylinder $= D$

Height of cylinder $= L$

Sp. gr. of cylinder, $S_1 = 0.6$

Sp. gr. of oil $S_2 = 0.9$

Let the depth of cylinder immersed in oil = h

For the principle of buoyancy

Weight of cylinder = wt. of oil displaced

$$\frac{\pi}{4} D^2 \times L \times 0.6 \times 1000 \times 9.81 = \frac{\pi}{4} D^2 \times h \times 0.9 \times 1000 \times 9.81$$

or $$L \times 0.6 = h \times 0.9$$

$$\therefore \quad h = \frac{0.6 \times L}{0.9} = \frac{2}{3} L.$$

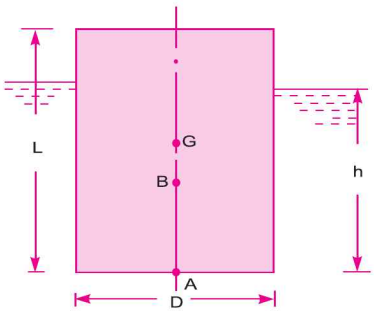


Fig. 4.18

The distance of centre of gravity G from A, $AG = \frac{L}{2}$

The distance of centre of buoyancy B from A,

$$AB = \frac{h}{2} = \frac{1}{2}\left[\frac{2}{3} L\right] = \frac{L}{3}$$

$$\therefore \quad BG = AG - AB = \frac{L}{2} - \frac{L}{3} = \frac{3L - 2L}{6} = \frac{L}{6}$$

The meta-centric height GM is given by

$$GM = \frac{I}{\forall} - BG$$

where $I = \frac{\pi}{64} D^4$ and $\forall$ = Volume of cylinder in oil = $\frac{\pi}{4} D^2 \times h$

$$\therefore \quad \frac{I}{\forall} = \left(\frac{\pi}{64} D^4 \Big/ \frac{\pi}{4} D^2 h\right) = \frac{1}{16} \frac{D^2}{h} = \frac{D^2}{16 \times \frac{2}{3} L} = \frac{3D^2}{32L} \qquad \left\{\because \ h = \frac{2}{3} L\right\}$$

$$\therefore \quad GM = \frac{3D^2}{32L} - \frac{L}{6}.$$

For stable equilibrium, GM should be +ve or

$$GM > 0 \quad \text{or} \quad \frac{3D^2}{32L} - \frac{L}{6} > 0$$

or $$\frac{3D^2}{32L} > \frac{L}{6} \quad \text{or} \quad \frac{3 \times 6}{32} > \frac{L^2}{D^2}$$

or $$\frac{L^2}{D^2} < \frac{18}{32} \quad \text{or} \quad \frac{9}{16}$$

$$\therefore \quad \frac{L}{D} < \sqrt{\frac{9}{16}} = \frac{3}{4}$$

$$\therefore \quad \textbf{L/D < 3/4. Ans.}$$

Problem 4.16 *Show that a cylindrical buoy of 1 m diameter and 2.0 m height weighing 7.848 kN will not float vertically in sea water of density 1030 kg/m^3. Find the force necessary in a vertical chain attached at the centre of base of the buoy that will keep it vertical.*

Solution. Given : Dia. of buoy, $D = 1$ m

Height, $H = 2.0$ m

Weight, $W = 7.848$ kN

$= 7.848 \times 1000 = 7848$ N

Density, $\rho = 1030$ kg/m^3

(*i*) Show the cylinder will not float vertically.

(*ii*) Find the force in the chain.

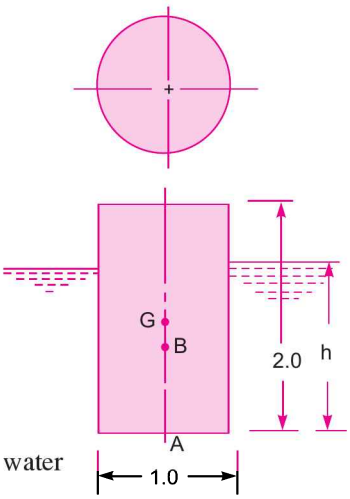


Fig. 4.19

Part I. The cylinder will not float if meta-centric height is –ve.

Let the depth of immersion be h

Then for equilibrium, Weight of cylinder

= Weight of water displaced

= Density $\times\, g \times$ Volume of cylinder in water

$$\therefore \quad 7848 = 1030 \times 9.81 \times \frac{\pi}{4} D^2 \times h$$

$$= 10104.3 \times \frac{\pi}{4} (1)^2 \times h$$

$$\therefore \quad h = \frac{4 \times 7848}{10104.3 \times \pi} = 0.989 \text{ m.}$$

$\therefore$ The distance of centre of buoyancy B from A,

$$AB = \frac{h}{2} = \frac{0.989}{2} = 0.494 \text{ m.}$$

And the distance of centre of gravity G, from A is $AG = \dfrac{2.0}{2} = 1.0$ m

$$\therefore \quad BG = AG - AB = 1.0 - .494 = .506 \text{ m.}$$

Now meta-centric height GM is given by $GM = \dfrac{I}{\forall} - BG$

where $I = \dfrac{\pi}{64} D^4 = \dfrac{\pi}{64} \times (1)^4$ m^4

and $\forall$ = Volume of cylinder in water $= \dfrac{\pi}{4} D^2 \times h = \dfrac{\pi}{4} 1^2 \times .989$

$$\therefore \quad \frac{I}{\forall} = \frac{\frac{\pi}{64} \times 1^4}{\frac{\pi}{4} D^2 \times h} = \frac{\frac{\pi}{64} \times 1^4}{\frac{\pi}{4} \times 1^2 \times .989}$$

$$= \frac{1}{16} \times 1^2 \times \frac{1}{.989} = \frac{1}{16 \times .989} = 0.063 \text{ m}$$

$$\therefore \quad GM = .063 - .506 = \mathbf{-\ 0.443\ m.\ Ans.}$$

As the meta-centric height is –ve, the point M lies below G and hence the cylinder will be in unstable equilibrium and hence cylinder will not float vertically.

Part II. Let the force applied in a vertical chain attached at the centre of the base of the buoy is T to keep the buoy vertical.

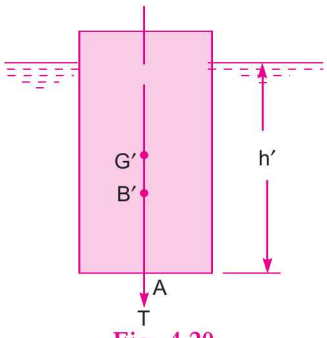


Fig. 4.20

Now find the combined position of centre of gravity (G') and centre of buoyancy (B'). For the combined centre of buoyancy, let h' = depth of immersion when the force T is applied. Then

Total downward force = Weight of water displaced

or $(7848 + T)$ = Density of water $\times\ g \times$ Volume of cylinder in water

$$= 1030 \times 9.81 \times \frac{\pi}{4} D' \times h' \text{ [where } h' = \text{depth of immersion]}$$

$$\therefore \qquad h' = \frac{7848 + T}{10104.3 \times \frac{\pi}{4} \times D^2} = \frac{7848 + T}{10104.3 \times \frac{\pi}{4} \times 1^2} = \frac{10104.3 + T}{7935.9} \text{ m}$$

$$\therefore \qquad AB' = \frac{h'}{2} = \frac{1}{2}\left[\frac{7848 + T}{7935.9}\right] = \frac{7848 + T}{15871.8} \text{ m.}$$

The combined centre of gravity (G') due to weight of cylinder and due to tension T in the chain from A is

$$AG' = [\text{Wt. of cylinder} \times \text{Distance of C.G. of cylinder from } A + T \times \text{Distance of C.G. of } T \text{ from } A] + [\text{Weight of cylinder} + T]$$

$$= \left(7848 \times \frac{2}{2} + T \times 0\right) + [7848 + T] = \frac{7848}{7848 + T} \text{ m}$$

$$\therefore \qquad B'G' = AG' - AB' = \frac{7848}{(7848 + T)} - \frac{(7848 + T)}{15871.8}$$

The meta-centric height GM is given by $GM = \dfrac{I}{\forall} - B'G'$

where $$= \frac{\pi}{64} \times D^4 = \frac{\pi}{64} \times 1^4 = \frac{\pi}{64} \text{ m}^4$$

and $$\forall = \frac{\pi}{4} D^2 \times h' = \frac{\pi}{4} \times 1^2 \times \frac{(7848 + T)}{7935.9} = \frac{\pi}{4} \times \frac{7848 + T}{7935.9}$$

$$\therefore \qquad \frac{I}{\forall} = \frac{\frac{\pi}{64}}{\frac{\pi}{4}\frac{(7848 + T)}{7935.9}} = \frac{1}{16} \times \frac{7935.9}{(7848 + T)}$$

$$\therefore \qquad GM = \frac{7935.9}{16\,(7848 + T)} - \left[\frac{7848}{(7848 + T)} - \frac{(7848 + T)}{15871.8}\right]$$

For stable equilibrium GM should be positive

or $$GM > 0$$

or $$\frac{7935.9}{16\,(7848 + T)} - \left[\frac{7848}{(7848 + T)} - \frac{(7848 + T)}{15871.8}\right] > 0$$

or $$\frac{7935.9}{16(7848+T)} - \frac{7848}{(7848+T)} + \frac{7848+T}{15871.8} > 0$$

or $$\frac{7935.9 - 16 \times 7848}{16(7848+T)} + \frac{(7848+T)}{15871.8} > 0$$

or $$\frac{-117632}{16(7848+T)} + \frac{(7848+T)}{15871.8} > 0$$

or $$\frac{(7848+T)}{15871.8} > \frac{117632}{16(7848+T)}$$

or $$(7848 + T)^2 > \frac{117632}{16.0} \times 15871.8$$

$$> 116689473.5$$

$$> (10802.3)^2$$

$\therefore$ $$7848 + T > 10802.3$$

$\therefore$ $$T > 10802.3 - 7848$$

$$\mathbf{> 2954.3\ N.\ Ans.}$$

$\therefore$ The force in the chain must be at least 2954.3 N so that the cylindrical buoy can be kept in vertical position. **Ans.**

Problem 4.17 *A solid cone floats in water with its apex downwards. Determine the least apex angle of cone for stable equilibrium. The specific gravity of the material of the cone is given 0.8.*

Solution. Given :

Sp. gr. of cone $= 0.8$

Density of cone, $\rho = 0.8 \times 1000 = 800\ kg/m^3$

Let D = Dia. of the cone

d = Dia. of cone at water level

2θ = Apex angle of cone

H = Height of cone

h = Depth of cone in water

G = Centre of gravity of the cone

B = Centre of buoyancy of the cone

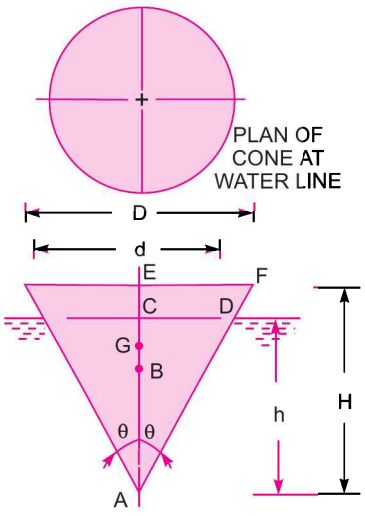


Fig. 4.21

For the cone, the distance of centre of gravity from the apex A is

$$AC = \tfrac{3}{4} \text{ height of cone} = \tfrac{3}{4} H$$

also $$AB = \tfrac{3}{4} \text{ depth of cone in water} = \tfrac{3}{4} h$$

Volume of water displaced $= \frac{1}{3} \pi r^2 \times h$

Volume of cone $= \frac{1}{3} \times \pi R^2 \times h$

$\therefore$ Weight of cone $= 800 \times g \times \frac{1}{3} \times \pi R^2 \times H$

Now from ΔAEF, $$\tan\theta = \frac{EF}{EA} = \frac{R}{H}$$

$\therefore$ $$R = H \tan\theta$$

Similarly, $$r = h \tan\theta$$

$\therefore$ Weight of cone $= 800 \times g \times \frac{1}{3} \times \pi \times (H \tan \theta)^2 \times H = \dfrac{800 \times g \times \pi \times H^3 \tan^2 \theta}{3}$

$\therefore$ Weight of water displaced $= 1000 \times g \times \frac{1}{3} \times \pi r^2 \times h$

$= 1000 \times g \times \frac{1}{3} \times \pi (h \tan \theta)^2 \times h = \dfrac{1000 \times g \times \pi \times h^3 \tan^2 \theta}{3.0}$

For equilibrium

Weight of cone = Weight of water displaced

or $$\frac{800 \times g \times \pi \times H^3 \tan^2 \theta}{3.0} = \frac{1000 \times 9.81 \times \pi \times h^3 \times \tan^2 \theta}{3.0}$$

or $$800 \times H^3 = 1000 \times h^3$$

$\therefore$ $$H^3 = \frac{1000}{800} \times h^3 \text{ or } \frac{H}{h} = \left(\frac{1000}{800}\right)^{1/3}$$

For stable equilibrium, Meta-centric height GM should be positive. But GM is given by

$$GM = \frac{I}{\forall} - BG$$

where I = M.O.I. of cone at water-line $= \dfrac{\pi}{64} d^4$

$\forall$ = Volume of cone in water $= \dfrac{1}{3} \dfrac{\pi}{4} d^2 \times h$

$\therefore$ $$\frac{I}{\forall} = \frac{\pi}{64} d^4 \Big/ \frac{1}{3} \times \frac{\pi}{4} d^2 \times h$$

$$= \frac{1 \times 3}{16} \times \frac{d^2}{h} = \frac{3d^2}{16h} = \frac{3}{16h} \times (2r)^2 = \frac{3}{4} \frac{r^2}{h}$$

$$= \frac{3}{4} \frac{(h \tan \theta)^2}{h} \qquad \{\because \quad r = h \tan \theta\}$$

$$= \tfrac{3}{4} h \tan^2 \theta$$

and $$BG = AG - AB = \tfrac{3}{4} H - \tfrac{3}{4} h = \tfrac{3}{4} (H - h)$$

$\therefore$ $$GM = \tfrac{3}{4} h \tan^2 \theta - \tfrac{3}{4} (H - h)$$

For stable equilibrium GM should be positive or

$$\tfrac{3}{4} h \tan^2 \theta - \tfrac{3}{4} (H - h) > 0 \quad \text{or} \quad h \tan^2 \theta - (H - h) > 0$$

or $$h \tan^2 \theta > (H - h) \quad \text{or} \quad h \tan^2 \theta + h > H$$

or $$h[\tan^2 \theta + 1] > H \quad \text{or} \quad 1 + \tan^2 \theta > H/h \quad \text{or} \quad \sec^2 \theta > \frac{H}{h}$$

But $$\frac{H}{h} = \left(\frac{1000}{800}\right)^{1/3} = 1.077$$

$\therefore$ $$\sec^2 \theta > 1.077 \text{ or } \cos^2 \theta > \frac{1}{1.077} = 0.9285$$

$\therefore$ $$\cos \theta > 0.9635$$

$\therefore$ $$\theta > 15^\circ\, 30' \quad \text{or} \quad 2\theta > 31^\circ$$

$\therefore$ Apex angle (2θ) should be at least **31°. Ans.**

Problem 4.18 *A cone of specific gravity S, is floating in water with its apex downwards. It has a diameter D and vertical height H. Show that for stable equilibrium of the cone* $H < \frac{1}{2}\left[\frac{D^2 . S^{1/3}}{2 - S^{1/3}}\right]^{1/2}$.

Solution. Given :

Dia. of cone = D

Height of cone = H

Sp. gr. of cone = S

Let G = Centre of gravity of cone

B = Centre of buoyancy

2θ = Apex angle

A = Apex of the cone

h = Depth of immersion

d = Dia. of cone at water surface

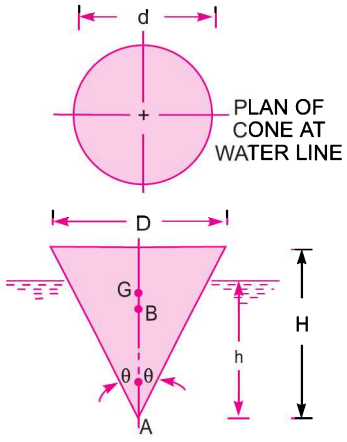


Fig. 4.22

Then $AG = \frac{3}{4} H$

$AB = \frac{3}{4} h$

Also weight of cone =Weight of water displaced.

$$1000\, S \times g \times \tfrac{1}{3}\pi R^2 \times \mathrm{H} = 1000 \times g \times \frac{1}{3}\pi r^2 \times h \quad \text{or} \quad SR^2H = r^2h$$

$\therefore$ $$h = \frac{SR^2H}{r^2}$$

But $$\tan\theta = \frac{R}{H} = \frac{r}{h}$$

$\therefore$ $$R = H \tan\theta,\ r = h \tan\theta$$

$\therefore$ $$h = \frac{S \times (H \tan\theta)^2 \times H}{(h \tan\theta)^2}$$

$$h = \frac{S \times H^2 \times \tan^2\theta \times H}{h^2 \tan^2\theta} = \frac{SH^3}{h^2} \quad \text{or} \quad h^3 = SH^3$$

or $$h = (SH^3)^{1/3} = S^{1/3} H \qquad ...(1)$$

Distance, $$BG = AG - AB$$

$$= \frac{3}{4} H - \frac{3}{4} h = \frac{3}{4} (H - h) = \frac{3}{4} (H - S^{1/3} H) \qquad \{\because h = S^{1/3} H\}$$

$$= \frac{3}{4} H[1 - S^{1/3}] \qquad ...(2)$$

Also I = M.O. Inertia of the plan of body at water surface

$$= \frac{\pi}{64} d^4$$

$\forall$ = Volume of cone in water = $\frac{1}{3} \times \frac{\pi}{4} \times d^2 \times h = \frac{1}{3}\frac{\pi}{4} d^2\ [H.S^{1/3}]$

$$\therefore \quad \frac{I}{\forall} = \frac{\frac{\pi}{64} d^4}{\frac{1}{3} \times \frac{\pi}{4} d^2 \, H.S^{1/3}} = \frac{3d^2}{16.H.S^{1/3}}$$

Now Meta-centric height GM is given as

$$GM = \frac{I}{\forall} - BG = \frac{3d^2}{16.H.S^{1/3}} - \frac{3H}{4} [1 - S^{1/3}]$$

GM should be +ve for stable equilibrium or $GM > 0$

or $$\frac{3d^2}{16.H.S^{1/3}} - \frac{3H}{4} (1 - S^{1/3}) > 0$$

or $$\frac{3d^2}{16.H.S^{1/3}} > \frac{3H}{4} (1 - S^{1/3}) \qquad ...(3)$$

Also we know $R = H \tan \theta$ and $r = h \tan \theta$

$$\therefore \quad \frac{R}{r} = \frac{H}{h} = \frac{D}{d}$$

$$\therefore \quad d = \frac{Dh}{H} = \frac{D}{H} \times HS^{1/3} = DS^{1/3}$$

Substituting the value of d in equation (3), we get

$$\frac{3\left(DS^{1/3}\right)^2}{16.H.S^{1/3}} > \frac{3H}{4} (1 - S^{1/3}) \quad \text{or} \quad \frac{D^2.S^{1/3}}{4.H} > H (1 - S^{1/3})$$

or $$\frac{D^2.S^{1/3}}{4\left(1 - S^{1/2}\right)} > H^2 \quad \text{or} \quad H^2 < \frac{D^2.S^{1/3}}{4\left(1 - S^{1/3}\right)}$$

or $$H < \frac{1}{2}\left[\frac{D^2.S^{1/3}}{1 - S^{1/3}}\right]^{1/2} \text{. Ans.}$$

▶ 4.8 EXPERIMENTAL METHOD OF DETERMINATION OF META-CENTRIC HEIGHT

The meta-centric height of a floating vessel can be determined, provided we know the centre of gravity of the floating vessel. Let w_1 is a known weight placed over the centre of the vessel as shown in Fig. 4.23 (*a*) and the vessel is floating.

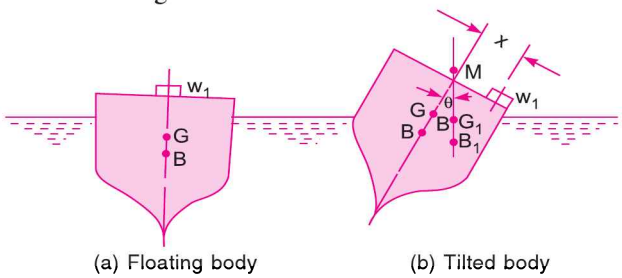


Fig. 4.23 *Meta-centric height.*

Let W = Weight of vessel including w_1

G = Centre of gravity of the vessel

B = Centre of buoyancy of the vessel

The weight w_1 is moved across the vessel towards right through a distance x as shown in Fig. 4.23 (*b*). The vessel will be tilted. The angle of heel θ is measured by means of a plumbline and a protractor attached on the vessel. The new centre of gravity of the vessel will shift to G_1 as the weight w_1 has been moved towards the right. Also the centre of buoyancy will change to B_1 as the vessel has tilted. Under equilibrium, the moment caused by the movement of the load w_1 through a distance x must be equal to the moment caused by the shift of the centre of gravity from G to G_1. Thus

The moment due to change of $G = GG_1 \times W = W \times GM \tan \theta$

The moment due to movement of $w_1 = w_1 \times x$

$\therefore \quad w_1 x = WGM \tan \theta$

Hence
$$GM = \frac{w_1 x}{W \tan \theta} \qquad ...(4.5)$$

Problem 4.19 *A ship 70 m long and 10 m broad has a displacement of 19620 kN. A weight of 343.35 kN is moved across the deck through a distance of 6 m. The ship is tilted through 6°. The moment of inertia of the ship at water-line about its fore and aft axis is 75% of M.O.I. of the circumscribing rectangle. The centre of buoyancy is 2.25 m below water-line. Find the meta-centric height and position of centre of gravity of ship. Specific weight of sea water is 10104 N/m^3.*

Solution. Given :

Length of ship, $L = 70$ m

Breadth of ship, $b = 10$ m

Displacement, $W = 19620$ kN

Angle of heel, $\theta = 6°$

M.O.I. of ship at water-line = 75% of M.O.I. of circumscribing rectangle

w for sea-water = 10104 N/m^3 = 10.104 kN/m^3

Movable weight, $w_1 = 343.35$ kN

Distance moved by w_1, $x = 6$ m

Centre of buoyancy = 2.25 m below water surface

Find (*i*) Meta-centric height, *GM*

(*ii*) Position of centre of gravity, *G*.

(*i*) **Meta-centric height, GM** is given by equation (4.5)

$$\therefore \quad GM = \frac{w_1 x}{W \tan \theta} = \frac{343.35 \text{ kN} \times 6.0}{19620 \text{ kN} \times \tan 6°}$$

$$= \frac{343.35 \text{ kN} \times 6.0}{19620 \text{ kN} \times .1051} = \mathbf{0.999 \text{ m. Ans.}}$$

(*ii*) **Position of Centre of Gravity, G**

$$GM = \frac{I}{\forall} - BG$$

where I = M.O.I. of the ship at water-line about *Y-Y*

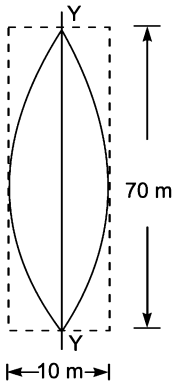


Fig. 4.24

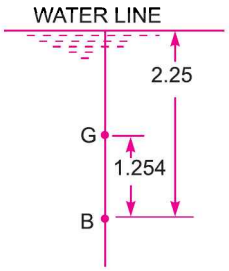


Fig. 4.25

$$= 75\% \text{ of } \frac{1}{12} \times 70 \times 10^3 = .75 \times \frac{1}{12} \times 70 \times 10^3 = 4375 \text{ m}^4$$

and $\forall$ = Volume of ship in water = $\dfrac{\text{Weight of ship}}{\text{Weight density of water}} = \dfrac{19620}{10.104} = 1941.74 \text{ m}^3$

$$\therefore \quad \frac{I}{\forall} = \frac{4375}{1941.74} = 2.253 \text{ m}$$

$$\therefore \quad GM = 2.253 - BG \text{ or } .999 = 2.253 - BG$$

$$\therefore \quad BG = 2.253 - .999 = 1.254 \text{ m.}$$

From Fig. 4.25, it is clear that the distance of G from free surface of the water = distance of B from water surface – BG

$$= 2.25 - 1.254 = \mathbf{0.996 \text{ m. Ans.}}$$

Problem 4.20 *A pontoon of 15696 kN displacement is floating in water. A weight of 245.25 kN is moved through a distance of 8 m across the deck of pontoon, which tilts the pontoon through an angle 4°. Find meta-centric height of the pontoon.*

Solution. Given :

Weight of pontoon = Displacement

or $W = 15696$ kN

Movable weight, $w_1 = 245.25$ kN

Distance moved by weight w_1, $x = 8$ m

Angle of heel, $\theta = 4°$

The meta-centric height, GM is given by equation (4.5)

or $$GM = \frac{w_1 x}{W \tan \theta} = \frac{245.25 \text{ kN} \times 8}{15696 \text{ kN} \times \tan 4°}$$

$$= \frac{1962}{15696 \times 0.0699} = \mathbf{1.788 \text{ m. Ans.}}$$

▶ 4.9 OSCILLATION (ROLLING) OF A FLOATING BODY

Consider a floating body, which is tilted through an angle by an overturning couple as shown in Fig. 4.26. Let the overturning couple is suddenly removed. The body will start oscillating. Thus, the

body will be in a state of oscillation as if suspended at the meta-centre M. This is similar to the case of a pendulum. The only force acting on the body is due to the restoring couple due to the weight W of the body force of buoyancy F_B.

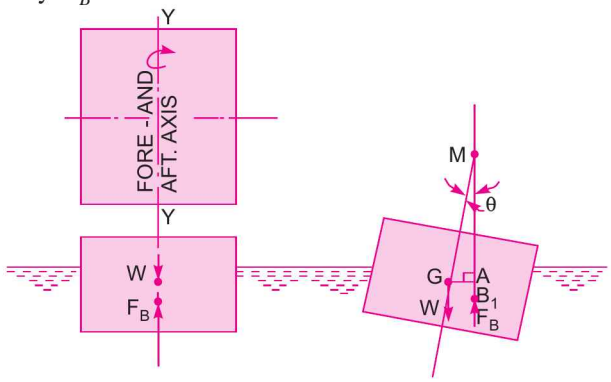


Fig. 4.26

$\therefore$ Restoring couple $= W \times$ Distance GA

$= W \times GM \sin \theta$ ...(*i*)

This couple tries to decrease the angle

Angular acceleration of the body, $\alpha = -\dfrac{d^2\theta}{dt^2}$.

–ve sign has been introduced as the restoring couple tries to decrease the angle θ.

Torque due to inertia = Moment of Inertia about Y-$Y \times$ Angular acceleration

$$= I_{Y\text{-}Y} \times \left(-\frac{d^2\theta}{dt^2}\right)$$

But $$I_{Y\text{-}Y} = \frac{W}{g} K^2$$

where W = Weight of body, K = Radius of gyration about Y-Y

$\therefore$ Inertia torque $$= \frac{W}{g} K^2 \left(-\frac{d^2\theta}{dt^2}\right) = -\frac{W}{g} K^2 \frac{d^2\theta}{dt^2}$$ ...(*ii*)

Equating (*i*) and (*ii*), we get

$$W \times GM \sin \theta = -\frac{W}{g} K^2 \frac{d^2\theta}{dt^2} \quad \text{or} \quad GM \sin \theta = -\frac{K^2}{g} \frac{d^2\theta}{dt^2}$$

For small angle θ, $\sin \theta \simeq \theta$

$\therefore$ $$GM \times \theta = -\frac{K^2}{g} \frac{d^2\theta}{dt^2} \quad \text{or} \quad \frac{K^2}{g} \frac{d^2\theta}{dt^2} + GM \times \theta = 0$$

Dividing by $\dfrac{K^2}{g}$, we get $\dfrac{d^2\theta}{dt^2} + \dfrac{GM \times g \times \theta}{K^2} = 0$

The above equation is a differential equation of second degree. The solution is

$$\theta = C_1 \sin \sqrt{\frac{GM.g}{K^2}} \times t + C_2 \cos \sqrt{\frac{GM.g \times t}{K^2}}$$ ...(*iii*)

where C_1 and C_2 are constants of integration.

The values of C_1 and C_2 are obtained from boundary conditions which are

(*i*) at $t = 0$, $\theta = 0$

(*ii*) at $t = \frac{T}{2}$, $\theta = 0$

where T is the time period of one complete oscillation.

Substituting the 1st boundary condition in (*iii*), we get

$$0 = C_1 \times 0 + C_2 \times 1 \qquad \{\because \sin\theta = 0, \cos\theta = 1\}$$

$$\therefore \qquad C_2 = 0$$

Substituting 2nd boundary conditions in (*iii*), we get

$$0 = C_1 \sin \sqrt{\frac{GM.g}{K^2}} \times \frac{T}{2}$$

But C_1 cannot be equal to zero and so the other alternative is

$$\sin \sqrt{\frac{GM.g}{K^2}} \times \frac{T}{2} = 0 = \sin \pi \qquad \{\because \sin \pi = 0\}$$

$$\therefore \qquad \sqrt{\frac{GM.g}{K^2}} \times \frac{T}{2} = \pi \quad \text{or} \quad T = 2\pi \sqrt{\frac{K^2}{GM.g}} \qquad \ldots(4.6)$$

$\therefore$ Time period of oscillation is given by equation (4.6).

Problem 4.21 *The least radius of gyration of a ship is 8 m and meta-centric height 70 cm. Calculate the time period of oscillation of the ship.*

Solution. Given :

Least radius of gyration, $K = 8$ m

Meta-centric height, $GM = 70$ cm $= 0.70$ m

The time period of oscillation is given by equation (4.6).

$$T = 2\pi \sqrt{\frac{K^2}{GM.g}} = 2\pi \sqrt{\frac{8 \times 8}{0.7 \times 9.81}} = \mathbf{19.18\ sec.\ Ans.}$$

Problem 4.22 *The time period of rolling of a ship of weight 29430 kN in sea water is 10 seconds. The centre of buoyancy of the ship is 1.5 m below the centre of gravity. Find the radius of gyration of the ship if the moment of inertia of the ship at the water line about fore and aft axis is 1000 m^4. Take specific weight of sea water as = 10100 N/m^3.*

Solution. Given :

Time period, $T = 10$ sec

Distance between centre of buoyancy and centre of gravity, $BG = 1.5$ m

Moment of Inertia, $I = 10000\ m^4$

Weight, $W = 29430$ kN $= 29430 \times 1000$ N

Let the radius of gyration $= K$

First calculate the meta-centric height GM, which is given as

$$GM = BM - BG = \frac{I}{\forall} - BG$$

where I = M.O. Inertia

and $\forall$ = Volume of water displaced

$$= \frac{\text{Weight of ship}}{\text{Sp. weight of sea water}} = \frac{29430 \times 1000}{10104} = 2912.6 \text{ m}^3$$

$$\therefore \quad GM = \frac{10000}{2912.6} - 1.5 = 3.433 - 1.5 = 1.933 \text{ m.}$$

Using equation (4.6), we get $T = 2\pi \sqrt{\frac{K^2}{GM \times g}}$

or $$10 = 2\pi \sqrt{\frac{K^2}{1.933 \times 9.81}} = \frac{2\pi K}{\sqrt{1.933 \times 9.81}}$$

or $$K = \frac{10 \times \sqrt{1.933 \times 9.81}}{2\pi} = \textbf{6.93 m. Ans.}$$

HIGHLIGHTS

1. The upward force exerted by a liquid on a body when the body is immersed in the liquid is known as buoyancy or force of buoyancy.
2. The point through which force of buoyancy is supposed to act is called centre of buoyancy.
3. The point about which a body starts oscillating when the body is tilted is known as meta-centre.
4. The distance between the meta-centre and centre of gravity is known as meta-centric height.
5. The meta-centric height (GM) is given by $GM = \frac{I}{\forall} - BG$

   where I = Moment of Inertia of the floating body (in plan) at water surface about the axis Y-Y

   $\forall$ = Volume of the body sub-merged in water

   BG = Distance between centre of gravity and centre of buoyancy.
6. Conditions of equilibrium of a floating and sub-merged body are :

Equilibrium	*Floating Body*	*Sub-merged Body*
(*i*) Stable Equilibrium	M is above G	B is above G
(*ii*) Unstable Equilibrium	M is below G	B is below G
(*iii*) Neutral Equilibrium	M and G coincide	B and G coincide

7. The value of meta-centric height GM, experimentally is given as $GM = \frac{w_1 x}{W \tan \theta}$

   where w_1 = Movable weight

   x = Distance through which w_1 is moved

   W = Weight of the ship or floating body including w_1

   θ = Angle through the ship or floating body is tilted due to the movement of w_1.
8. The time period of oscillation or rolling of a floating body is given by $T = 2\pi \sqrt{\frac{K^2}{GM \times g}}$

   where K = Radius of gyration, GM = Meta-centric height

   T = Time of one complete oscillation.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define the terms 'buoyancy' and 'centre of buoyancy'.
2. Explain the terms 'meta-centre' and 'meta-centric height'.
3. Derive an expression for the meta-centric height of a floating body.
4. Show that the distance between the meta-centre and centre of buoyancy is given by $BM = \frac{I}{\forall}$

   where I = Moment of inertia of the plan of the floating body at water surface about longitudinal axis.

   $\forall$ = Volume of the body sub-merged in liquid.
5. What are the conditions of equilibrium of a floating body and a sub-merged body ?
6. How will you determine the meta-centric height of a floating body experimentally ? Explain with neat sketch.
7. Select the correct statement :

   (*a*) The buoyant force for a floating body passes through the

   (*i*) centre of gravity of the body (*ii*) centroid of volume of the body

   (*iii*) meta-centre of the body (*iv*) centre of gravity of the sub-merged part of the body

   (*v*) centroid of the displaced volume.

   (*b*) A body sub-merged in liquid is in equilibrium when :

   (*i*) its meta-centre is above the centre of gravity

   (*ii*) its meta-centre is above the centre of buoyancy

   (*iii*) its centre of gravity is above the centre of buoyancy

   (*iv*) its centre of buoyancy is above the centre of gravity

   (*v*) none of these. [**Ans.** 7 (*a*) (*v*), (*b*) (*iv*)]
8. Derive an expression for the time period of the oscillation of a floating body in terms of radius of gyration and meta-centric height of the floating body.
9. Define the terms : meta-centre, centre of buoyancy, meta-centric height, gauge pressure and absolute pressure.
10. What do you understand by the hydrostatic equation ? With the help of this equation, derive the expression for the buoyant force acting on a sub-merged body.
11. With neat sketches, explain the conditions of equilibrium for floating and sub-merged bodies.
12. Differentiate between :

    (*i*) Dynamic viscosity and kinematic viscosity, (*ii*) Absolute and gauge pressure (*iii*) Simple and differential manometers (*iv*) Centre of gravity and centre of buoyancy.

    (*Delhi University, Dec. 2002*)

(B) NUMERICAL PROBLEMS

1. A wooden block of width 2 m, depth 1.5 m and length 4 m floats horizontally in water. Find the volume of water displaced and position of centre of buoyancy. The specific gravity of the wooden block is 0.7.

   [**Ans.** 8.4 m^3, 0.525 m from the base]

2. A wooden log of 0.8 m diameter and 6 m length is floating in river water. Find the depth of wooden log in water when the sp. gr. of the wooden log is 0.7. [**Ans.** 0.54 m]

3. A stone weighs 490.5 N in air and 196.2 N in water. Determine the volume of stone and its specific gravity. [**Ans.** 0.03 m^3 or 3×10^4 cm^3, 1.67]

4. A body of dimensions 2.0 m × 1.0 m × 3.0 m weighs 3924 N in water. Find its weight in air. What will be its specific gravity ? [**Ans.** 62784 N, 1.0667]

5. A metallic body floats at the interface of mercury of sp. gr. 13.6 and water in such a way that 30% of its volume is sub-merged in mercury and 70% in water. Find the density of the metallic body. [**Ans.** 4780 kg/m^3]

6. A body of dimensions 0.5 m × 0.5 m × 1.0 m and of sp. gr. 3.0 is immersed in water. Determine the least force required to lift the body. [**Ans.** 4905 N]

7. A rectangular pontoon is 4 m long, 3 m wide and 1.40 m high. The depth of immersion of the pontoon is 1.0 m in sea-water. If the centre of gravity is 0.70 m above the bottom of the pontoon, determine the meta-centric height. Take the density of sea-water as 1030 kg/m^3. [**Ans.** 0.45 m]

8. A uniform body of size 4 m long × 2 m wide × 1 m deep floats in water. What is the weight of the body if depth of immersion is 0.6 m ? Determine the meta-centric height also. [**Ans**. 47088 N, 0.355 m]

9. A block of wood of specific gravity 0.8 floats in water. Determine the meta-centric height of the block if its size is 3 m × 2 m × 1 m. [**Ans.** 0.316 m]

10. A solid cylinder of diameter 3.0 m has a height of 2 m. Find the meta-centric height of the cylinder when it is floating in water with its axis vertical. The sp. gr. of the cylinder is 0.7. [**Ans.** 0.1017 m]

11. A body has the cylindrical upper portion of 4 m diameter and 2 m deep. The lower portion is a curved one, which displaces a volume of 0.9 m^3 of water. The centre of buoyancy of the curved portion is at a distance of 2.10 m below the top of the cylinder. The centre of gravity of the whole body is 1.50 m below the top of the cylinder. The total displacement of water is 4.5 tonnes. Find the meta-centric height of the body. [**Ans.** 2.387 m]

12. A solid cylinder of diameter 5.0 m has a height of 5.0 m. Find the meta-centric height of the cylinder if the specific gravity of the material of cylinder is 0.7 and it is floating in water with its axis vertical. State whether the equilibrium is stable or unstable. [**Ans.** – 0.304 m, Unstable Equilibrium]

13. A solid cylinder of 15 cm diameter and 60 cm long, consists of two parts made of different materials. The first part at the base is 1.20 cm long and of specific gravity = 5.0. The other parts of the cylinder is made of the material having specific gravity 0.6. State, if it can float vertically in water. [**Ans.** $GM = -5.26$, Unstable, Equilibrium]

14. A rectangular pontoon 8.0 m long, 7 m broad and 3.0 m deep weighs 588.6 kN. It carries on its upper deck an empty boiler of 4.0 m diameter weighing 392.4 kN. The centre of gravity of the boiler and the pontoon are at their respective centres along a vertical line. Find the meta-centric height. Weight density of sea-water is 10104 N/m^3. [**Ans.** 0.325 m]

15. A wooden cylinder of sp. gr. 0.6 and circular in cross-section is required to float in oil (sp. gr. 0.8). Find the L/D ratio for the cylinder to float with its longitudinal axis vertical in oil where L is the height of cylinder and D is its diameter. [**Ans.** $(L/D) < 0.8164$]

16. Show that a cylindrical buoy of 1.5 m diameter and 3 m long weighing 2.5 tonnes will not float vertically in sea-water of density 1030 kg/m^3. Find the force necessary in a vertical chain attached at the centre of the base of the buoy that will keep it vertical. [**Ans.** 10609.5 N]

17. A solid cone floats in water its apex downwards. Determine the least apex angle of cone for stable equilibrium. The specific gravity of the material of the cone is given 0.7. [**Ans.** 39° 7′]

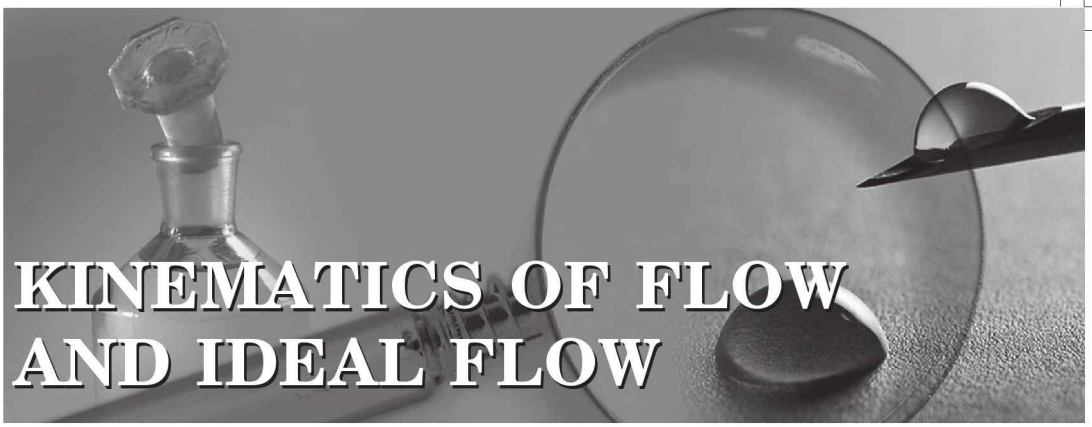
18. A ship 60 m long and 12 m broad has a displacement of 19620 kN. A weight of 294.3 kN is moved across the deck through a distance of 6.5 m. The ship is tilted through 5°. The moment of inertia of the ship at

water line about its force and aft axis is 75% of moment of inertia the circumscribing rectangle. The centre of buoyancy is 2.75 m below water line. Find the meta-centric height and position of centre of gravity of ship. Take specific weight of sea water = 10104 N/m^3. [**Ans.** 1.1145 m, 0.53 m below water surface]

19. A pontoon of 1500 tonnes displacement is floating in water. A weight of 20 tonnes is moved through a distance of 6 m across the deck of pontoon, which tilts the pontoon through an angle of 5°. Find meta-centric height of the pontoon. [**Ans.** 0.9145 m]

20. Find the time period of rolling of a solid circular cylinder of radius 2.5 m and 5.0 m long. The specific gravity of the cylinder is 0.9 and is floating in water with its axis vertical. [**Ans.** 0.35 sec]

5 CHAPTER



KINEMATICS OF FLOW AND IDEAL FLOW

A. KINEMATICS OF FLOW

▶ 5.1 INTRODUCTION

Kinematics is defined as that branch of science which deals with motion of particles without considering the forces causing the motion. The velocity at any point in a flow field at any time is studied in this branch of fluid mechanics. Once the velocity is known, then the pressure distribution and hence forces acting on the fluid can be determined. In this chapter, the methods of determining velocity and acceleration are discussed.

▶ 5.2 METHODS OF DESCRIBING FLUID MOTION

The fluid motion is described by two methods. They are —(*i*) Lagrangian Method, and (*ii*) Eulerian Method. In the Lagrangian method, a **single fluid particle** is followed during its motion and its velocity, acceleration, density, etc., are described. In case of Eulerian method, the velocity, acceleration, pressure, density etc., are described **at a point** in flow field. The Eulerian method is commonly used in fluid mechanics.

▶ 5.3 TYPES OF FLUID FLOW

The fluid flow is classified as :

(*i*) Steady and unsteady flows ;
(*ii*) Uniform and non-uniform flows ;
(*iii*) Laminar and turbulent flows ;
(*iv*) Compressible and incompressible flows ;
(*v*) Rotational and irrotational flows ; and
(*vi*) One, two and three-dimensional flows.

5.3.1 Steady and Unsteady Flows. Steady flow is defined as that type of flow in which the fluid characteristics like velocity, pressure, density, etc., at a point do not change with time. Thus for steady flow, mathematically, we have

$$\left(\frac{\partial V}{\partial t}\right)_{x_0, y_0, z_0} = 0, \left(\frac{\partial p}{\partial t}\right)_{x_0, y_0, z_0} = 0, \left(\frac{\partial \rho}{\partial t}\right)_{x_0, y_0, z_0} = 0$$

where (x_0, y_0, z_0) is a fixed point in fluid field.

Unsteady flow is that type of flow, in which the velocity, pressure or density at a point changes with respect to time. Thus, mathematically, for unsteady flow

$$\left(\frac{\partial V}{\partial t}\right)_{x_0, y_0, z_0} \neq 0, \left(\frac{\partial p}{\partial t}\right)_{x_0, y_0, z_0} \neq 0 \text{ etc.}$$

5.3.2 Uniform and Non-uniform Flows. Uniform flow is defined as that type of flow in which the velocity at any given time does not change with respect to space (*i.e.*, length of direction of the flow). Mathematically, for uniform flow

$$\left(\frac{\partial V}{\partial s}\right)_{t = \text{constant}} = 0$$

where ∂V = Change of velocity

∂s = Length of flow in the direction S.

Non-uniform flow is that type of flow in which the velocity at any given time changes with respect to space. Thus, mathematically, for non-uniform flow

$$\left(\frac{\partial V}{\partial s}\right)_{t = \text{constant}} \neq 0.$$

5.3.3 Laminar and Turbulent Flows. Laminar flow is defined as that type of flow in which the fluid particles move along well-defined paths or stream line and all the stream-lines are straight and parallel. Thus the particles move in laminas or layers gliding smoothly over the adjacent layer. This type of flow is also called stream-line flow or viscous flow.

Turbulent flow is that type of flow in which the fluid particles move in a *zig-zag* way. Due to the movement of fluid particles in a *zig-zag* way, the eddies formation takes place which are responsible for high energy loss. For a pipe flow, the type of flow is determined by a non-dimensional number $\frac{VD}{\nu}$ called the Reynold number,

where D = Diameter of pipe

V = Mean velocity of flow in pipe

and ν = Kinematic viscosity of fluid.

If the Reynold number is less than 2000, the flow is called laminar. If the Reynold number is more than 4000, it is called turbulent flow. If the Reynold number lies between 2000 and 4000, the flow may be laminar or turbulent.

5.3.4 Compressible and Incompressible Flows. Compressible flow is that type of flow in which the density of the fluid changes from point to point or in other words the density (ρ) is not constant for the fluid. Thus, mathematically, for compressible flow

$$\rho \neq \text{Constant}$$

Incompressible flow is that type of flow in which the density is constant for the fluid flow. Liquids are generally incompressible while gases are compressible. Mathematically, for incompressible flow

$$\rho = \text{Constant.}$$

5.3.5 Rotational and Irrotational Flows. Rotational flow is that type of flow in which the fluid particles while flowing along stream-lines, also rotate about their own axis. And if the fluid particles while flowing along stream-lines, do not rotate about their own axis then that type of flow is called irrotational flow.

5.3.6 One-, Two- and Three-Dimensional Flows. **One-dimensional flow** is that type of flow in which the flow parameter such as velocity is a function of time and one space co-ordinate only, say x. For a steady one-dimensional flow, the velocity is a function of one-space-co-ordinate only. The variation of velocities in other two mutually perpendicular directions is assumed negligible. Hence mathematically, for one-dimensional flow

$$u = f(x),\ v = 0 \text{ and } w = 0$$

where u, v and w are velocity components in x, y and z directions respectively.

Two-dimensional flow is that type of flow in which the velocity is a function of time and two rectangular space co-ordinates say x and y. For a steady two-dimensional flow the velocity is a function of two space co-ordinates only. The variation of velocity in the third direction is negligible. Thus, mathematically for two-dimensional flow

$$u = f_1(x, y),\ v = f_2(x, y) \text{ and } w = 0.$$

Three-dimensional flow is that type of flow in which the velocity is a function of time and three mutually perpendicular directions. But for a steady three-dimensional flow the fluid parameters are functions of three space co-ordinates (x, y and z) only. Thus, mathematically, for three-dimensional flow

$$u = f_1(x, y, z),\ v = f_2(x, y, z) \text{ and } w = f_3(x, y, z).$$

▶ 5.4 RATE OF FLOW OR DISCHARGE (Q)

It is defined as the quantity of a fluid flowing per second through a section of a pipe or a channel. For an incompressible fluid (or liquid) the rate of flow or discharge is expressed as the volume of fluid flowing across the section per second. For compressible fluids, the rate of flow is usually expressed as the weight of fluid flowing across the section. Thus

(*i*) For liquids the units of Q are m^3/s or litres/s

(*ii*) For gases the units of Q is kgf/s or Newton/s

Consider a liquid flowing through a pipe in which

A = Cross-sectional area of pipe

V = Average velocity of fluid across the section

Then discharge $Q = A \times V.$ ...(5.1)

▶ 5.5 CONTINUITY EQUATION

The equation based on the principle of conservation of mass is called continuity equation. Thus for a fluid flowing through the pipe at all the cross-section, the quantity of fluid per second is constant. Consider two cross-sections of a pipe as shown in Fig. 5.1.

Let V_1 = Average velocity at cross-section 1-1

ρ_1 = Density at section 1-1

A_1 = Area of pipe at section 1-1

and V_2, ρ_2, A_2 are corresponding values at section, 2-2.

Then rate of flow at section 1-1 = $\rho_1 A_1 V_1$

Rate of flow at section 2-2 = $\rho_2 A_2 V_2$

According to law of conservation of mass

Rate of flow at section 1-1 = Rate of flow at section 2-2

or $$\rho_1 A_1 V_1 = \rho_2 A_2 V_2 \quad ...(5.2)$$

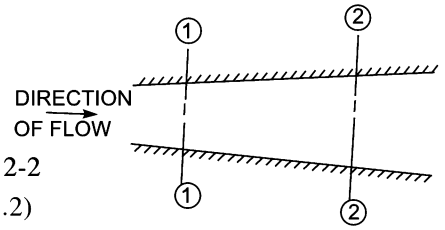


Fig. 5.1 *Fluid flowing through a pipe.*

Equation (5.2) is applicable to the compressible as well as incompressible fluids and is called **Continuity Equation**. If the fluid is incompressible, then $\rho_1 = \rho_2$ and continuity equation (5.2) reduces to

$$A_1 V_1 = A_2 V_2 \quad ...(5.3)$$

Problem 5.1 *The diameters of a pipe at the sections 1 and 2 are 10 cm and 15 cm respectively. Find the discharge through the pipe if the velocity of water flowing through the pipe at section 1 is 5 m/s. Determine also the velocity at section 2.*

Solution. Given :

At section 1, $D_1 = 10 \text{ cm} = 0.1 \text{ m}$

$$A_1 = \frac{\pi}{4}(D_1^2) = \frac{\pi}{4}(.1)^2 = 0.007854 \text{ m}^2$$

$V_1 = 5$ m/s.

At section 2, $D_2 = 15 \text{ cm} = 0.15 \text{ m}$

$$A_2 = \frac{\pi}{4}(.15)^2 = 0.01767 \text{ m}^2$$

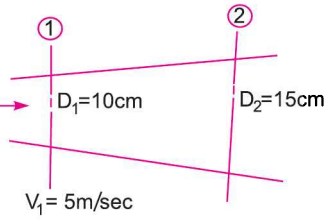


Fig. 5.2

(*i*) Discharge through pipe is given by equation (5.1)

or $$Q = A_1 \times V_1$$

$$= 0.007854 \times 5 = \mathbf{0.03927 \text{ m}^3/\text{s}}. \textbf{ Ans.}$$

Using equation (5.3), we have $A_1 V_1 = A_2 V_2$

(*ii*) $\therefore$ $$V_2 = \frac{A_1 V_1}{A_2} = \frac{0.007854}{0.01767} \times 5.0 = \mathbf{2.22 \text{ m/s}}. \textbf{ Ans.}$$

Problem 5.2 *A 30 cm diameter pipe, conveying water, branches into two pipes of diameters 20 cm and 15 cm respectively. If the average velocity in the 30 cm diameter pipe is 2.5 m/s, find the discharge in this pipe. Also determine the velocity in 15 cm pipe if the average velocity in 20 cm diameter pipe is 2 m/s.*

Solution. Given :

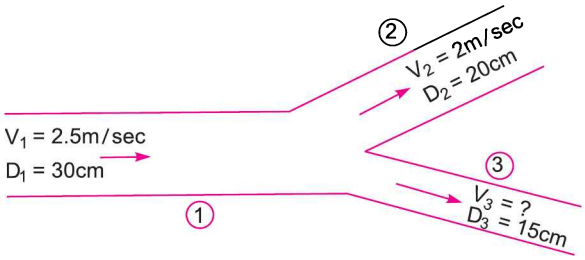


Fig. 5.3

$D_1 = 30 \text{ cm} = 0.30 \text{ m}$

$\therefore \quad A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} \times .3^2 = 0.07068 \text{ m}^2$

$V_1 = 2.5 \text{ m/s}$

$D_2 = 20 \text{ cm} = 0.20 \text{ m}$

$\therefore \quad A_2 = \frac{\pi}{4} (.2)^2 = \frac{\pi}{4} \times .4 = 0.0314 \text{ m}^2,$

$V_2 = 2 \text{ m/s}$

$D_3 = 15 \text{ cm} = 0.15 \text{ m}$

$\therefore \quad A_3 = \frac{\pi}{4} (.15)^2 = \frac{\pi}{4} \times 0.225 = 0.01767 \text{ m}^2$

Find (*i*) Discharge in pipe 1 or Q_1

(*ii*) Velocity in pipe of dia. 15 cm or V_3

Let Q_1, Q_2 and Q_3 are discharges in pipe 1, 2 and 3 respectively.

Then according to continuity equation

$$Q_1 = Q_2 + Q_3 \qquad \ldots(1)$$

(*i*) **The discharge Q_1** in pipe 1 is given by

$Q_1 = A_1 V_1 = 0.07068 \times 2.5 \text{ m}^3/\text{s} =$ **0.1767 m³/s. Ans.**

(*ii*) **Value of V_3**

$Q_2 = A_2 V_2 = 0.0314 \times 2.0 = 0.0628 \text{ m}^3/\text{s}$

Substituting the values of Q_1 and Q_2 in equation (1)

$0.1767 = 0.0628 + Q_3$

$\therefore \quad Q_3 = 0.1767 - 0.0628 = 0.1139 \text{ m}^3/\text{s}$

But $\quad Q_3 = A_3 \times V_3 = 0.01767 \times V_3 \quad \text{or} \quad 0.1139 = 0.01767 \times V_3$

$\therefore \quad V_3 = \frac{0.1139}{0.01767} =$ **6.44 m/s. Ans.**

Problem 5.3 *Water flows through a pipe AB 1.2 m diameter at 3 m/s and then passes through a pipe BC 1.5 m diameter. At C, the pipe branches. Branch CD is 0.8 m in diameter and carries one-third of the flow in AB. The flow velocity in branch CE is 2.5 m/s. Find the volume rate of flow in AB, the velocity in BC, the velocity in CD and the diameter of CE.*

Solution. Given :

Diameter of pipe *AB*, $D_{AB} = 1.2 \text{ m}$

Velocity of flow through *AB*, $V_{AB} = 3.0 \text{ m/s}$

Dia. of pipe *BC*, $D_{BC} = 1.5 \text{ m}$

Dia. of branched pipe *CD*, $D_{CD} = 0.8 \text{ m}$

Velocity of flow in pipe *CE*, $V_{CE} = 2.5 \text{ m/s}$

Let the flow rate in pipe $AB = Q \text{ m}^3/\text{s}$

Velocity of flow in pipe $BC = V_{BC} \text{ m/s}$

Velocity of flow in pipe $CD = V_{CD} \text{ m/s}$

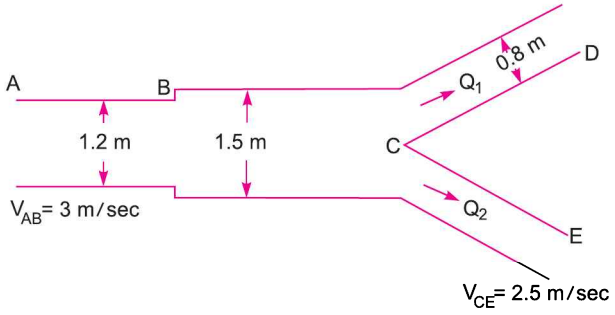


Fig. 5.4

Diameter of pipe $CE = D_{CE}$

Then flow rate through $CD = Q/3$

and flow rate through $CE = Q - Q/3 = \dfrac{2Q}{3}$

(*i*) Now volume flow rate through AB = Q = V_{AB} × Area of AB

$$= 3.0 \times \frac{\pi}{4} (D_{AB})^2 = 3.0 \times \frac{\pi}{4} (1.2)^2 = \mathbf{3.393\ m^3/s.\ Ans.}$$

(*ii*) Applying continuity equation to pipe AB and pipe BC,

V_{AB} × Area of pipe AB = V_{BC} × Area of pipe BC

or $$3.0 \times \frac{\pi}{4} (D_{AB})^2 = V_{BC} \times \frac{\pi}{4} (D_{BC})^2$$

or $$3.0 \times (1.2)^2 = V_{BC} \times (1.5)^2 \qquad \left[\text{Divide by } \frac{\pi}{4}\right]$$

or $$V_{BC} = \frac{3 \times 1.2^2}{1.5^2} = \mathbf{1.92\ m/s.\ Ans.}$$

(*iii*) The flow rate through pipe

$$CD = Q_1 = \frac{Q}{3} = \frac{3.393}{3} = 1.131\ \text{m}^3/\text{s}$$

$\therefore$ $$Q_1 = V_{CD} \times \text{Area of pipe } CD \times \frac{\pi}{4} (D_{CD})^2$$

or $$1.131 = V_{CD} \times \frac{\pi}{4} \times 0.8^2 = 0.5026\ V_{CD}$$

$\therefore$ $$V_{CD} = \frac{1.131}{0.5026} = \mathbf{2.25\ m/s.\ Ans.}$$

(*iv*) Flow rate through CE,

$$Q_2 = Q - Q_1 = 3.393 - 1.131 = 2.262\ \text{m}^3/\text{s}$$

$\therefore$ $$Q_2 = V_{CE} \times \text{Area of pipe } CE = V_{CE} \frac{\pi}{4} (D_{CE})^2$$

or $$2.263 = 2.5 \times \frac{\pi}{4} \times (D_{CE})^2$$

or $$D_{CE} = \sqrt{\frac{2.263 \times 4}{2.5 \times \pi}} = \sqrt{1.152} = 1.0735\ \text{m}$$

$\therefore$ Diameter of pipe CE = **1.0735 m. Ans.**

Problem 5.4 *A 25 cm diameter pipe carries oil of sp. gr. 0.9 at a velocity of 3 m/s. At another section the diameter is 20 cm. Find the velocity at this section and also mass rate of flow of oil.*

Solution. Given :

at section 1, $D_1 = 25 \text{ cm} = 0.25 \text{ m}$

$$A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} \times 0.25^2 = 0.049 \text{ m}^3$$

$V_1 = 3$ m/s

at section 2, $D_2 = 20 \text{ cm} = 0.2 \text{ m}$

$$A_2 = \frac{\pi}{4} (0.2)^2 = 0.0314 \text{ m}^2$$

$V_2 = ?$

Mass rate of flow of oil = ?

Applying continuity equation at sections 1 and 2,

$$A_1 V_1 = A_2 V_2$$

or $$0.049 \times 3.0 = 0.0314 \times V_2$$

$$\therefore \quad V_2 = \frac{0.049 \times 3.0}{0.0314} = \textbf{4.68 m/s. Ans.}$$

Mass rate of flow of oil = Mass density $\times Q = \rho \times A_1 \times V_1$

$$\text{Sp. gr. of oil} = \frac{\text{Density of oil}}{\text{Density of water}}$$

$\therefore$ Density of oil = Sp. gr. of oil $\times$ Density of water

$$= 0.9 \times 1000 \text{ kg/m}^3 = \frac{900 \text{ kg}}{\text{m}^3}$$

$\therefore$ Mass rate of flow $= 900 \times 0.049 \times 3.0$ kg/s = **132.23 kg/s. Ans.**

Problem 5.5 *A jet of water from a 25 mm diameter nozzle is directed vertically upwards. Assuming that the jet remains circular and neglecting any loss of energy, that will be the diameter at a point 4.5 m above the nozzle, if the velocity with which the jet leaves the nozzle is 12 m/s.*

Solution. Given :

Dia. of nozzle, $D_1 = 25 \text{ mm} = 0.025 \text{ m}$

Velocity of jet at nozzle, $V_1 = 12$ m/s

Height of point A, $h = 4.5$ m

Let the velocity of the jet at a height 4.5 m = V_2

Consider the vertical motion of the jet from the outlet of the nozzle to the point A (neglecting any loss of energy).

Initial velocity, $u = V_1 = 12$ m/s

Final velocity, $V = V_2$

Value of $g = -9.81 \text{ m/s}^2$ and $h = 4.5$ m

Using, $V^2 - u^2 = 2gh$, we get

$$V_2^2 - 12^2 = 2 \times (-9.81) \times 4.5$$

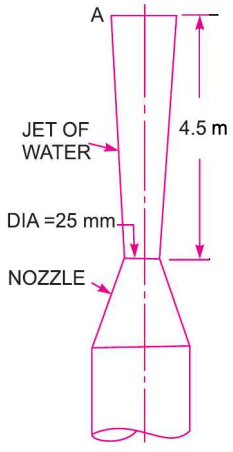


Fig. 5.5

$\therefore \quad V_2 = \sqrt{12^2 - 2 \times 9.81 \times 4.5} = \sqrt{144 - 88.29} = 7.46$ m/s

Now applying continuity equation to the outlet of nozzle and at point A, we get

$$A_1 V_1 = A_2 V_2$$

or $$A_2 = \frac{A_1 V_1}{V_2} = \frac{\frac{\pi}{4} D_1^2 \times V_1}{V_2} = \frac{\pi \times (0.025)^2 \times 12}{4 \times 7.46} = 0.0007896$$

Let D_2 = Diameter of jet at point A.

Then $$A_2 = \frac{\pi}{2} D_2^2 \text{ or } 0.0007896 = \frac{\pi}{4} \times D_2^2$$

$\therefore \quad D_2 = \sqrt{\frac{0.0007896 \times 4}{\pi}} = 0.0317$ m = **31.7 mm. Ans.**

▶ 5.6 CONTINUITY EQUATION IN THREE-DIMENSIONS

Consider a fluid element of lengths dx, dy and dz in the direction of x, y and z. Let u, v and w are the inlet velocity components in x, y and z directions respectively. Mass of fluid entering the face $ABCD$ per second

$$= \rho \times \text{Velocity in } x\text{-direction} \times \text{Area of } ABCD$$
$$= \rho \times u \times (dy \times dz)$$

Then mass of fluid leaving the face $EFGH$ per second $= \rho u\, dydz + \frac{\partial}{\partial x}(\rho u\, dydz)\, dx$

$\therefore$ Gain of mass in x-direction

$$= \text{Mass through } ABCD - \text{Mass through } EFGH \text{ per second}$$
$$= \rho u\, dydz - \rho u\, dydz - \frac{\partial}{\partial x}(\rho u\, dydz) dx$$
$$= -\frac{\partial}{\partial x}(\rho u\, dydz)\, dx$$
$$= -\frac{\partial}{\partial x}(\rho u)\, dx\, dydz \qquad \{\because dydz \text{ is constant}\}$$

Similarly, the net gain of mass in y-direction

$$= -\frac{\partial}{\partial y}(\rho v)\, dxdydz$$

and in z-direction $$= -\frac{\partial}{\partial z}(\rho w)\, dxdydz$$

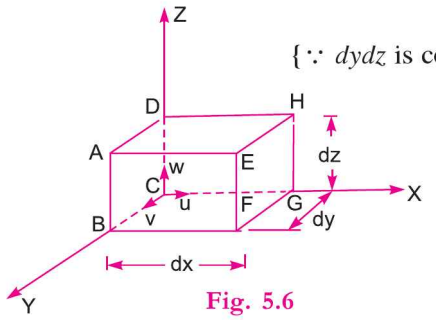


Fig. 5.6

$\therefore \quad$ Net gain of masses $= -\left[\frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) + \frac{\partial}{\partial z}(\rho w)\right] dxdydz$

Since the mass is neither created nor destroyed in the fluid element, the net increase of mass per unit time in the fluid element must be equal to the rate of increase of mass of fluid in the element. But mass

of fluid in the element is ρ. *dx*. *dy*. *dz* and its rate of increase with time is $\frac{\partial}{\partial t}$ (ρ *dx*. *dy*. *dz*) or

$\frac{\partial \rho}{\partial t}$. *dx dy dz*.

Equating the two expressions,

or $$-\left[\frac{\partial}{\partial x}(\rho u)+\frac{\partial}{\partial y}(\rho v)+\frac{\partial}{\partial z}(\rho w)\right] dxdydz = \frac{\partial \rho}{\partial t}.\ dxdydz$$

or $$\frac{\partial \rho}{\partial t}+\frac{\partial}{\partial x}(\rho u)+\frac{\partial}{\partial y}(\rho v)+\frac{\partial}{\partial z}(\rho w) = 0$$ [Cancelling *dx.dy.dz* from both sides] ...(5.3*A*)

Equation (5.3*A*) is the continuity equation in cartesian co-ordinates in its most general form. This equation is applicable to :

(*i*) Steady and unsteady flow,

(*ii*) Uniform and non-uniform flow, and

(*iii*) Compressible and incompressible fluids.

For steady flow, $\frac{\partial \rho}{\partial t} = 0$ and hence equation (5.3*A*) becomes as

$$\frac{\partial}{\partial x}(\rho u)+\frac{\partial}{\partial y}(\rho v)+\frac{\partial}{\partial z}(\rho w) = 0 \qquad ...(5.3B)$$

If the fluid is incompressible, then ρ is constant and the above equation becomes as

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z} = 0 \qquad ...(5.4)$$

Equation (5.4) is the continuity equation in three-dimensions. For a two-dimensional flow, the component $w = 0$ and hence continuity equation becomes as

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y} = 0. \qquad ...(5.5)$$

5.6.1 Continuity Equation in Cylindrical Polar Co-ordinates. The continuity equation in cylindrical polar co-ordinates (*i.e.*, *r*, θ, *z* co-ordinates) is derived by the procedure given below.

Consider a two-dimensional incompressible flow field. The two-dimensional polar co-ordinates are *r* and θ. Consider a fluid element *ABCD* between the radii *r* and *r* + *dr* as shown in Fig. 5.7. The angle subtended by the element at the centre is *d*θ. The components of the velocity *V* are u_r in the radial direction and u_θ in the tangential direction. The sides of the element are having the lengths as

Side *AB* = *r*dθ, *BC* = *dr*, *DC* = (*r* + *dr*) *d*θ, *AD* = *dr*.

The thickness of the element perpendicular to the plane of the paper is assumed to be unity.

Consider the flow in radial direction

Mass of fluid entering the face AB per unit time

= ρ × Velocity in *r*-direction × Area

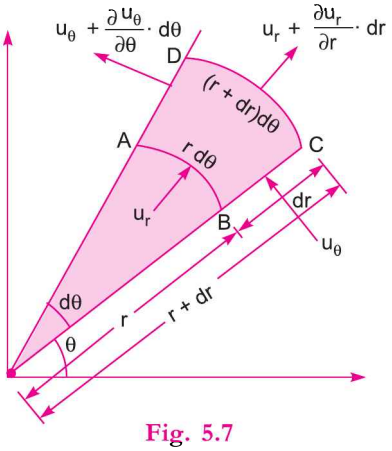


Fig. 5.7

$= \rho \times u_r \times (AB \times 1)$ ($\because$ Area = $AB \times$ Thickness = $rd\theta \times 1$)

$= \rho \times u_r \times (rd\theta \times 1) = \rho \,.\, u_r .\, rd\theta$

Mass of fluid leaving the face CD per unit time

$= \rho \times$ Velocity $\times$ Area

$$= \rho \times \left(u_r + \frac{\partial u_r}{\partial r}.dr \right) \times (CD \times 1)$$ ($\because$ Area = $CD \times 1$)

$$= \rho \times \left(u_r + \frac{\partial u_r}{\partial r}\, dr \right) \times (r + dr)d\theta$$ [$\because$ $CD = (r + dr)\, d\theta$]

$$= \rho \times \left[u_r \times r + u_r dr + r\frac{\partial u_r}{\partial r}\, dr + \frac{\partial u_r}{\partial r}(dr)^2 \right] d\theta$$

$$= \rho \left[u_r \times r + u_r \times dr + r\frac{\partial u_r}{\partial r}.dr \right] d\theta$$

[The term containing $(dr)^2$ is very small and has been neglected]

$\therefore$ Gain of mass in r-direction per unit time

= (Mass through AB – Mass through CD) per unit time

$$= \rho .\, u_r .\, rd\theta - \rho \left[u_r .r + u_r .dr + r\frac{\partial u_r}{\partial r}.dr \right] d\theta$$

$$= \rho .\, u_r .\, rd\theta - \rho .\, u_r .\, r .\, d\theta - \rho \left[u_r .dr + r\frac{\partial u_r}{\partial r}.dr \right] d\theta$$

$$= -\rho \left[u_r .dr + r\frac{\partial u_r}{\partial r}.dr \right] .\, d\theta$$

$$= -\rho \left[\frac{u_r}{r} + \frac{\partial u_r}{\partial r} \right] r .\, dr .\, d\theta$$

[This is written in this form because ($r .\, d\theta .\, dr .\, 1$) is equal to volume of element]

Now consider the flow in θ*-direction*

Gain in mass in θ-direction per unit time

= (Mass through BC – Mass through AD) per unit time

= [$\rho \times$ Velocity through $BC \times$ Area – $\rho \times$ Velocity through $AD \times$ Area]

$$= \left[\rho .u_\theta .dr \times 1 - \rho \left(u_\theta + \frac{\partial u_\theta}{\partial \theta}.d\theta \right) \times dr \times 1 \right]$$

$$= -\rho \left(\frac{\partial u_\theta}{\partial \theta}.d\theta \right) dr \times 1$$ ($\because$ Area = $dr \times 1$)

$$= -\rho \frac{\partial u_\theta}{\partial \theta} . \frac{r .\, d\theta .\, dr}{r}$$ [Multiplying and dividing by r]

$\therefore$ Total gain in fluid mass per unit time

$$= -\rho \left[\frac{u_r}{r} + \frac{\partial u_r}{\partial r} \right] .\, r .\, dr .\, d\theta - \rho \frac{\partial u_\theta}{\partial \theta} . \frac{rd\theta .\, dr}{r} \qquad \text{...(5.5A)}$$

But mass of fluid element $= \rho \times$ Volume of fluid element
$= \rho \times [rd\theta \times dr \times 1]$
$= \rho \times rd\theta \,.\, dr$

Rate of increase of fluid mass in the element with time

$$= \frac{\partial}{\partial t}[\rho \,.\, rd\theta \,.\, dr] = \frac{\partial \rho}{\partial t} . rd\theta\, dr \qquad ...(5.5B)$$

($\because$ $rd\theta \,.\, dr \,.\, 1$ is the volume of element and is a constant quantity)

Since the mass is neither created nor destroyed in the fluid element, hence net gain of mass per unit time in the fluid element must be equal to the rate of increase of mass of fluid in the element.

Hence equating the two expressions given by equations (5.5 *A*) and (5.5 *B*), we get

$$-\rho\left[\frac{u_r}{r} + \frac{\partial u_r}{\partial r}\right] r \,.\, dr.\, d\theta - \rho \frac{\partial u_\theta}{\partial \theta} \frac{rd\theta \,.\, dr}{r} = \frac{\partial \rho}{\partial t}\, rd\theta\, dr$$

or $$-\rho\left[\frac{u_r}{r} + \frac{\partial u_r}{\partial r}\right] - \rho \frac{\partial u_\theta}{\partial \theta} \cdot \frac{1}{r} = \frac{\partial \rho}{\partial t}$$ [Cancelling $rdr \,.\, d\theta$ from both sides]

or $$\frac{\partial \rho}{\partial t} + \rho\left[\frac{u_r}{r} + \frac{\partial u_r}{\partial r}\right] + \rho \frac{\partial u_\theta}{\partial \theta} \cdot \frac{1}{r} = 0 \qquad ...(5.5C)$$

Equation (5.5 *C*) is the continuity equation in polar co-ordinates for two-dimensional flow.

For steady flow $\frac{\partial \rho}{\partial t} = 0$ and hence equation (5.5 *C*) reduces to

$$\rho\left[\frac{u_r}{r} + \frac{\partial u_r}{\partial r}\right] + \rho \frac{\partial u_\theta}{\partial \theta} \cdot \frac{1}{r} = 0$$

or $$\frac{u_r}{r} + \frac{\partial u_r}{\partial r} + \frac{\partial u_\theta}{\partial \theta} \cdot \frac{1}{r} = 0$$

or $$u_r + r\frac{\partial u_r}{\partial r} + \frac{\partial u_\theta}{\partial \theta} = 0$$

or $$\frac{\partial}{\partial r}(ru_r) + \frac{\partial}{\partial \theta}(u_\theta) = 0 \quad \left[\because \frac{\partial}{\partial r}(r \,.\, u_r) = r \,.\, \frac{\partial u_r}{\partial r} + u_r\right] \qquad ...(5.5D)$$

Equation (5.5 *D*) represents the continuity equation in polar co-ordinates for two-dimensional steady incompressible flow.

Problem 5.5A *Examine whether the following velocity components represent a physically possible flow ?*

$$u_r = r \sin \theta,\ u_\theta = 2r \cos \theta.$$

Solution. Given : $u_r = r \sin \theta$ and $u_\theta = 2r \cos \theta$

For physically possible flow, the continuity equation,

$$\frac{\partial}{\partial r}(ru_r) + \frac{\partial}{\partial \theta}(u_\theta) = 0 \text{ should be satisfied.}$$

Now $u_r = r \sin \theta$

Multiplying the above equation by r, we get

$$ru_r = r^2 \sin \theta$$

Differentiating the preceding equation w.r.t. r, we get

$$\frac{\partial}{\partial r}(ru_r) = \frac{\partial}{\partial r}(r^2 \sin\theta)$$

$$= 2r \sin\theta \qquad (\because \sin\theta \text{ is constant w.r.t. } r)$$

Now $u_\theta = 2r \cos\theta$

Differentiating the above equation w.r.t. θ, we get

$$\frac{\partial}{\partial \theta}(u_\theta) = \frac{\partial}{\partial \theta}(2r \cos\theta)$$

$$= 2r(-\sin\theta) \qquad (\because 2r \text{ is constant w.r.t. } \theta)$$

$$= -2r \sin\theta$$

$$\therefore \quad \frac{\partial}{\partial r}(ru_r) + \frac{\partial}{\partial \theta}(u_\theta) = 2r \sin\theta - 2r \sin\theta = 0$$

Hence the continuity equation is satisfied. Hence the given velocity components represent a physically possible flow.

▶ 5.7 VELOCITY AND ACCELERATION

Let V is the resultant velocity at any point in a fluid flow. Let u, v and w are its component in x, y and z directions. The velocity components are functions of space-co-ordinates and time. Mathematically, the velocity components are given as

$$u = f_1(x, y, z, t)$$
$$v = f_2(x, y, z, t)$$
$$w = f_3(x, y, z, t)$$

and Resultant velocity, $V = ui + vj + wk = \sqrt{u^2 + v^2 + w^2}$

Let a_x, a_y and a_z are the **total acceleration** in x, y and z directions respectively. Then by the chain rule of differentiation, we have

$$a_x = \frac{du}{dt} = \frac{\partial u}{\partial x}\frac{dx}{dt} + \frac{\partial u}{\partial y}\frac{dy}{dt} + \frac{\partial u}{\partial z}\cdot\frac{dz}{dt} + \frac{\partial u}{\partial t}$$

But $\frac{dx}{dt} = u, \frac{dy}{dt} = v$ and $\frac{dz}{dt} = w$

$\therefore$

$$a_x = \frac{du}{dt} = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} + \frac{\partial u}{\partial t}$$

Similarly,

$$a_y = \frac{dv}{dt} = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + \frac{\partial v}{\partial t} \qquad \text{...(5.6)}$$

$$a_z = \frac{dw}{dt} = u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} + \frac{\partial w}{\partial t}$$

For steady flow, $\frac{\partial V}{\partial t} = 0$, where V is resultant velocity

or $$\frac{\partial u}{\partial t} = 0, \frac{\partial v}{\partial t} = 0 \text{ and } \frac{\partial w}{\partial t} = 0$$

Hence acceleration in x, y and z directions becomes

$$\left.\begin{aligned} a_x &= \frac{du}{dt} = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \\ a_y &= \frac{dv}{dt} = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} \\ a_z &= \frac{dw}{dt} = u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} \end{aligned}\right\} \quad ...(5.7)$$

Acceleration vector $$\left.\begin{aligned} A &= a_x i + a_y j + a_z k \\ &= \sqrt{a_x^2 + a_y^2 + a_z^2}\,. \end{aligned}\right\} \quad ...(5.8)$$

5.7.1 Local Acceleration and Convective Acceleration. **Local acceleration** is defined as the rate of increase of velocity with respect to time at a given point in a flow field. In the equation given by (5.6), the expression $\frac{\partial u}{\partial t}, \frac{\partial v}{\partial t}$ or $\frac{\partial w}{\partial t}$ is known as local acceleration.

Convective acceleration is defined as the rate of change of velocity due to the change of position of fluid particles in a fluid flow. The expressions other than $\frac{\partial u}{\partial t}, \frac{\partial v}{\partial t}$ and $\frac{\partial w}{\partial t}$ in equation (5.6) are known as convective acceleration.

Problem 5.6 *The velocity vector in a fluid flow is given*

$$V = 4x^3 i - 10x^2 yj + 2tk.$$

Find the velocity and acceleration of a fluid particle at (2, 1, 3) at time t = 1.

Solution. The velocity components u, v and w are $u = 4x^3$, $v = -10x^2 y$, $w = 2t$

For the point (2, 1, 3), we have $x = 2$, $y = 1$ and $z = 3$ at time $t = 1$.

Hence velocity components at (2, 1, 3) are

$$u = 4 \times (2)^3 = 32 \text{ units}$$
$$v = -10(2)^2(1) = -40 \text{ units}$$
$$w = 2 \times 1 = 2 \text{ units}$$

$\therefore$ Velocity vector V at (2, 1, 3) $= 32i - 40j + 2k$

or Resultant velocity $= \sqrt{u^2 + v^2 + w^2}$

$$= \sqrt{32^2 + (-40)^2 + 2^2} = \sqrt{1024 + 1600 + 4} = \mathbf{51.26 \text{ units. Ans.}}$$

Acceleration is given by equation (5.6)

$$a_x = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} + \frac{\partial u}{\partial t}$$

$$a_y = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + \frac{\partial v}{\partial t}$$

$$a_z = u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} + \frac{\partial w}{\partial t}$$

Now from velocity components, we have

$$\frac{\partial u}{\partial x} = 12x^2,\ \frac{\partial u}{\partial y} = 0,\ \frac{\partial u}{\partial z} = 0 \text{ and } \frac{\partial u}{\partial t} = 0$$

$$\frac{\partial v}{\partial x} = -20xy,\ \frac{\partial v}{\partial y} = -10x^2,\ \frac{\partial v}{\partial z} = 0\ \frac{\partial v}{\partial t} = 0$$

$$\frac{\partial w}{\partial x} = 0,\ \frac{\partial w}{\partial y} = 0,\ \frac{\partial w}{\partial z} = 0 \text{ and } \frac{\partial w}{\partial t} = 2.1$$

Substituting the values, the acceleration components at (2, 1, 3) at time $t = 1$ are

$$a_x = 4x^3(12x^2) + (-10x^2y)(0) + 2t \times (0) + 0$$
$$= 48x^5 = 48 \times (2)^5 = 48 \times 32 = 1536 \text{ units}$$
$$a_y = 4x^3(-20xy) + (-10x^2y)(-10x^2) + 2t(0) + 0$$
$$= -80x^4y + 100x^4y$$
$$= -80(2)^4(1) + 100(2)^4 \times 1 = -1280 + 1600 = 320 \text{ units.}$$
$$a_z = 4x^3(0) + (-10x^2y)(0) + (2t)(0) + 2.1 = 2.0 \text{ units}$$

$\therefore$ Acceleration is $A = a_x i + a_y j + a_z k =$ **1536i + 320j + 2k. Ans.**

or Resultant $A = \sqrt{(1536)^2 + (320)^2 + (2)^2}$ units

$= \sqrt{2359296 + 102400 + 4}$ = **1568.9 units. Ans.**

Problem 5.7 *The following cases represent the two velocity components, determine the third component of velocity such that they satisfy the continuity equation :*

(i) $u = x^2 + y^2 + z^2$; $v = xy^2 - yz^2 + xy$

(ii) $v = 2y^2$, $w = 2xyz$.

Solution. The continuity equation for incompressible fluid is given by equation (5.4) as

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

Case I. $u = x^2 + y^2 + z^2 \quad \therefore \quad \frac{\partial u}{\partial x} = 2x$

$v = xy^2 - yz^2 + xy \quad \therefore \quad \frac{\partial v}{\partial y} = 2xy - z^2 + x$

Substituting the values of $\frac{\partial u}{\partial x}$ and $\frac{\partial v}{\partial y}$ in continuity equation.

$$2x + 2xy - z^2 + x + \frac{\partial w}{\partial z} = 0$$

or $$\frac{\partial w}{\partial z} = -3x - 2xy + z^2 \text{ or } \partial w = (-3x - 2xy + z^2)\,\partial z$$

Integration of both sides gives $\int dw = \int\left(-3x - 2xy + z^2\right) dz$

or $$w = \left(-3xz - 2xyz + \frac{z^3}{3}\right) + \text{Constant of integration,}$$

where constant of integration cannot be a function of z. But it can be a function of x and y that is $f(x, y)$.

$\therefore$ $$w = \left(\mathbf{-3xz - 2xyz + \frac{z^3}{3}}\right) \mathbf{+ f(x, y).\ Ans.}$$

Case II. $v = 2y^2 \quad \therefore \quad \dfrac{\partial v}{\partial y} = 4y$

$$w = 2xyz \quad \therefore \quad \frac{\partial w}{\partial z} = 2xy$$

Substituting the values of $\dfrac{\partial v}{\partial y}$ and $\dfrac{\partial w}{\partial z}$ in continuity equation, we get

$$\frac{\partial u}{\partial x} + 4y + 2xy = 0$$

or $$\frac{\partial u}{\partial x} = -4y - 2xy \text{ or } du = (-4y - 2xy)\, dx$$

Integrating, we get $$u = -4xy - 2y\frac{x^2}{2} + f(y, z) = \mathbf{-4xy - x^2y + f(y, z).\ Ans.}$$

Problem 5.8 *A fluid flow field is given by*

$$V = x^2yi + y^2zj - (2xyz + yz^2)k$$

Prove that it is a case of possible steady incompressible fluid flow. Calculate the velocity and acceleration at the point (2, 1, 3).

Solution. For the given fluid flow field $u = x^2y \quad \therefore \quad \dfrac{\partial u}{\partial x} = 2xy$

$$v = y^2z \quad \therefore \quad \frac{\partial v}{\partial y} = 2yz$$

$$w = -2xyz - yz^2 \quad \therefore \quad \frac{\partial w}{\partial z} = -2xy - 2yz.$$

For a case of possible steady incompressible fluid flow, the continuity equation (5.4) should be satisfied.

i.e., $$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0.$$

Substituting the values of $\dfrac{\partial u}{\partial x}$, $\dfrac{\partial v}{\partial y}$ and $\dfrac{\partial w}{\partial z}$, we get

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 2xy + 2yz - 2xy - 2yz = 0$$

Hence the velocity field $V = x^2yi + y^2zj - (2xyz + yz^2)\,k$ is a possible case of fluid flow. **Ans.**

Velocity at (2, 1, 3)

Substituting the values $x = 2$, $y = 1$ and $z = 3$ in velocity field, we get

$$V = x^2yi + y^2zj - (2xyz + yz^2)\,k$$
$$= 2^2 \times 1i + 1^2 \times 3j - (2 \times 2 \times 1 \times 3 + 1 \times 3^2)\,k$$
$$= \mathbf{4i + 3j - 21k.\ Ans.}$$

and Resultant velocity $= \sqrt{4^2 + 3^2 + (-21)^2} = \sqrt{16 + 9 + 441} = \sqrt{466} = \mathbf{21.587\ units.\ Ans.}$

Acceleration at (2, 1, 3)

The acceleration components a_x, a_y and a_z for steady flow are

$$a_x = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}$$

$$a_y = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z}$$

$$a_z = u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z}$$

$$u = x^2y, \frac{\partial u}{\partial x} = 2xy, \frac{\partial u}{\partial y} = x^2 \text{ and } \frac{\partial u}{\partial z} = 0$$

$$v = y^2z, \frac{\partial v}{\partial x} = 0, \frac{\partial v}{\partial y} = 2yz, \frac{\partial v}{\partial z} = y^2$$

$$w = -2xyz - yz^2, \frac{\partial w}{\partial x} = -2yz, \frac{\partial w}{\partial y} = -2xz - z^2, \frac{\partial w}{\partial z} = -2xy - 2yz.$$

Substituting these values in acceleration components, we get acceleration at (2, 1, 3)

$$a_x = x^2y\,(2xy) + y^2z\,(x)^2 - (2xyz + yz^2)\,(0)$$
$$= 2x^3y^2 + x^2y^2z$$
$$= 2\,(2)^3 1^2 + 2^2 \times 1^2 \times 3 = 2 \times 8 + 12$$
$$= 16 + 12 = 28 \text{ units}$$
$$a_y = x^2y\,(0) + y^2z\,(2yz) - (2xyz + yz^2)\,(y^2)$$
$$= 2y^3z^2 - 2xy^3z - y^3z^2$$
$$= 2 \times 1^3 \times 3^2 - 2 \times 2 \times 1^3 \times 3 - 1^3 \times 3^2 = 18 - 12 - 9 = -3 \text{ units}$$
$$a_z = x^2y\,(-2yz) + y^2z\,(-2xz - z^2) - (2xyz + yz^2)\,(-2xy - 2yz)$$
$$= -2x^2y^2z - 2xy^2z^2 - y^2z^3 + [4x^2y^2z + 2xy^2z^2 + 4xy^2z^2 + 2y^2z^3]$$
$$= -2 \times 2^2 \times 1^2 \times 3 - 2 \times 2 \times 1^2 \times 3^2 - 1^2 \times 3^3$$
$$+ [4 \times 2^2 \times 1^2 \times 3 + 2 \times 2 \times 1^2 \times 3^2 + 4 \times 2 \times 1^2 \times 3^2 + 2 \times 1^2 \times 3^3]$$
$$= -24 - 36 - 27 + [48 + 36 + 72 + 54]$$
$$= -24 - 36 - 27 + 48 + 36 + 72 + 54 = 123$$

$\therefore$ Acceleration $= a_xi + a_yj + a_zk = \mathbf{28i - 3j + 123k.\ Ans.}$

or Resultant acceleration $= \sqrt{28^2 + (-3)^2 + 123^2} = \sqrt{784 + 9 + 15129}$

$= \sqrt{15922} =$ **126.18 units. Ans.**

Problem 5.9 *Find the convective acceleration at the middle of a pipe which converges uniformly from 0.4 m diameter to 0.2 m diameter over 2 m length. The rate of flow is 20 lit/s. If the rate of flow changes uniformly from 20 l/s to 40 l/s in 30 seconds, find the total acceleration at the middle of the pipe at 15th second.*

Solution. Given :

Diameter at section 1, $D_1 = 0.4$ m ; $D_2 = 0.2$ m, $L = 2$ m, $Q = 20$ *l*/s $= 0.02$ m^3/s as one litre $= 0.001$m$^3 = 1000$ cm^3

Find (*i*) Convective acceleration at middle *i.e.,* at A when $Q = 20$ *l*/s.

(*ii*) Total acceleration at A when Q changes from 20 *l*/s to 40 *l*/s in 30 seconds.

Case I. In this case, the rate of flow is constant and equal to 0.02 m^3/s. The velocity of flow is in x-direction only. Hence this is one-dimensional flow and velocity components in y and z directions are zero or $v = 0$, $z = 0$.

$$\therefore \quad \text{Convective acceleration} = u\frac{\partial u}{\partial y} \text{ only} \qquad \ldots(i)$$

Let us find the value of u and $\frac{\partial u}{\partial x}$ at a distance x from inlet

The diameter (D_x) at a distance x from inlet or at section X-X is given by,

$$D_x = 0.4 - \frac{0.4 - 0.2}{2} \times x$$

$$= (0.4 - 0.1\,x) \text{ m}$$

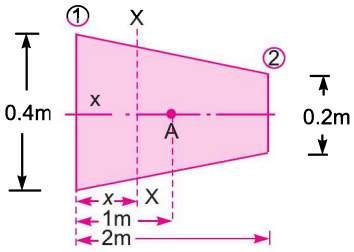


Fig. 5.8

The area of cross-section (A_x) at section X-X is given by,

$$A_x = \frac{\pi}{4} D_x^2 = \frac{\pi}{4} (0.4 - 0.1\,x)^2$$

Velocity (u) at the section X-X in terms of Q (*i.e.*, in terms of rate of flow)

$$u = \frac{Q}{\text{Area}} = \frac{Q}{A_x} = \frac{Q}{\frac{\pi}{4} D_x^2} = \frac{4Q}{\pi\,(0.4 - 0.1x)^2}$$

$$= \frac{1.273Q}{(0.4 - 0.1x)^2} = 1.273\,Q\,(0.4 - 0.1\,x)^{-2} \text{ m/s} \qquad \ldots(ii)$$

To find $\frac{\partial u}{\partial x}$, we must differentiate equation (*ii*) with respect to x.

$$\frac{\partial u}{\partial x} = \frac{\partial}{\partial x} [1.273\,Q\,(0.4 - 0.1\,x)^{-2}]$$

$$= 1.273\,Q\,(-2)\,(0.4 - 0.1\,x)^{-1} \times (-0.1) \qquad \text{[Here Q is constant]}$$

$$= 0.2546\,Q\,(0.4 - 0.1\,x)^{-1} \qquad \ldots(iii)$$

Substituting the value of u and $\frac{\partial u}{\partial x}$ in equation (*i*), we get

Convective acceleration $= [1.273\,Q\,(0.4 - 0.1\,x)^{-2}] \times [0.2546\,Q\,(0.4 - 0.1\,x)^{-1}]$

$$= 1.273 \times 0.2546 \times Q^2 \times (0.4 - 0.1\,x)^{-3}$$

$$= 1.273 \times 0.2546 \times (0.02)^2 \times (0.4 - 0.1\,x)^{-3} \quad [\because Q = 0.02\ m^3/s]$$

$\therefore$ Convective acceleration at the middle (where $x = 1$ m)

$$= 1.273 \times 0.2546 \times (0.02)^2 \times (0.4 - 0.1 \times 1)^{-3}\ m/s^2$$
$$= 1.273 \times 0.2546 \times (0.02)^2 \times (0.3)^{-3}\ m/s^2$$
$$= \mathbf{0.0048\ m/s^2.\ Ans.}$$

Case II. When Q changes from 0.02 m^3/s to 0.04 m^3/s in 30 seconds, find the total acceleration at $x = 1$ m and $t = 15$ seconds.

Total acceleration = Convective acceleration + Local acceleration at $t = 15$ seconds.

The rate of flow at $t = 15$ seconds is given by

$$Q = Q_1 + \frac{Q_2 - Q_1}{30} \times 15 \text{ where } Q_2 = 0.04\ m^3/s \text{ and } Q_1 = 0.02\ m^3/s$$

$$= 0.02 + \frac{(0.04 - 0.02)}{30} \times 15 = 0.03\ m^3/s$$

The velocity (u) and gradient $\left(\frac{\partial u}{\partial x}\right)$ in terms of Q are given by equations (*ii*) and (*iii*) respectively

$\therefore$ Convective acceleration $= u.\ \frac{\partial u}{\partial x}$

$$= [1.273\ Q\ (0.4 - 0.1\,x)^{-2}] \times [0.2546\ Q\ (0.4 - 0.1\,x)^{-1}]$$
$$= 1.273 \times 0.2546\ Q^2 \times (0.4 - 0.1 \times 1)^{-3}$$

$\therefore$ Convective acceleration (when $Q = 0.03\ m^3/s$ and $x = 1$ m)

$$= 1.273 \times 0.2546 \times (0.03)^2 \times (0.4 - 0.1 \times 1)^{-3}$$
$$= 1.273 \times 0.2546 \times (0.03)^2 \times (0.3)^{-3}\ m/s^2$$
$$= 0.0108\ m/s^2 \qquad \ldots(iv)$$

$$\text{Local acceleration} = \frac{\partial u}{\partial t} = \frac{\partial}{\partial t}\,[1.273\ Q\ (0.4 - 0.1\,x)^{-2}]$$

$[\because u$ from equation (*ii*) is $u = 1.273\ Q\ (0.4 - 0.1\,x)^{-2}]$

$$= 1.273 \times (0.4 - 0.1\,x)^{-2} \times \frac{\partial Q}{\partial t}$$

[$\because$ Local acceleration is at a point where x is constant but Q is changing]

Local acceleration (at $x = 1$ m)

$$= 1.273 \times (0.4 - 0.1 \times 1)^{-2} \times \frac{\partial Q}{\partial t}$$

$$= 1.273 \times (0.3)^{-2} \times \frac{0.02}{30} \qquad \left[\because \frac{\partial Q}{\partial t} = \frac{Q_2 - Q_1}{t} = \frac{0.04 - 0.02}{30} = \frac{0.02}{30}\right]$$

$$= 0.00943\ m/s^2 \qquad \ldots(v)$$

Hence adding equations (*iv*) and (*v*), we get total acceleration.

$\therefore$ Total acceleration = Convective acceleration + Local acceleration

$$= 0.0108 + 0.00943 = \mathbf{0.02023\ m/s^2.\ Ans.}$$

▶ 5.8 VELOCITY POTENTIAL FUNCTION AND STREAM FUNCTION

5.8.1 Velocity Potential Function. It is defined as a scalar function of space and time such that its negative derivative with respect to any direction gives the fluid velocity in that direction. It is defined by ϕ (Phi). Mathematically, the velocity, potential is defined as $\phi = f(x, y, z)$ for steady flow such that

$$\left.\begin{aligned} u &= -\frac{\partial \phi}{\partial x} \\ v &= -\frac{\partial \phi}{\partial y} \\ w &= -\frac{\partial \phi}{\partial z} \end{aligned}\right\} \quad ...(5.9)$$

where u, v and w are the components of velocity in x, y and z directions respectively.

The velocity components in cylindrical polar co-ordinates in terms of velocity potential function are given by

$$\left.\begin{aligned} u_r &= \frac{\partial \phi}{\partial r} \\ u_\theta &= \frac{1}{r}\frac{\partial \phi}{\partial \theta} \end{aligned}\right\} \quad ...(5.9A)$$

where u_r = velocity component in radial direction (*i.e.*, in r direction)
and u_θ = velocity component in tangential direction (*i.e.*, in θ direction)

The continuity equation for an incompressible steady flow is $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$.

Substituting the values of u, v and w from equation (5.9), we get

$$\frac{\partial}{\partial x}\left(-\frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(-\frac{\partial \phi}{\partial y}\right) + \frac{\partial}{\partial z}\left(-\frac{\partial \phi}{\partial z}\right) = 0$$

or

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} = 0. \quad ...(5.10)$$

Equation (5.10) is a Laplace equation.

For two-dimension case, equation (5.10) reduces to $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$. ...(5.11)

If any value of ϕ that satisfies the Laplace equation, will correspond to some case of fluid flow.

Properties of the Potential Function. The rotational components* are given by

$$\omega_z = \frac{1}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)$$

* Please, refer to equation (5.17) on page 192.

$$\omega_y = \frac{1}{2}\left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)$$

$$\omega_x = \frac{1}{2}\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)$$

Substituting the values, of u, v and w from equation (5.9) in the above rotational components, we get

$$\omega_z = \frac{1}{2}\left[\frac{\partial}{\partial x}\left(-\frac{\partial \phi}{\partial y}\right) - \frac{\partial}{\partial y}\left(-\frac{\partial \phi}{\partial x}\right)\right] = \frac{1}{2}\left[-\frac{\partial^2 \phi}{\partial x \partial y} + \frac{\partial^2 \phi}{\partial y \partial x}\right]$$

$$\omega_y = \frac{1}{2}\left[\frac{\partial}{\partial z}\left(-\frac{\partial \phi}{\partial x}\right) - \frac{\partial}{\partial x}\left(-\frac{\partial \phi}{\partial z}\right)\right] = \frac{1}{2}\left[-\frac{\partial^2 \phi}{\partial z \partial x} + \frac{\partial^2 \phi}{\partial x \partial z}\right]$$

and

$$\omega_x = \frac{1}{2}\left[\frac{\partial}{\partial y}\left(-\frac{\partial \phi}{\partial z}\right) - \frac{\partial}{\partial z}\left(-\frac{\partial \phi}{\partial y}\right)\right] = \frac{1}{2}\left[-\frac{\partial^2 \phi}{\partial y \partial z} + \frac{\partial^2 \phi}{\partial z \partial y}\right]$$

If ϕ is a continuous function, then $\frac{\partial^2 \phi}{\partial x \partial y} = \frac{\partial^2 \phi}{\partial y \partial x}$; $\frac{\partial^2 \phi}{\partial z \partial x} = \frac{\partial^2 \phi}{\partial x \partial z}$; etc.

$\therefore \quad \omega_z = \omega_y = \omega_x = 0.$

When rotational components are zero, the flow is called irrotational. Hence the properties of the potential function are :

1. If velocity potential (ϕ) exists, the flow should be irrotational.

2. If velocity potential (ϕ) satisfies the Laplace equation, it represents the possible steady incompressible irrotational flow.

5.8.2 Stream Function. It is defined as the scalar function of space and time, such that its partial derivative with respect to any direction gives the velocity component at right angles to that direction. It is denoted by ψ (*Psi*) and defined only for two-dimensional flow. Mathematically, for steady flow it is defined as $\psi = f(x, y)$ such that

$$\left.\begin{aligned} \frac{\partial \psi}{\partial x} &= v \\ \text{and} \quad \frac{\partial \psi}{\partial y} &= -u \end{aligned}\right\} \qquad \ldots(5.12)$$

The velocity components in cylindrical polar co-ordinates in terms of stream function are given as

$$u_r = \frac{1}{r}\frac{\partial \psi}{\partial \theta} \text{ and } u_\theta = -\frac{\partial \psi}{\partial r} \qquad \ldots(5.12A)$$

where u_r= radial velocity and u_θ = tangential velocity

The continuity equation for two-dimensional flow is $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$

Substituting the values of u and v from equation (5.12), we get

$$\frac{\partial}{\partial x}\left(-\frac{\partial \psi}{\partial y}\right)+\frac{\partial}{\partial y}\left(\frac{\partial \psi}{\partial x}\right)=0 \text{ or } -\frac{\partial^2 \psi}{\partial x \partial y}+\frac{\partial^2 \psi}{\partial x \partial y}=0.$$

Hence existence of ψ means a possible case of fluid flow. The flow may be rotational or irrotational.

The rotational component ω_z is given by $\omega_z = \frac{1}{2}\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)$.

Substituting the values of u and v from equation (5.12) in the above rotational component, we get

$$\omega_z = \frac{1}{2}\left[\frac{\partial}{\partial x}\left(\frac{\partial \psi}{\partial x}\right)-\frac{\partial}{\partial y}\left(-\frac{\partial \psi}{\partial y}\right)\right]=\frac{1}{2}\left[\frac{\partial^2 \psi}{\partial x^2}+\frac{\partial^2 \psi}{\partial y^2}\right]$$

For irrotational flow, $\omega_z = 0$. Hence above equation becomes as $\frac{\partial^2 \psi}{\partial x^2}+\frac{\partial \psi}{\partial y^2}=0$

which is Laplace equation for ψ.

The **properties** of stream function (ψ) are :

1. If stream function (ψ) exists, it is a possible case of fluid flow which may be rotational or irrotational.
2. If stream function (ψ) satisfies the Laplace equation, it is a possible case of an irrotational flow.

5.8.3 Equipotential Line. A line along which the velocity potential ϕ is constant, is called equipotential line.

For equipotential line ϕ = Constant

$\therefore$ $d\psi = 0$

But $\phi = f(x, y)$ for steady flow

$\therefore$ $$d\phi = \frac{\partial \phi}{\partial x}dx+\frac{\partial \phi}{\partial y}dy$$

$$= -udx - vdy \qquad \left\{\because \frac{\partial \phi}{\partial x}=-u, \frac{\partial \phi}{\partial x}=-v\right\}$$

$$= -(udx + vdy).$$

For equipotential line, $d\phi = 0$

or $-(udx + vdy) = 0$ or $udx + vdy = 0$

$\therefore$ $$\frac{dy}{dx} = -\frac{u}{v} \qquad ...(5.13)$$

But $\frac{dy}{dx}$ = Slope of equipotential line.

5.8.4 Line of Constant Stream Function

ψ = Constant

$\therefore$ $d\psi = 0$

But $$d\psi = \frac{\partial \psi}{\partial x}dx+\frac{\partial \psi}{\partial y}dy = +vdx - udy \qquad \left\{\because \frac{\partial \psi}{\partial x}=v; \frac{\partial \psi}{\partial y}=-u\right\}$$

For a line of constant stream function

$$= d\psi = 0 \text{ or } vdx - udy = 0$$

or
$$\frac{dy}{dx} = \frac{v}{u} \qquad ...(5.14)$$

But $\frac{dy}{dx}$ is slope of stream line.

From equations (5.13) and (5.14) it is clear that the product of the slope of the equipotential line and the slope of the stream line at the point of intersection is equal to – 1. Thus the equipotential lines are othogonal to the stream lines at all points of intersection.

5.8.5 Flow Net. A grid obtained by drawing a series of equipotential lines and stream lines is called a flow net. The flow net is an important tool in analysing two-dimensional irrotational flow problems.

5.8.6 Relation between Stream Function and Velocity Potential Function

From equation (5.9),

we have
$$u = -\frac{\partial \phi}{\partial x} \text{ and } v = -\frac{\partial \phi}{\partial y}$$

From equation (5.12), we have $u = -\frac{\partial \psi}{\partial y}$ and $v = \frac{\partial \psi}{\partial x}$

Thus, we have
$$u = -\frac{\partial \phi}{\partial x} = -\frac{\partial \psi}{\partial y} \text{ and } v = -\frac{\partial \phi}{\partial y} = \frac{\partial \psi}{\partial x}$$

Hence
$$\frac{\partial \phi}{\partial x} = \frac{\partial \psi}{\partial y}$$

and
$$\frac{\partial \phi}{\partial y} = -\frac{\partial \psi}{\partial x} \qquad ...(5.15)$$

Problem 5.10 *The velocity potential function (ϕ) is given by an expression*

$$\phi = -\frac{xy^3}{3} - x^2 + \frac{x^3 y}{3} + y^2$$

(*i*) *Find the velocity components in x and y direction.*
(*ii*) *Show that ϕ represents a possible case of flow.*

Solution. Given :
$$\phi = -\frac{xy^3}{3} - x^2 + \frac{x^3 y}{3} + y^2$$

The partial derivatives of ϕ w.r.t. x and y are

$$\frac{\partial \phi}{\partial x} = -\frac{y^3}{3} - 2x + \frac{3x^2 y}{3} \qquad ...(1)$$

and
$$\frac{\partial \phi}{\partial y} = -\frac{3xy^2}{3} + \frac{x^3}{3} + 2y \qquad ...(2)$$

(*i*) The velocity components u and v are given by equation (5.9)

$$u = -\frac{\partial \phi}{\partial x} = -\left[-\frac{y^3}{3} - 2x + \frac{3x^2 y}{3}\right] = \frac{y^3}{3} + 2x - x^2 y$$

$\therefore$ $$u = \mathbf{\frac{y^2}{3} + 2x - x^2 y.\ Ans.}$$

$\therefore$ $$v = -\frac{\partial \phi}{\partial y} = -\left[-\frac{3xy^2}{3} + \frac{x^3}{3} + 2y\right] = \frac{3xy^2}{3} - \frac{x^3}{3} - 2y = \mathbf{xy^2 - \frac{x^3}{3} - 2y.}$$

Ans.

(*ii*) The given value of ϕ, will represent a possible case of flow if it satisfies the Laplace equation, *i.e.*,

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

From equations (1) and (2), we have

Now $$\frac{\partial \phi}{\partial x} = -y^3/3 - 2x + x^2 y$$

$\therefore$ $$\frac{\partial^2 \phi}{\partial x^2} = -2 + 2xy$$

and $$\frac{\partial \phi}{\partial y} = -xy^2 + \frac{x^3}{3} + 2y$$

$\therefore$ $$\frac{\partial^2 \phi}{\partial y^2} = -2xy + 2$$

$\therefore$ $$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = (-2 + 2xy) + (-2xy + 2) = 0$$

$\therefore$ Laplace equation is satisfied and hence ϕ represent a possible case of flow. **Ans.**

Problem 5.11 *The velocity potential function is given by $\phi = 5\,(x^2 - y^2)$. Calculate the velocity components at the point (4, 5).*

Solution. $$\phi = 5\,(x^2 - y^2)$$

$\therefore$ $$\frac{\partial \phi}{\partial x} = 10x$$

$$\frac{\partial \phi}{\partial y} = -10y.$$

But velocity components u and v are given by equation (5.9) as

$$u = -\frac{\partial \phi}{\partial x} = -10x$$

$$v = -\frac{\partial \phi}{\partial y} = -(-10y) = 10y$$

The velocity components at the point (4, 5), *i.e.*, at $x = 4$, $y = 5$

$u = -10 \times 4 =$ **$-$ 40 units. Ans.**

$v = 10 \times 5 =$ **50 units. Ans.**

Problem 5.12 *A stream function is given by* $\psi = 5x - 6y$.
Calculate the velocity components and also magnitude and direction of the resultant velocity at any point.

Solution. $\psi = 5x - 6y$

$\therefore \quad \frac{\partial \psi}{\partial x} = 5$ and $\frac{\partial \psi}{\partial y} = -6.$

But the velocity components u and v in terms of stream function are given by equation (5.12) as

$$u = -\frac{\partial \psi}{\partial y} = -(-6) = \mathbf{6\ units/sec.\ Ans.}$$

$$v = \frac{\partial \psi}{\partial x} = \mathbf{5\ units/sec.\ Ans.}$$

Resultant velocity $= \sqrt{u^2 + v^2} = \sqrt{6^2 + 5^2} = \sqrt{36 + 25} = \sqrt{61} = 7.81$ unit/sec

Direction is given by, $\tan \theta = \frac{v}{u} = \frac{5}{6} = 0.833$

$\therefore \quad \theta = \tan^{-1} .833 = \mathbf{39° 48'.\ Ans.}$

Problem 5.13 *If for a two-dimensional potential flow, the velocity potential is given by*

$$\phi = x\,(2y - 1)$$

determine the velocity at the point P (4, 5). Determine also the value of stream function ψ *at the point P.*

Solution. Given : $\phi = x\,(2y - 1)$

(*i*) The velocity components in the direction of x and y are

$$u = -\frac{\partial \phi}{\partial x} = -\frac{\partial}{\partial x}[x\,(2y - 1)] = -[2y - 1] = 1 - 2y$$

$$v = -\frac{\partial \phi}{\partial y} = -\frac{\partial}{\partial y}[x\,(2y - 1)] = -[2x] = -2x$$

At the point P (4, 5), *i.e.*, at $x = 4$, $y = 5$

$u = 1 - 2 \times 5 = -9$ units/sec

$v = -2 \times 4 = -8$ units/sec

$\therefore$ Velocity at $P = -9i - 8j$

or Resultant velocity at $P = \sqrt{9^2 + 8^2} = \sqrt{81 + 64} = 12.04$ units/sec = **12.04 units/sec. Ans.**

(*ii*) **Value of Stream Function at P**

We know that $\frac{\partial \psi}{\partial y} = -u = -(1 - 2y) = 2y - 1$ ...(*i*)

and $\frac{\partial \psi}{\partial x} = v = -2x$ ...(*ii*)

Integrating equation (*i*) w.r.t. 'y', we get

$$\int d\psi = \int (2y - 1)\,dy \text{ or } \psi = \frac{2y^2}{2} - y + \text{Constant of integration.}$$

The constant of integration is not a function of y but it can be a function of x. Let the value of constant of integration is k. Then

$$\psi = y^2 - y + k. \qquad ...(iii)$$

Differentiating the above equation w.r.t. 'x', we get

$$\frac{\partial \psi}{\partial x} = \frac{\partial k}{\partial x}.$$

But from equation (*ii*), $\frac{\partial \psi}{\partial x} = -2x$

Equating the value of $\frac{\partial \psi}{\partial x}$, we get $\frac{\partial k}{\partial x} = -2x$.

Integrating this equation, we get $k = \int -2x dx = -\frac{2x^2}{2} = -x^2$.

Substituting this value of k in equation (*iii*), we get $\psi = \mathbf{y^2 - y - x^2}$. **Ans.**

$\therefore$ Stream function ψ at P (4, 5) $= 5^2 - 5 - 4^2 = 25 - 5 - 16 =$ **4 units. Ans.**

Problem 5.14 *The stream function for a two-dimensional flow is given by $\psi = 2xy$, calculate the velocity at the point P (2, 3). Find the velocity potential function ϕ.*

Solution. Given : $\psi = 2xy$

The velocity components u and v in terms of ψ are

$$u = -\frac{\partial \psi}{\partial y} = -\frac{\partial}{\partial y}(2xy) = -2x$$

$$v = \frac{\partial \psi}{\partial x} = \frac{\partial}{\partial x}(2xy) = 2y.$$

At the point P (2, 3), we get $u = -2 \times 2 = -4$ units/sec

$$v = 2 \times 3 = 6 \text{ units/sec}$$

$\therefore$ Resultant velocity at $P = \sqrt{u^2 + v^2} = \sqrt{4^2 + 6^2} = \sqrt{16 + 36} = \sqrt{52} = 7.21$ units/sec.

Velocity Potential Function ϕ

We know $$\frac{\partial \phi}{\partial x} = -u = -(-2x) = 2x \qquad ...(i)$$

$$\frac{\partial \phi}{\partial y} = -v = -2y \qquad ...(ii)$$

Integrating equation (*i*), we get

$$\int d\phi = \int 2x dx$$

or $$\phi = \frac{2x^2}{2} + C = x^2 + C \qquad ...(iii)$$

where C is a constant which is independent of x but can be a function of y.

Differentiating equation (*iii*) w.r.t. 'y', we get $\frac{\partial \phi}{\partial y} = \frac{\partial C}{\partial y}$

But from (*ii*), $\frac{\partial \phi}{\partial y} = -2y$

$\therefore$ $\frac{\partial C}{\partial y} = -2y$

Integrating this equation, we get $C = \int -2y\, dy = -\frac{2y^2}{2} = -y^2$

Substituting this value of C in equation (*iii*), we get $\phi = \mathbf{x^2 - y^2}$. **Ans.**

Problem 5.15 *Sketch the stream lines represented by* $\psi = x^2 + y^2$. *Also find out the velocity and its direction at point (1, 2).*

Solution. Given : $\psi = x^2 + y^2$

The velocity components u and v are

$$u = -\frac{\partial \psi}{\partial y} = -\frac{\partial}{\partial y}(x^2 + y^2) = -2y$$

$$v = \frac{\partial \psi}{\partial x} = \frac{\partial}{\partial x}(x^2 + y^2) = 2x$$

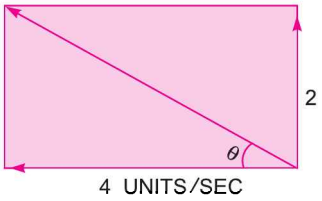


Fig. 5.9

At the point (1, 2), the velocity components are

$u = -2 \times 2 = -4$ units/sec

$v = 2 \times 1 = 2$ units/sec

Resultant velocity $= \sqrt{u^2 + v^2} = \sqrt{(-4)^2 + 2^2}$

$= \sqrt{20} = 4.47$ units/sec

and $\tan \theta = \frac{v}{u} = \frac{2}{4} = \frac{1}{2}$

$\therefore$ $\theta = \tan^{-1} .\, 5 = 26° 34'$

$\therefore$ Resultant velocity makes an angle of 26° 34′ with x-axis.

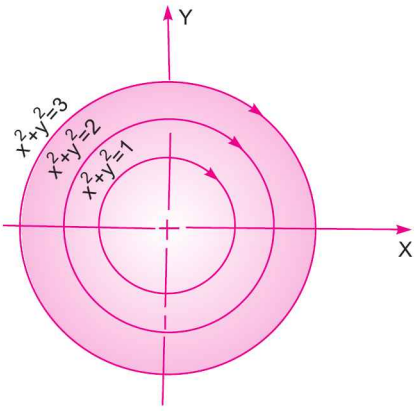


Fig. 5.10

Sketch of Stream Lines

$\psi = x^2 + y^2$

Let $\psi = 1, 2, 3$ and so on.

Then we have $1 = x^2 + y^2$

$2 = x^2 + y^2$

$3 = x^2 + y^2$

and so on.

Each equation is a equation of a circle. Thus we shall get concentric circles of different diameters as shown in Fig. 5.10.

Problem 5.16 *The velocity components in a two-dimensional flow field for an incompressible fluid are as follows :*

$$u = \frac{y^3}{3} + 2x - x^2 y \text{ and } v = xy^2 - 2y - x^3/3$$

obtain an expression for the stream function ψ.

Solution. Given : $u = y^3/3 + 2x - x^2y$

$v = xy^2 - 2y - x^3/3.$

The velocity components in terms of stream function are

$$\frac{\partial \psi}{\partial x} = v = xy^2 - 2y - x^3/3 \qquad ...(i)$$

$$\frac{\partial \psi}{\partial y} = -u = -y^3/3 - 2x + x^2y \qquad ...(ii)$$

Integrating (i) w.r.t. x, we get $\psi = \int (xy^2 - 2y - x^3/3)\, dx$

or $$\psi = \frac{x^2y^2}{2} - 2xy - \frac{x^4}{4 \times 3} + k, \qquad ...(iii)$$

where k is a constant of integration which is independent of x but can be a function of y.

Differentiating equation (iii) w.r.t. y, we get

$$\frac{\partial \psi}{\partial y} = \frac{2x^2y}{2} - 2x + \frac{\partial k}{\partial y} = x^2y - 2x + \frac{\partial k}{\partial y}$$

But from (ii), $$\frac{\partial \psi}{\partial y} = -y^3/3 - 2x + x^2y$$

Comparing the value of $\frac{\partial \psi}{\partial y}$, we get $x^2y - 2x + \frac{\partial k}{\partial y} = -y^3/3 - 2x + x^2y$

$\therefore$ $$\frac{\partial k}{\partial y} = -y^3/3$$

Integrating, we get $$k = \int (-y^3/3)\, dy = \frac{-y^4}{4 \times 3} = \frac{-y^4}{12}$$

Substituting this value in (iii), we get

$$\psi = \frac{\mathbf{x^2y^2}}{\mathbf{2}} - \mathbf{2xy} - \frac{\mathbf{x^4}}{\mathbf{12}} - \frac{\mathbf{y^4}}{\mathbf{12}}. \textbf{ Ans.}$$

Problem 5.17 *In a two-dimensional incompressible flow, the fluid velocity components are given by* $u = x - 4y$ *and* $v = -y - 4x$.

Show that velocity potential exists and determine its form. Find also the stream function.

Solution. Given : $u = x - 4y$ and $v = -y - 4x$

$\therefore$ $$\frac{\partial u}{\partial x} = 1 \quad \text{and} \quad \frac{\partial v}{\partial y} = -1$$

$\therefore$ $$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 1 - 1 = 0$$

Hence flow is continuous and velocity potential exists.

Let ϕ = Velocity potential.

Let velocity components in terms of velocity potential is given by

$$\frac{\partial \phi}{\partial x} = -u = -(x - 4y) = -x + 4y \qquad ...(i)$$

and

$$\frac{\partial \phi}{\partial y} = -v = -(-y - 4x) = y + 4x \qquad ...(ii)$$

Integrating equation (*i*), we get $\phi = -\frac{x^2}{2} + 4xy + C$ ...(*iii*)

where C is a constant of integration, which is independent of x.

This constant can be a function of y.

Differentiating the above equation, *i.e.*, equation (*iii*) with respect to 'y', we get

$$\frac{\partial \phi}{\partial y} = 0 + 4x + \frac{\partial C}{\partial y}$$

But from equation (*iii*), we have $\frac{\partial \phi}{\partial y} = y + 4x$

Equating the two values of $\frac{\partial \phi}{\partial y}$, we get

$$4x + \frac{\partial C}{\partial y} = y + 4x \quad \text{or} \quad \frac{\partial C}{\partial y} = y$$

Integrating the above equation, we get

$$C = \frac{y^2}{2} + C_1$$

where C_1 is a constant of integration, which is independent of x and y.

Taking it equal to zero, we get $C = \frac{y^2}{2}$.

Substituting the value of C in equation (*iii*), we get

$$\phi = -\frac{x^2}{2} + 4xy + \frac{y^2}{2}. \textbf{ Ans.}$$

Value of Stream functions

Let ψ = Stream function

The velocity components in terms of stream function are

$$\frac{\partial \psi}{\partial x} = v = -y - 4x \qquad ...(iv)$$

and

$$\frac{\partial \psi}{\partial y} = -u = -(x - 4y) = -x + 4y \qquad ...(v)$$

Integrating equation (*iv*) w.r.t. x, we get

$$\psi = -yx - \frac{4x^2}{2} + k \qquad ...(vi)$$

where k is a constant of integration which is independent of x but can be a function of y.

Differentiating equation (*vi*) w.r.t. y, we get $\frac{\partial \psi}{\partial y} = -x - 0 + \frac{\partial k}{\partial y}$

But from equation (*v*), we have $\frac{\partial \psi}{\partial y} = -x + 4y$

Equating the two values of $\frac{\partial \psi}{\partial y}$, we get $-x + \frac{\partial k}{\partial y} = -x + 4y$ or $\frac{\partial k}{\partial y} = 4y$

Integrating the above equation, we get $k = \frac{4y^2}{2} = 2y^2$

Substituting the value of k in equation (*vi*), we get

$$\psi = -yx - 2x^2 + 2y^2. \text{ Ans.}$$

▶ 5.9 TYPES OF MOTION

A fluid particle while moving may undergo anyone or combination of following four types of displacements :

(*i*) Linear Translation or Pure Translation,
(*ii*) Linear Deformation,
(*iii*) Angular Deformation, and
(*iv*) Rotation.

5.9.1 Linear Translation. It is defined as the movement of a fluid element in such a way that it moves bodily from one position to another position and the two axes ab and cd represented in new positions by $a'b'$ and $c'd'$ are parallel as shown in Fig. 5.11 (*a*).

5.9.2 Linear Deformation. It is defined as the deformation of a fluid element in linear direction when the element moves. The axes of the element in the deformed position and un-deformed position are parallel, but their lengths change as shown in Fig. 5.11 (*b*).

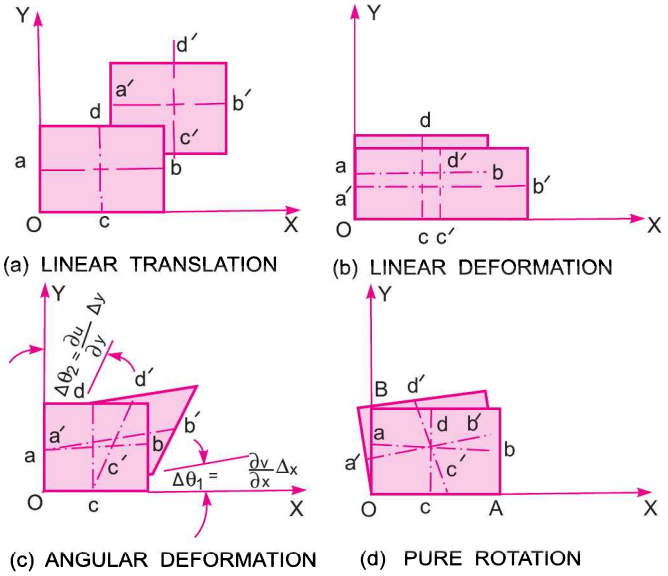


Fig. 5.11. *Displacement of a fluid element.*

5.9.3 Angular Deformation or Shear Deformation. It is defined as the average change in the angle contained by two adjacent sides. Let $\Delta\theta_1$ and $\Delta\theta_2$ is the change in angle between two adjacent sides of a fluid element as shown in Fig. 5.11 (*c*), then angular deformation or shear strain rate

$$= \frac{1}{2}[\Delta\theta_1 + \Delta\theta_2]$$

Now $$\Delta\theta_1 = \frac{\partial v}{\partial x} \times \frac{\Delta x}{\Delta x} = \frac{\partial v}{\partial x} \text{ and } \Delta\theta_2 = \frac{\partial u}{\partial y} . \frac{\Delta y}{\Delta y} = \frac{\partial u}{\partial y}.$$

$\therefore$ **Angular deformation** $= \frac{1}{2}[\Delta\theta_1 + \Delta\theta_2]$

or **Shear strain rate** $= \frac{1}{2}\left[\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right]$ ...(5.16)

5.9.4 Rotation. It is defined as the movement of a fluid element in such a way that both of its axes (horizontal as well as vertical) rotate in the same direction as shown in Fig. 5.11 (*d*). It is equal to $\frac{1}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)$ for a two-dimensional element in *x*-*y* plane. The rotational components are

$$\left.\begin{aligned} \omega_z &= \frac{1}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) \\ \omega_x &= \frac{1}{2}\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right) \\ \omega_y &= \frac{1}{2}\left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right) \end{aligned}\right\} \quad ...(5.17)$$

5.9.5 Vorticity. It is defined as the value twice of the rotation and hence it is given as 2ω.

Problem 5.18 *A fluid flow is given by* $V = 8x^3 i - 10x^2 yj$.

Find the shear strain rate and state whether the flow is rotational or irrotational.

Solution. Given : $V = 8x^3 i - 10x^2 yj$

$\therefore$ $$u = 8x^3, \frac{\partial u}{\partial x} = 24x^2, \frac{\partial u}{\partial y} = 0$$

and $$v = -10x^2 y, \frac{\partial v}{\partial x} = -20xy, \frac{\partial v}{\partial y} = -10x^2$$

(*i*) Shear strain rate is given by equation (5.16) as

$$= \frac{1}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) = \frac{1}{2}(-20xy + 0) = \mathbf{-10xy.\ Ans.}$$

(*ii*) Rotation in x - y plane is given by equation (5.17) or

$$\omega_z = \frac{1}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) = \frac{1}{2}(-20xy - 0) = -10xy$$

As rotation $\omega_z \neq 0$. Hence flow is rotational. **Ans.**

Problem 5.19 *The velocity components in a two-dimensional flow are*

$$u = y^3/3 + 2x - x^2y \text{ and } v = xy^2 - 2y - x^3/3.$$

Show that these components represent a possible case of an irrotational flow.

Solution. Given : $u = y^3/3 + 2x - x^2y$

$$\therefore \quad \frac{\partial u}{\partial x} = 2 - 2xy$$

$$\frac{\partial u}{\partial y} = \frac{3y^2}{3} - x^2 = y^2 - x^2$$

Also $v = xy^2 - 2y - x^3/3$

$$\therefore \quad \frac{\partial v}{\partial y} = 2xy - 2$$

$$\frac{\partial v}{\partial x} = y^2 - \frac{3x^2}{3} = y^2 - x^2.$$

(*i*) For a two-dimensional flow, continuity equation is $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$

Substituting the value of $\frac{\partial u}{\partial x}$ and $\frac{\partial v}{\partial y}$, we get

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 2 - 2xy + 2xy - 2 = 0$$

$\therefore$ It is a possible case of fluid flow.

(*ii*) Rotation, ω_z is given by $\omega_z = \frac{1}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) = \frac{1}{2}[(y^2 - x^2) - (y^2 - x^2)] = 0$

$\therefore$ Rotation is zero, which means it is case of irrotational flow. **Ans.**

▶ 5.10 VORTEX FLOW

Vortex flow is defined as the flow of a fluid along a curved path or the flow of a rotating mass of fluid is known a 'Vortex Flow'. The vortex flow is of two types namely :

1. Forced vortex flow, and
2. Free vortex flow.

5.10.1 Forced Vortex Flow. Forced vortex flow is defined as that type of vortex flow, in which some external torque is required to rotate the fluid mass. The fluid mass in this type of flow, rotates at constant angular velocity, ω. The tangential velocity of any fluid particle is given by

$$v = \omega \times r \qquad ...(5.18)$$

where r = Radius of fluid particle from the axis of rotation.

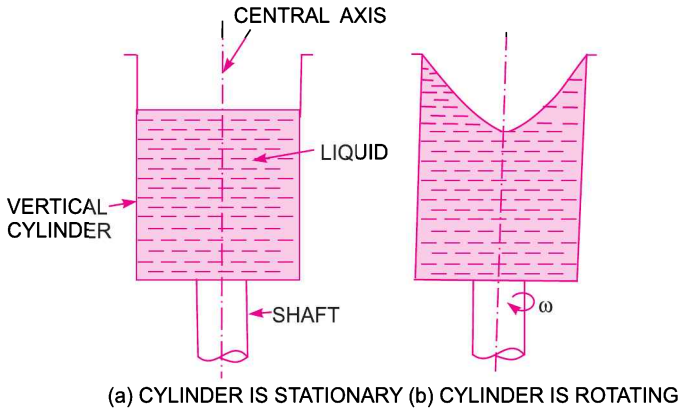


Fig. 5.12 *Forced vortex flow.*

Hence angular velocity ω is given by

$$\omega = \frac{v}{r} = \text{Constant.} \qquad ...(5.19)$$

Examples of forced vortex are :

1. A vertical cylinder containing liquid which is rotated about its central axis with a constant angular velocity ω, as shown in Fig. 5.12.
2. Flow of liquid inside the impeller of a centrifugal pump.
3. Flow of water through the runner of a turbine.

5.10.2 Free Vortex Flow. When no external torque is required to rotate the fluid mass, that type of flow is called free vortex flow. Thus the liquid in case of free vortex is rotating due to the rotation which is imparted to the fluid previously.

Examples of the free vortex flow are :

1. Flow of liquid through a hole provided at the bottom of a container.
2. Flow of liquid around a circular bend in a pipe.
3. A whirlpool in a river.
4. Flow of fluid in a centrifugal pump casing.

The relation between velocity and radius, in free vortex is obtained by putting the value of external torque equal to zero, or, the time rate of change of angular momentum, *i.e.*, moment of momentum must be zero. Consider a fluid particle of mass 'm' at a radial distance r from the axis of rotation, having a tangential velocity v. Then

Angular momentum = Mass × Velocity = $m \times v$

Moment of momentum = Momentum × r = $m \times v \times r$

$\therefore$ Time rate of change of angular momentum = $\frac{\partial}{\partial t}(mvr)$

$\therefore$ For free vortex $\frac{\partial}{\partial t}(mvr) = 0$

Integrating, we get $mvr = \text{Constant}$ or $vr = \frac{\text{Constant}}{m} = \text{Constant}$ ...(5.20)

5.10.3 Equation of Motion for Vortex Flow. Consider a fluid element *ABCD* (shown shaded) in Fig. 5.13 rotating at a uniform velocity in a horizontal plane about an axis perpendicular to the plane of paper and passing through *O*.

Let r = Radius of the element from O.

$\Delta\theta$ = Angle subtended by the element at O.

Δr = Radial thickness of the element.

ΔA = Area of cross-section of element.

The forces acting on the element are :

(*i*) Pressure force, $p\Delta A$, on the face *AB*.

(*ii*) Pressure force, $\left(p + \frac{\partial p}{\partial r}\Delta r\right)\Delta A$ on the face *CD*.

(*iii*) Centrifugal force, $\frac{mv^2}{r}$ acting in the direction away from the centre, *O*.

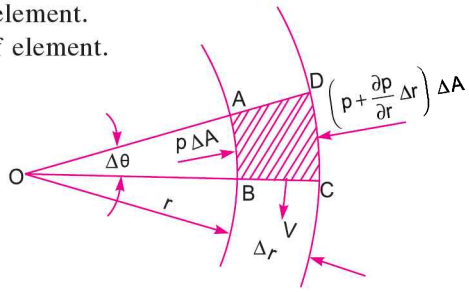


Fig. 5.13

Now, the mass of the element = Mass density × Volume

$= \rho \times \Delta A \times \Delta r$

$\therefore$ Centrifugal force $= \rho\Delta A\Delta r \frac{v^2}{r}$.

Equating the forces in the radial direction, we get

$$\left(p + \frac{\partial p}{\partial r}\Delta r\right)\Delta A - p\Delta A = \rho\Delta A\Delta r \frac{v^2}{r}$$

or $$\frac{\partial p}{\partial r}\Delta r\,\Delta A = \rho\Delta A\Delta r \frac{v^2}{r}.$$

Cancelling $\Delta r \times \Delta A$ from both sides, we get $\frac{\partial p}{\partial r} = \rho \frac{v^2}{r}$ ...(5.21)

Equation (5.21) gives the pressure variation along the radial direction for a forced or free vortex flow in a horizontal plane. The expression $\frac{\partial p}{\partial r}$ is called pressure gradient in the radial direction. As $\frac{\partial p}{\partial r}$ is positive, hence pressure increases with the increase of radius '*r*'.

The pressure variation in the vertical plane is given by the hydrostatic law, *i.e.*,

$$\frac{\partial p}{\partial z} = -\rho g \qquad ...(5.22)$$

In equation (5.22), z is measured vertically in the upward direction.

The pressure, p varies with respect to r and z or p is a function of r and z and hence total derivative of p is

$$dp = \frac{\partial p}{\partial r}dr + \frac{\partial p}{\partial z}dz.$$

Substituting the values of $\frac{\partial p}{\partial r}$ from equation (5.21) and $\frac{\partial p}{\partial z}$ from equation (5.22), we get

$$dp = \rho \frac{v^2}{r} dr - \rho g dz \qquad ...(5.23)$$

Equation (5.23) gives the variation of pressure of a rotating fluid in any plane.

5.10.4 Equation of Forced Vortex Flow. For the forced vortex flow, from equation (5.18), we have

$$v = \omega \times r$$

where ω = Angular velocity = Constant.

Substituting the value of v in equation (5.23), we get

$$dp = \rho \times \frac{\omega^2 r^2}{r} dr - \rho g\, dz.$$

Consider two points 1 and 2 in the fluid having forced vortex flow as shown in Fig. 5.14. Integrating the above equation for points 1 and 2, we get

$$\int_1^2 dp = \int_1^2 \rho\omega^2 r dr - \int_1^2 \rho g dz$$

or
$$(p_2 - p_1) = \left[\rho\omega^2 \frac{r^2}{2}\right]_1^2 - \rho g [z]_1^2$$

or
$$(p_2 - p_1) = \frac{\rho\omega^2}{2} [r_2^2 - r_1^2] - \rho g [z_2 - z_1]$$

$$= \frac{\rho}{2} [\omega^2 r_2^2 - \omega^2 r_1^2] - \rho g [z_2 - z_1]$$

$$= \frac{\rho}{2} [v_2^2 - v_1^2] - \rho g [z_2 - z_1] \quad \left\{ \because \begin{array}{l} v_2 = \omega r_2 \\ v_1 = \omega r_1 \end{array} \right\}$$

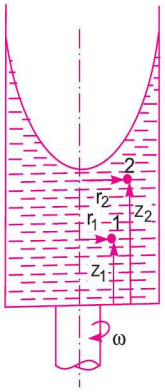


Fig. 5.14

If the points 1 and 2 lie on the free surface of the liquid, then $p_1 = p_2$ and hence above equation becomes

$$0 = \frac{\rho}{2} [v_2^2 - v_1^2] - \rho g [z_2 - z_1]$$

or
$$\rho g [z_2 - z_1] = \frac{\rho}{2} [v_2^2 - v_1^2]$$

or
$$[z_2 - z_1] = \frac{1}{2g} [v_2^2 - v_1^2].$$

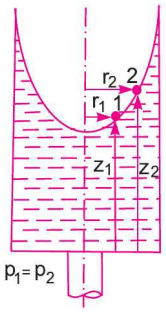


Fig. 5.15

If the point 1 lies on the axis of rotation, then $v_1 = \omega \times r_1 = \omega \times 0 = 0$. The above equation becomes as

$$z_2 - z_1 = \frac{1}{2g} v_2^2 = \frac{v_2^2}{2g}$$

Let
$$z_2 - z_1 = Z, \text{ then we have } Z = \frac{v_2^2}{2g} = \frac{\omega^2 \times r_2^2}{2g} \qquad ...(5.24)$$

Thus Z varies with the square of r. Hence equation (5.24) is an equation of parabola. This means the free surface of the liquid is a paraboloid.

Problem 5.20 *Prove that in case of forced vortex, the rise of liquid level at the ends is equal to the fall of liquid level at the axis of rotation.*

Solution. Let R = radius of the cylinder.

O-O = Initial level of liquid in cylinder when the cylinder is not rotating.

$\therefore$ Initial height of liquid $= (h + x)$

$\therefore$ Volume of liquid in cylinder = $\pi R^2 \times$ Height of liquid

$= \pi R^2 \times (h + x)$ ...(i)

Let the cylinder is rotated at constant angular velocity ω. The liquid will rise at the ends and will fall at the centre.

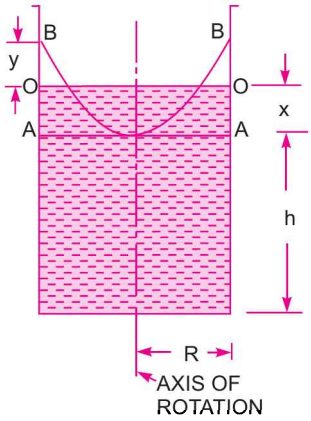


Fig. 5.16

Let y = Rise of liquid at the ends from O-O

x = Fall of liquid at the centre from O-O.

Then volume of liquid

= [Volume of cylinder upto level B-B]

$-$ [Volume of paraboloid]

$= [\pi R^2 \times$ Height of liquid upto level B-$B]$

$$- \left[\frac{\pi R^2}{2} \times \text{Height of paraboloid}\right]$$

$$= \pi R^2 \times (h + x + y) - \frac{\pi R^2}{2} \times (x + y)$$

$$= \pi R^2 \times h + \pi R^2 (x + y) - \frac{\pi R^2}{2} \times (x + y)$$

$$= \pi R^2 \times h + \frac{\pi R^2}{2} (x + y) \quad \text{...}(ii)$$

Equating (i) and (ii), we get

$$\pi R^2 (h + x) = \pi R^2 \times h + \frac{\pi R^2}{2} (x + y)$$

or $$\pi R^2 h + \pi R^2 x = \pi R^2 \times h + \frac{\pi R^2}{2} x + \frac{\pi R^2}{2} y$$

or $$\pi R^2 x - \frac{\pi R^2}{2} x = \frac{\pi R^2}{2} y \quad \text{or} \quad \frac{\pi R^2}{2} x = \frac{\pi R^2}{2} y \quad \text{or} \quad x = y$$

or Fall of liquid at centre = Rise of liquid at the ends.

Problem 5.21 *An open circular tank of 20 cm diameter and 100 cm long contains water upto a height of 60 cm. The tank is rotated about its vertical axis at 300 r.p.m., find the depth of parabola formed at the free surface of water.*

Solution. Given :

Diameter of cylinder = 20 cm

$\therefore$ Radius, $R = \frac{20}{2} = 10$ cm

Height of liquid, $H = 60$ cm

Speed, $N = 300$ r.p.m.

Angular velocity, $\omega = \dfrac{2\pi N}{60} = \dfrac{2 \times \pi \times 300}{60} = 31.41$ rad/sec.

Let the depth of parabola $= Z$

Using equation (5.24), $Z = \dfrac{\omega^2 r_2^{\,2}}{2g}$, where $r_2 = R$

$$= \frac{\omega^2 R^2}{2g} = \frac{(31.41)^2 \times (10)^2}{2 \times 981} = \textbf{50.28 cm. Ans.}$$

Problem 5.22 *An open circular cylinder of 15 cm diameter and 100 cm long contains water upto a height of 80 cm. Find the maximum speed at which the cylinder is to be rotated about its vertical axis so that no water spills.*

Solution. Given :

Diameter of cylinder $= 15$ cm

$\therefore$ Radius, $R = \dfrac{15}{2} = 7.5$ cm

Length of cylinder, $L = 100$ cm

Initial height of water $= 80$ cm.

Let the cylinder is rotated at an angular speed of ω rad/sec, when the water is about to spill. Then using,

Rise of liquid at ends = Fall of liquid at centre

But rise of liquid at ends = Length – Initial height

$= 100 - 80 = 20$ cm

$\therefore$ Fall of liquid at centre $= 20$ cm

$\therefore$ Height of parabola $= 20 + 20 = 40$ cm

$\therefore$ $Z = 40$ cm

Using the relation, $Z = \dfrac{\omega^2 R^2}{2g}$, we get $40 = \dfrac{\omega^2 (7.5)^2}{2 \times 981}$

$\therefore$ $\omega^2 = \dfrac{40 \times 2 \times 981}{7.5 \times 7.5} = 1395.2$

$\therefore$ $\omega = \sqrt{1395.2} = 37.35$ rad/s

$\therefore$ Speed, N is given by $\omega = \dfrac{2\pi N}{60}$

or $N = \dfrac{60 \times \omega}{2\pi} = \dfrac{60 \times 37.35}{2 \times \pi} = \textbf{356.66 r.p.m. Ans.}$

Problem 5.23 *A cylindrical vessel 12 cm in diameter and 30 cm deep is filled with water upto the top. The vessel is open at the top. Find the quantity of liquid left in the vessel, when it is rotated about its vertical axis with a speed of (a) 3000 r.p.m., and (b) 600 r.p.m.*

Solution. Given :

Diameter of cylinder $= 12$ cm

$\therefore$ Radius, $R = 6$ cm

Initial height of water $= 30$ cm

Initial volume of water = Area × Initial height of water

$$= \frac{\pi}{4} \times 12^2 \times 30 \text{ cm}^3 = 3392.9 \text{ cm}^3$$

(*a*) Speed, $N = 300$ r.p.m.

$$\therefore \quad \omega = \frac{2\pi N}{60} = \frac{2\pi \times 300}{60} = 31.41 \text{ rad/s}$$

Height of parabola is given by $Z = \frac{\omega^2 R^2}{2g} = \frac{(31.41)^2 \times 6^2}{2 \times 981} = 18.10$ cm.

As vessel is initially full of water, water will be spilled if it is rotated. Volume of water spilled is equal to the volume of paraboloid.

But volume of paraboloid = [Area of cross-section × Height of parabola] ÷ 2

$$= \frac{\pi}{4} D^2 \times \frac{Z}{2} = \frac{\pi}{4} \times 12^2 \times \frac{18.10}{2} = 1023.53 \text{ cm}^3$$

Volume of water left = Initial volume – Volume of water spilled

= 3392.9 – 1023.53 = **2369.37 cm^3. Ans.**

(*b*) Speed, $N = 600$ r.p.m.

$$\therefore \quad \omega = \frac{2\pi N}{60} = \frac{2\pi \times 600}{60} = 62.82 \text{ rad/s}$$

Height of parabola, $Z = \frac{\omega^2 R^2}{2g} = \frac{(62.82)^2 \times 6^2}{2 \times 981} = 72.40$ cm.

As the height of parabola is more than the height of cylinder the shape of imaginary parabola will be as shown in Fig. 5.17.

Let r = Radius of the parabola at the bottom of the vessel.

Height of imaginary parabola

= 72.40 – 30 = 42.40 cm.

Volume of water left in the vessel

= Volume of water in portions *ABC* and *DEF*

= Initial volume of water

– Volume of paraboloid *AOF*

+ Volume of paraboloid *COD*.

Now volume of paraboloid

$$AOF = \frac{\pi}{4} \times D^2 \times \text{Height of parabola}$$

$$= \frac{\pi}{4} \times 12^2 \times \frac{72.4^2}{2} = 4094.12 \text{ cm}^3$$

For the imaginary parabola (*COD*), $\omega = 62.82$ rad/sec

$Z = 42.4$ cm

r = Radius at the bottom of vessel

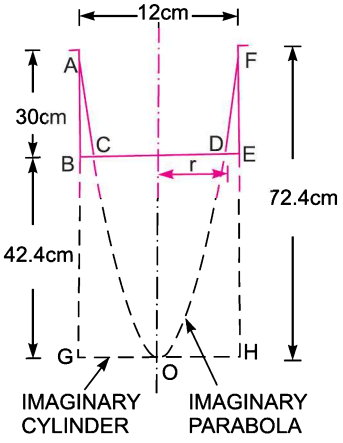


Fig. 5.17

Using the relation $Z = \frac{\omega^2 r^2}{2g}$, we get $42.4 = \frac{62.82^2 \times r^2}{2 \times 981}$

$\therefore \quad r^2 = \frac{2 \times 981 \times 42.40}{62.82 \times 62.82} = 21.079$

$\therefore \quad r = \sqrt{21.079} = 4.59$ cm

$\therefore$ Volume of paraboloid *COD*

$= \frac{1}{2} \times$ Area at the top of the imaginary parabola $\times$ Height of parabola

$= \frac{1}{2} \times \pi r^2 \times 42.4 = \frac{1}{2} \times \pi \times 4.59^2 \times 42.4 = 1403.89$ cm^3

$\therefore$ Volume of water left = 3392.9 – 4094.12 + 1403.89 = **702.67 cm^3. Ans.**

Problem 5.24 *An open circular cylinder of 15 cm diameter and 100 cm long contains water upto a height of 70 cm. Find the speed at which the cylinder is to be rotated about its vertical axis, so that the axial depth becomes zero.*

Solution. Given :

Diameter of cylinder = 15 cm

$\therefore$ Radius, $R = \frac{15}{2} = 7.5$ cm

Length of cylinder = 100 cm

Initial height of water = 70 cm.

When axial depth is zero, the depth of paraboloid = 100 cm.

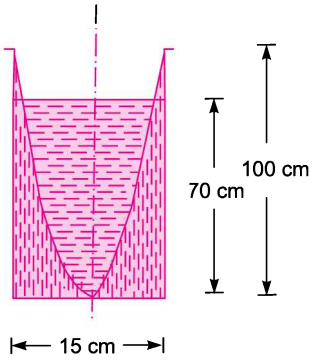


Fig. 5.18

Using the relation, $Z = \frac{\omega^2 R^2}{2g}$, we get

$100 = \frac{\omega^2 \times 7.5^2}{2 \times 9.81}$

$\therefore \quad \omega^2 = \frac{100 \times 2 \times 9.81}{7.5 \times 7.5}$

$\therefore \quad \omega = \sqrt{\frac{100 \times 2 \times 9.81}{7.5 \times 7.5}} = \frac{442.92}{7.5} = 59.05$ rad/s

$\therefore$ Speed, N is given by $\omega = \frac{2\pi H}{60}$

or $N = \frac{60 \times \omega}{2\pi} = \frac{60 \times 59.05}{2\pi} =$ **563.88 r.p.m. Ans.**

Problem 5.25 *For the problem (5.24), find the difference in total pressure force (i) at the bottom of cylinder, and (ii) at the sides of the cylinder due to rotation.*

Solution. (*i*) The data is given in Problem 5.24. The difference in total pressure force at the bottom of cylinder is obtained by finding total hydrostatic force at the bottom before rotation and after rotation.

Before rotation, force $= \rho g A \bar{h}$

where $\rho = 1000$ kg/m^3, A = Area of bottom $= \frac{\pi}{3} D^2 = \frac{\pi}{4} \times (0.15)^2$ m^2, $\bar{h} = 70$ cm $= 0.70$ m

$$\therefore \quad \text{Force} = 1000 \times 9.81 \times \frac{\pi}{4} \times (0.15)^2 \times 0.7 \text{ N} = 121.35 \text{ N}$$

After rotation, the depth of water at the bottom is not constant and hence pressure force due to the height of water, will not be constant. Consider a circular ring of radius r and width dr as shown in Fig. 5.19. Let the height of water from the bottom of the tank upto free surface of water at a radius

$$r = Z = \frac{\omega^2 r^2}{2g}.$$

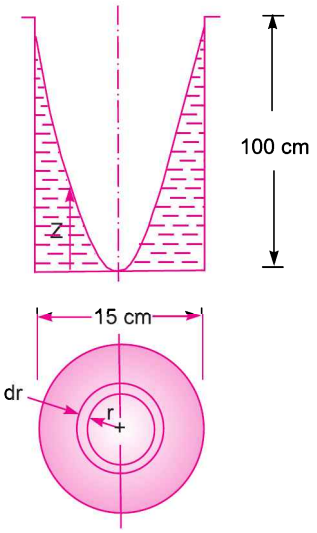


Fig. 5.19

Hydrostatic force on ring at the bottom,

$$dF = \rho g \times \text{Area of ring} \times Z$$

$$= 1000 \times 9.81 \times 2\pi r dr \times \frac{\omega^2 r^2}{2g}$$

$$= 9810 \times 2 \times \pi r \times \frac{\omega^2 r^2}{2g} \times dr$$

$\therefore$ Total pressure force at the bottom

$$= \int dF = \int_0^R 9810 \times 2 \times \pi r \times \frac{\omega^2 r^2}{2g} dr$$

$$= \int_0^{0.075} 19620 \times \pi \times \frac{\omega^2}{2g} r^3 \, dr$$

From Problem 5.24, $\omega = 59.05$ rad/s

$R = 7.5$ cm $= .075$ m.

Substituting these values, we get total pressure force

$$= \frac{19620 \times \pi \times (59.05)^2}{2 \times 9.81} \left[\frac{r^4}{4}\right]_0^{.075}$$

$$= \frac{19620 \times \pi \times (59.05)^2}{2 \times 9.81} \times \frac{(.075)^4}{4} = 86.62 \text{ N}$$

$\therefore$ Difference in pressure forces at the bottom

$121.35 - 86.62 =$ **34.73 N. Ans.**

(*ii*) Forces on the sides of the cylinder

Before rotation $= \rho g A \bar{h}$

where A = Surface area of the sides of the cylinder upto height of water

$= \pi D \times$ Height of water $= \pi \times .15 \times 0.70$ m$^2 = 0.33$ m^2

$\bar{h}$ = C.G. of the wetted area of the sides

$$= \frac{1}{2} \times \text{height of water} = \frac{0.70}{2} = 0.35 \text{ m}$$

$\therefore$ Force on the sides before rotation = $1000 \times 9.81 \times 0.33 \times 0.35 = 1133$ N

After rotation, the water is upto the top of the cylinder and hence force on the sides

$$= 1000 \times 9.81 \times \text{Wetted area of the sides} \times \frac{1}{2} \times \text{Height of water}$$

$$= 9810 \times \pi D \times 1.0 \times \frac{1}{2} \times 1.0 = 9810 \times \pi \times .15 \times \frac{1}{2} = 2311.43 \text{ N}$$

$\therefore$ Difference in pressure on the sides

$2311.43 - 1133 =$ **1178.43 N. Ans.**

5.10.5 Closed Cylindrical Vessels. If a cylindrical vessel is closed at the top, which contains some liquid, the shape of paraboloid formed due to rotation of the vessel will be as shown in Fig. 5.20 for different speed of rotations.

Fig. 5.20 (*a*) shows the initial stage of the cylinder, when it is not rotated. Fig. 5.20 (*b*) shows the shape of the paraboloid formed when the speed of rotation is ω_1. If the speed is increased further say ω_2, the shape of paraboloid formed will be as shown in Fig. 5.20 (*c*). In this case the radius of the parabola at the top of the vessel is unknown. Also the height of the paraboloid formed corresponding to angular speed ω_2 is unknown. Thus to solve the two unknown, we should have two equations. One equation is

$$Z = \frac{\omega_2^2 r^2}{2g}$$

The second equation is obtained from the fact that for closed vessel, volume of air before rotation is equal to the volume of air after rotation.

Volume of air before rotation = Volume of closed vessel – Volume of liquid in vessel

Volume of air after rotation = Volume of paraboloid formed = $\frac{\pi r^2 \times Z}{2}$.

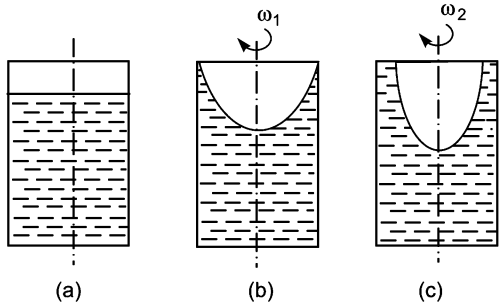


Fig. 5.20

Problem 5.26 *A vessel, cylindrical in shape and closed at the top and bottom, contains water upto a height of 80 cm. The diameter of the vessel is 20 cm and length of vessel is 120 cm. The vessel is rotated at a speed of 400 r.p.m. about its vertical axis. Find the height of paraboloid formed.*

Solution. Given :

Initial height of water = 80 cm

Diameter of vessel = 20 cm

$\therefore$ Radius, R = 10 cm

Length of vessel = 120 cm

Speed, N = 400 r.p.m.

$$\therefore \quad \omega = \frac{2\pi N}{60} = \frac{2\pi \times 400}{60} = 41.88 \text{ rad/s}$$

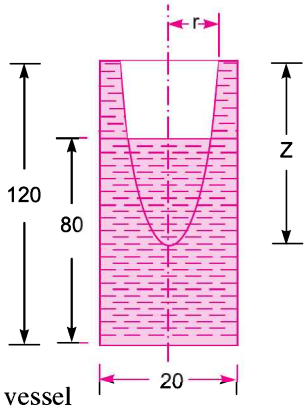


Fig. 5.21

When the vessel is rotated, let Z

= Height of paraboloid formed

r = Radius of paraboloid at the top of the vessel

This is the case of closed vessel.

$\therefore$ Volume of air before rotation = Volume of air after rotation

or $$\frac{\pi}{4} D^2 \times L - \frac{\pi}{4} D^2 \times 80 = \pi r^2 \times \frac{Z}{2}$$

where Z = Height of paraboloid, r = Radius of parabola.

or $$\frac{\pi}{4} D^2 \times 120 - \frac{\pi}{4} D^2 \times 80 = \pi r^2 \times \frac{Z}{2}$$

or $$\frac{\pi}{4} \times D^2 \times (120 - 80) = \frac{\pi}{4} D^2 \times 40 = \pi r^2 \times \frac{Z}{2}$$

or $$\frac{\pi}{4} \times 20^2 \times 40 = 4000 \times \pi = \pi r^2 \times \frac{Z}{2}$$

$$\therefore \quad r^2 \times Z = \frac{4000 \times \pi \times 2}{\pi} = 8000 \qquad ...(i)$$

Using relation $$Z = \frac{\omega^2 r^2}{2g}, \text{ we get } Z = \frac{41.88^2 \times r^2}{2g} = \frac{41.88^2 \times r^2}{2 \times 981} = 0.894\ r^2$$

$$\therefore \quad r^2 = \frac{z}{0.894}$$

Substituting this value of r^2 in (i), we get

$$\frac{Z}{0.894} \times Z = 8000$$

$$\therefore \quad Z^2 = 8000 \times 0.894 = 7152$$

$$\therefore \quad Z = \sqrt{7152} = \textbf{84.56 cm. Ans.}$$

IInd Method

Let Z_1 = Height of paraboloid, if the vessel would not have been closed at the top, corresponding to speed,

N = 400 r.p.m.

or ω = 41.88 rad/s

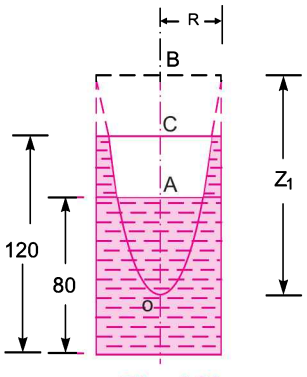


Fig. 5.22

Then $$Z_1 = \frac{\omega^2 R^2}{2g} = \frac{41.88^2 \times 10^2}{2 \times 981} = 89.34 \text{ cm.}$$

Half of Z_1 will be below the initial height of water in the vessel

i.e., $$AO = \frac{Z_1}{2} = \frac{89.34}{2} = 44.67 \text{ cm}$$

But height of paraboloid for closed vessel

$$= CO = CA + AO = (120 - 80) + 44.67 \text{ cm}$$

$$= 40 + 44.67 = \textbf{84.67 cm. Ans.}$$

Problem 5.27 *For the data given in Problem 5.26, find the speed of rotation of the vessel, when axial depth of water is zero.*

Solution. Given :

Diameter of vessel = 20 cm

$\therefore$ Radius, R = 10 cm

Initial height of water = 80 cm

Length of vessel = 120 cm

Let ω is the angular speed, when axial depth is zero.

When axial depth is zero, the height of paraboloid is 120 cm and radius of the parabola at the top of the vessel is r.

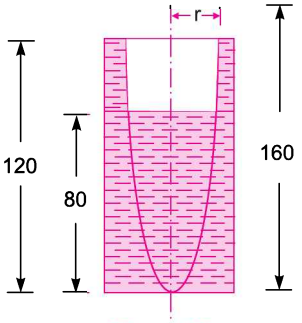


Fig. 5.23

$\therefore$ Using the relation, $$Z = \frac{\omega^2 r^2}{2g} \text{ or } 120 = \frac{\omega^2 \times r^2}{2 \times 980}$$

$\therefore$ $$\omega^2 r^2 = 2 \times 980 \times 120 = 235200 \quad \ldots(i)$$

Volume of air before rotation = Volume of air after paraboloid

$\therefore$ $$\pi R^2 \times (120 - 180) = \text{Volume of paraboloid}$$

$$= \pi r^2 \times \frac{Z}{2}$$

or $$\pi \times 10^2 \times 40 = \frac{\pi r^2 \times Z}{2} = \frac{\pi r^2}{2} \times 120$$

or $$r^2 = \frac{\pi \times 10^2 \times 40 \times 2}{\pi \times 120} = \frac{8000}{120} = 66.67$$

Substituting the value of r^2 in equation (i), we get

$$\omega^2 \times 66.67 = 235200$$

$$\omega = \sqrt{\frac{235200}{66.67}} = 59.4 \text{ rad/s}$$

$\therefore$ Speed N is given by $\omega = \frac{2\pi N}{60}$

or $$N = \frac{60 \times \omega}{2\pi} = \frac{60 \times 59.4}{2\pi} = \textbf{567.22 r.p.m. Ans.}$$

Problem 5.28 *The cylindrical vessel of the problem 5.26 is rotated at 700 r.p.m. about its vertical axis. Find the area uncovered at the bottom of the tank.*

Solution. Given :

Initial height of water = 80 cm

$\therefore$ Diameter of vessel = 20 cm

$\therefore$ Radius, R = 10 cm

Length of vessel = 120 cm

Speed, $N = 700$ r.p.m.

$$\therefore \quad \omega = \frac{2\pi N}{60} = \frac{2 \times \pi \times 700}{60} = 73.30 \text{ rad/s.}$$

If the tank is not closed at the top and also is very long, then the height of parabola corresponding to $\omega = 73.3$ will be

$$= \frac{\omega^2 \times R^2}{2 \times g} = \frac{73.3^2 \times 10^2}{2 \times 980} = 274.12 \text{ cm}$$

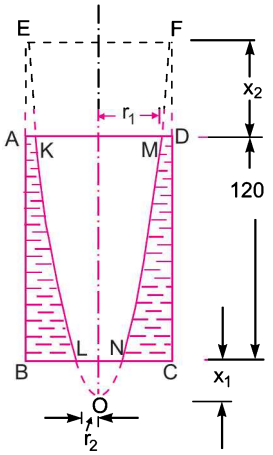


Fig. 5.24

From Fig. 5.24,

$$x_1 + 120 + x_2 = 274.12$$

or $$x_1 + x_2 = 274.12 - 120 = 154.12 \text{ cm} \quad ...(i)$$

From the parabola, *KOM*, we have

$$(120 + x_1) = \frac{\omega^2 r_1^2}{2g} = \frac{73.3^2 \times r_1^2}{2 \times 980} \quad ...(ii)$$

For the parabola, *LON*, we have

$$x_1 = \frac{\omega^2 r_2^2}{2g} = \frac{73.3^2 \times r_2^2}{2 \times 980} \quad ...(iii)$$

Now, volume of air before rotation = Volume of air after rotation

Volume of air before rotation $= \pi R^2 \times (120 - 80) = \pi \times 10^2 \times 40 = 12566.3 \text{ cm}^3$ ...(*iv*)

Volume of air after rotation = Volume of paraboloid *KOM* – volume of paraboloid *LON*

$$= \pi r_1^2 \times \frac{(120 + x_1)}{2} - \pi r_2^2 \times \frac{x_1}{2} \quad ...(v)$$

Equating (*iv*) and (*v*), we get

$$12566.3 = \frac{\pi r_1^2 (120 + x_1)}{2} - \frac{\pi r_2^2 \times x_1}{2} \quad ...(vi)$$

Substituting the value of r_1^2 from (*ii*) in (*vi*), we get

$$12566.3 = \pi \times \frac{(120 + x_1) \times 2 \times 980}{73.3^2} \times \frac{(120 + x_1)}{2} - \frac{\pi r_2^2 + x_1}{2}$$

$$\left\{ \because \text{ From } (ii),\ r_1^2 = \frac{2 \times 980 \times (120 + x_1)}{(73.3)^2} \right\}$$

or $$12566.3 = 0.573\ (120 + x_1)^2 - \frac{\pi r_2^2 \times x_1}{2}$$

Substituting the value of x_1 from (*iii*) in the above equation

$$12566.3 = 0.573 \left(120 + \frac{73.3^2 \times r_2^2}{2 \times 980} \right)^2 - \frac{\pi r_2^2}{2} \times \frac{73.3^2\, r_2^2}{2 \times 980}$$

$$= 0.573\ (120 + 2.74\ r_2^2)^2 - 4.3 \times r_2^2 \times r_2^2$$

$$= 0.573\ [120^2 + 2.74^2\ r_2^4 + 2 \times 120 \times 2.74\ r_2^2] - 4.3\ r_2^4$$

$$= 0.573\ [14400 + 7.506\ r_2^4 + 657.6\ r_2^2] - 4.3\ r_2^4$$

$$\frac{12566.3}{0.573} = 21930 = 14400 + 7.506\ r_2^4 + 657.6\ r_2^2 - 4.3\ r_2^4$$

or $$r_2^4\ (7.506 - 4.3) + 657.6\ r_2^2 + 14400 - 21930 = 0$$

or $$3.206\ r_2^4 + 657.6\ r_2^2 - 7530 = 0$$

$$\therefore \quad r_2^2 = \frac{-657.6 \pm \sqrt{657.6^2 - 4 \times (-7530) \times (3.206)}}{2 \times 3.206}$$

$$= \frac{-657.6 \pm \sqrt{432437.76 + 96564.72}}{6.412}$$

$$= \frac{-657.6 \pm 727.32}{6.412} = -215.98 \text{ or } 10.87$$

Negative value is not possible

$$\therefore \quad r_2^2 = 10.87 \text{ cm}^2$$

$\therefore$ Area uncovered at the base $= \pi r_2^2 = \pi \times 10.87 =$ **34.149 cm^2. Ans.**

Problem 5.29 *A closed cylindrical vessel of diameter 30 cm and height 100 cm contains water upto a depth of 80 cm. The air above the water surface is at a pressure of 5.886 N/cm^2. The vessel is rotated at a speed of 250 r.p.m. about its vertical axis. Find the pressure head at the bottom of the vessel : (a) at the centre, and (b) at the edge.*

Solution. Given :

Diameter of vessel $= 30$ cm

$\therefore$ Radius, $R = 15$ cm

Initial height of water, $H = 80$ cm

Length of cylinder, $L = 100$ cm

Pressure of air above water $= 5.886$ N/cm^2

or $$p = 5.886 \times 10^4 \frac{\text{N}}{\text{m}^2}$$

Head due to pressure, $h = p/\rho g$

$$= \frac{5.886 \times 10^4}{1000 \times 9.81} = 6 \text{ m of water}$$

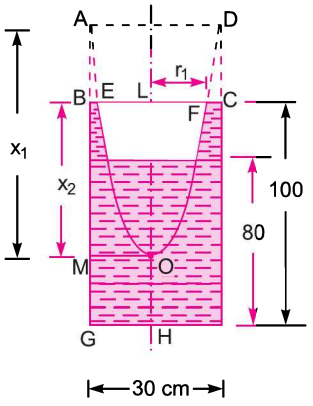


Fig. 5.25

Speed, $N = 250$ r.p.m.

$$\therefore \quad \omega = \frac{2\pi N}{60} = \frac{2\pi \times 250}{60} = 26.18 \text{ rad/s}$$

Let x_1 = Height of paraboloid formed, if the vessel is assumed open at the top and it is very long.

Then we have $$x_1 = \frac{\omega^2 R^2}{2g} = \frac{26.18^2 \times 15^2}{2 \times 981} = 78.60 \text{ cm} \qquad \ldots(i)$$

Let r_1 is the radius of the actual parabola of height x_2

Then $$x_2 = \frac{\omega^2 r_1^2}{2g} = \frac{26.18^2 \times r_1^2}{2 \times 981} = 0.35\ r_1^2 \qquad \ldots(ii)$$

The volume of air before rotation

$$= \pi R^2\ (100 - 80) = \pi \times 15^2 \times 20 = 14137\ \text{cm}^3$$

Volume of air after rotation = Volume of paraboloid EOF

$$= \frac{1}{2} \times \pi r_1^2 \times x_2$$

But volume of air before and after rotation is same.

$$\therefore \qquad 14137 = \frac{1}{2} \times \pi r_1^2 \times x_2$$

But from (*ii*), $$x_2 = 0.35\ r_1^2$$

$$\therefore \qquad 14137 = \frac{1}{2} \times \pi r_1^2 \times 0.35\ r_1^2$$

$$\therefore \qquad r_1^4 = \frac{2 \times 14137}{\pi \times 0.35} = 25714$$

$$r_1 = (25714)^{1/4} = 12.66\ \text{cm}$$

Substituting the value of r_1 in (*ii*), we get

$$x_2 = 0.35 \times 12.66^2 = 56.1\ \text{cm}$$

Pressure head at the bottom of the vessel

(*a*) At the centre. The pressure head at the centre, *i.e.*, at H = Pressure head due to air + OH

$$= 6.0 + (HL - LO) \qquad \{\because\ OH = LH - LO\}$$

$$= 6.0 + (1.0 - 0.561) \qquad \left\{\begin{array}{l}\because\ HL = 100\ \text{cm} = 1\ \text{m} \\ LO = x_2 = 56.1\ \text{cm} = .561\ \text{m}\end{array}\right\}$$

$$= \mathbf{6.439\ m\ of\ water.\ Ans.}$$

(*b*) At the edge, *i.e.*, at G = Pressure head due to air + height of water above G

$$= 6.0 + AG = 6.0 + (GM + MA) = 6.0 + (HO + x_1)$$

$$= 6.0 + HO + 0.786 \qquad \{\because\ x_1 = 78.6\ \text{cm} = 0.786\ \text{m}\}$$

$$= 6.0 + 0.439 + 0.786 \qquad \left\{\begin{array}{l}\because\ HO = LH - LO = 100 - 56.1 \\ \qquad = 43.9\ \text{cm} = 0.439\ \text{m}\end{array}\right\}$$

$$= \mathbf{7.225\ m\ of\ water.\ Ans.}$$

Problem 5.30 *A closed cylinder of radius R and height H is completely filled with water. It is rotated about its vertical axis with a speed of* ω *radians/s. Determine the total pressure exerted by water on the top and bottom of the cylinder.*

Solution. Given :

Radius of cylinder $= R$

Height of cylinder $= H$

Angular speed $= \omega$

As the cylinder is closed and completely filled with water, the rise of water level at the ends and depression of water at the centre due to rotation of the vessel, will be prevented. Thus the water will exert force on the complete top of the vessel. Also the pressure will be exerted at the bottom of the cylinder.

Total Pressure exerted on the top of cylinder. The top of cylinder is in contact with water and is in horizontal plane. The pressure variation at any radius in horizontal plane is given by equation (5.21)

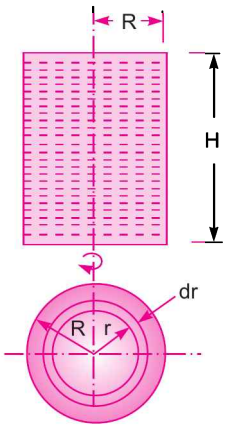


Fig. 5.26

or $$\frac{\partial p}{\partial r} = \frac{\rho v^2}{r} = \frac{\rho \omega^2 r^2}{r} = \rho \omega^2 r \quad \{\because \ v = \omega \times r\}$$

Integrating, we get

$$\int dp = \int \rho \omega^2 r dr \quad \text{or} \quad p = \frac{\rho \omega^2 r^2}{2} = \frac{\rho}{2} \omega^2 r^2$$

Consider an elementary circular ring of radius r and width dr on the top of the cylinder as shown in Fig. 5.26.

Area of circular ring = $2\pi r dr$

$\therefore$ Force on the elementary ring = Intensity of pressure × Area of ring

$$= p \times 2\pi r dr$$

$$= \frac{\rho}{2} \omega^2 r^2 \times 2\pi r dr. \qquad \left\{\because \ p = \frac{\rho}{2} \omega^2 r^2\right\}$$

$\therefore$ Total force on the top of the cylinder is obtained by integrating the above equation between the limits 0 and R.

$\therefore$ Total force or $$F_T = \int_0^R \frac{\rho}{2} \omega^2 r^2 \times 2\pi r dr = \frac{\rho}{2} \omega^2 \times 2\pi \int_0^R r^3 dr$$

$$= \frac{\rho}{2} \omega^2 \times 2\pi \left[\frac{r^4}{4}\right]_0^R = \frac{\rho}{2} \omega^2 \times 2\pi \times \frac{R^4}{4}$$

$$= \frac{\rho \omega^2}{4} \times \pi R^4 \qquad \ldots(5.25)$$

Total pressure force on the bottom of cylinder, F_B

= Weight of water in cylinder + total force on the top of cylinder

$$= \rho g \times \pi R^2 \times H + \frac{\rho}{4} \omega^2 \times \pi R^4 = \rho g \times \pi R^2 \times H + F_T \qquad \ldots(5.26)$$

ρ = Density of water.

Problem 5.31 *A closed cylinder of diameter 200 mm and height 150 mm is completely filled with water. Calculate the total pressure force exerted by water on the top and bottom of the cylinder, if it is rotated about its vertical axis at 200 r.p.m.*

Solution. Given :

Dia. of cylinder = 200 mm = 0.20 m

Radius, R = 0.1 m

Height of cylinder, H = 150 mm = 0.15 m

Speed, N = 200 r.p.m.

$\therefore$ Angular speed, $$\omega = \frac{2\pi N}{60} = \frac{2\pi \times 200}{60} = 20.94 \text{ rad/s}$$

Total pressure force on the top of the cylinder is given by equation (5.25)

$$F_T = \frac{\rho}{4} \times \omega^2 \times \pi \times R^4 = \frac{1000}{4} \times 20.94^2 \times \pi \times (0.1)^4 = \mathbf{34.44\ N.\ Ans.}$$

Now total pressure force on the bottom of the cylinder is given by equation (5.26) as

$$\begin{aligned} F_B &= \rho g \times \pi R^2 \times H + F_T \\ &= 1000 \times 9.81 \times \pi \times (0.1)^2 \times 0.15 + 34.44 \\ &= 46.22 + 34.44 = \mathbf{80.66\ N.\ Ans.} \end{aligned}$$

5.10.6 Equation of Free Vortex Flow. For the free vortex, from equation (5.20), we have

$$v \times r = \text{Constant} = \text{say } c$$

or $$v = \frac{c}{r}$$

Substituting the value of v in equation (5.23), we get

$$dp = \rho \frac{v^2}{r} dr - \rho g\, dz = \rho \times \frac{c^2}{r^2 \times r} dr - \rho g\, dz = \rho \times \frac{c^2}{r^3} dr - \rho g\, dz$$

Consider two points 1 and 2 in the fluid having radius r_1 and r_2 from the central axis respectively as shown in Fig. 5.27. The heights of the points from bottom of the vessel is z_1 and z_2.

Integrating the above equation for the points 1 and 2, we get

$$\int_1^2 dp = \int_1^2 \frac{\rho c^2}{r^3} dr - \int_1^2 \rho g dz$$

or $$p_2 - p_1 = \rho c^2 \int_1^2 r^{-3}\, dr - \rho g \int_1^2 dz$$

$$= \rho c^2 \left[\frac{r^{-3+1}}{-2}\right]_1^2 - \rho g\,[z_2 - z_1] = \frac{\rho c^2}{-2}\,[r_2^{-2} - r_1^{-2}] - \rho g\,[z_2 - z_1]$$

$$= -\frac{\rho c^2}{2}\left[\frac{1}{r_2^2} - \frac{1}{r_1^2}\right] - \rho g[z_2 - z_1] = -\frac{\rho}{2}\left[\frac{c^2}{r_2^2} - \frac{c^2}{r_1^2}\right] - \rho g[z_2 - z_1]$$

$$= -\frac{\rho}{2}\left[v_2^2 - v_1^2\right] - \rho g[z_2 - z_1] \qquad \left\{\because\ v_2 = \frac{c}{r_2}, v_1 = \frac{c}{r_1}\right\}$$

$$= \frac{\rho}{2}\left[v_1^2 - v_2^2\right] - \rho g\,[z_2 - z_1]$$

Dividing by ρg, we get

$$\frac{p_2 - p_1}{\rho g} = \frac{v_1^2 - v_2^2}{2g} - [z_2 - z_1]$$

or $$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2 \qquad \text{...(5.27)}$$

Equation (5.27) is Bernoulli's equation. Hence in case of free vortex flow, Bernoulli's equation is applicable.

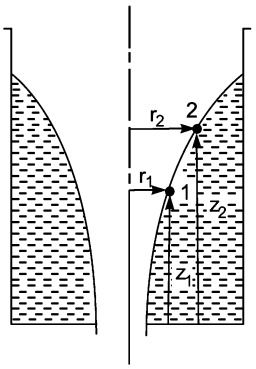


Fig. 5.27

Problem 5.32 *In a free cylindrical vortex flow, at a point in the fluid at a radius of 200 mm and at a height of 100 mm, the velocity and pressures are 10 m/s and 117.72 kN/m^2 absolute. Find the pressure at a radius of 400 mm and at a height of 200 mm. The fluid is air having density equal to 1.24 kg/m^3.*

Solution. At Point 1 : Given :

Radius, $r_1 = 200 \text{ mm} = 0.20 \text{ m}$

Height, $z_1 = 100 \text{ mm} = 0.10 \text{ m}$

Velocity, $v_1 = 10 \text{ m/s}$

Pressure, $p_1 = 117.72 \text{ kN/m}^2 = 117.72 \times 10^3 \text{ N/m}^2$

At Point 2 : $r_2 = 400 \text{ mm} = 0.4 \text{ m}$

$z_2 = 200 \text{ mm} = 0.2 \text{ m}$

$p_2 =$ pressure at point 2

$\rho = 1.24 \text{ kg/m}^3$

For the free vortex from equation (5.20), we have

$$v \times r = \text{constant or } v_1 r_1 = v_2 r_2$$

$$v_2 = \frac{v_1 \times r_1}{r_2} = \frac{10 \times 0.2}{0.4} = 5 \text{ m/s}$$

Now using equation (5.27), we get

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2$$

But $\rho = 1.24 \text{ kg/m}^3$

$$\therefore \quad \frac{117.72 \times 10^3}{1.24 \times 9.81} + \frac{10^2}{2 \times 9.81} + 0.1 = \frac{p_2}{\rho g} + \frac{5^2}{2 \times 9.81} + 0.2$$

or

$$\frac{p_2}{\rho g} = \frac{117.72 \times 10^3}{1.24 \times 9.81} + \frac{10^2}{2 \times 9.81} + 0.1 - \frac{5^2}{2 \times 9.81} - 0.2$$

$$= 9677.4 + 5.096 + 0.1 - 1.274 - 0.2 = 9676.22$$

$$p_2 = 9676.22 \times \rho g = 9676.22 \times 1.24 \times 9.81$$

$$= 117705 \text{ N/m}^2 = 117.705 \times 10^3 \text{ N/m}^2$$

$$= 117.705 \text{ kN/m}^2 \text{ (abs.)} = \mathbf{117.705 \text{ kN/m}^2}. \textbf{ Ans.}$$

(B) IDEAL FLOW (POTENTIAL FLOW)

▶ 5.11 INTRODUCTION

Ideal fluid is a fluid which is incompressible and inviscid. Incompressible fluid is a fluid for which density (ρ)remains constant. Inviscid fluid is a fluid for which viscosity (μ) is zero. Hence a fluid for which density is constant and viscosity is zero, is known as an ideal fluid.

The shear stress is given by, $\tau = \mu \frac{du}{dy}$. Hence for ideal fluid the shear stress will be zero as $\mu = 0$ for ideal fluid. Also the shear force (which is equal to shear stress multiplied by area) will be zero in

case of ideal or potential flow. The ideal fluids will be moving with uniform velocity. All the fluid particles will be moving with the same velocity.

The concept of ideal fluid simplifies the typical mathematical analysis. Fluids such as water and air have low viscosity. Also when the speed of air is appreciably lower than that of sound in it, the compressibility is so low that air is assumed to be incompressible. Hence under certain conditions, certain real fluids such as water and air may be treated like ideal fluids.

▶ 5.12 IMPORTANT CASES OF POTENTIAL FLOW

The following are the important cases of potential flow :

(*i*) Uniform flow, (*ii*) Source flow,
(*iii*) Sink flow, (*iv*) Free-vortex flow,
(*v*) Superimposed flow.

▶ 5.13 UNIFORM FLOW

In a uniform flow, the velocity remains constant. All the fluid particles are moving with the same velocity. The uniform flow may be :

(*i*) Parallel to *x*-axis (*ii*) Parallel to *y*-axis.

5.13.1 Uniform Flow Parallel to x-Axis. Fig. 5.27 (*a*) shows the uniform flow parallel to *x*-axis. In a uniform flow, the velocity remains constant. All the fluid particles are moving with the same velocity.

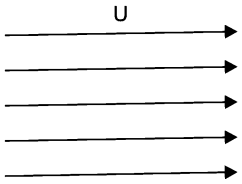


Fig. 5.27 (*a*)

Let U = Velocity which is uniform or constant along x-axis

u and v = Components of uniform velocity U along x and y-axis.

For the uniform flow, parallel to x-axis, the velocity components u and v are given as

$$u = U \text{ and } v = 0 \qquad \ldots(5.28)$$

But the velocity u in terms of stream function is given by,

$$u = \frac{\partial \psi}{\partial y}$$

and in terms of velocity potential the velocity u is given by,

$$u = \frac{\partial \phi}{\partial x}$$

$$\therefore \qquad u = \frac{\partial \psi}{\partial y} = \frac{\partial \phi}{\partial x} \qquad \ldots(5.29)$$

Similarly, it can be shown that $v = -\frac{\partial \psi}{\partial x} = \frac{\partial \phi}{\partial y}$ $\qquad \ldots(5.29A)$

But $u = U$ from equation (5.28). Substituting $u = U$ in equation (5.29), we have

$$\therefore \qquad U = \frac{\partial \psi}{\partial y} = \frac{\partial \phi}{\partial x} \qquad \ldots(5.30)$$

or $$U = \frac{\partial \psi}{\partial y} \text{ and also } U = \frac{\partial \phi}{\partial x}$$

First part gives $d\psi = U\, dy$ whereas second part gives $d\phi = U\, dx$.

Integration of these parts gives as

$$\psi = Uy + C_1 \text{ and } \phi = Ux + C_2$$

where C_1 and C_2 are constant of integration.

Now let us plot the stream lines and potential lines for uniform flow parallel to x-axis.

Plotting of Stream lines. For stream lines, the equation is

$$\psi = U \times y + C_1$$

Let $\psi = 0$, where $y = 0$. Substituting these values in the above equation, we get

$$0 = U \times 0 + C_1 \text{ or } C_1 = 0$$

Hence the equation of stream lines becomes as

$$\psi = U.\, y \qquad \ldots(5.31)$$

The stream lines are straight lines parallel to x-axis and at a distance y from the x-axis as shown in Fig. 5.28. In equation (5.31), $U.\, y$ represents the volume flow rate (*i.e.*, m^3/s) between x-axis and that stream line at a distance y.

Note. The thickness of the fluid stream perpendicular to the plane is assumed to be unity. Then $y \times 1$ or y represents the area of flow. And $U\, .\, y$ represents the product of velocity and area. Hence $U.\, y$ represents the volume flow rate.

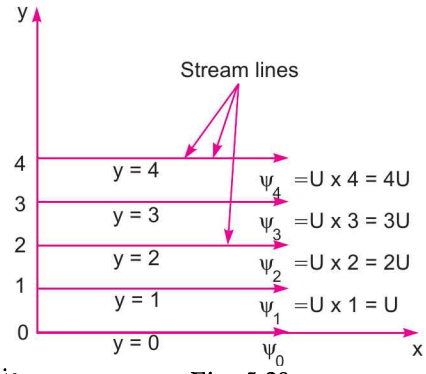


Fig. 5.28

Plotting of potential lines. For potential lines, the equation is

$$\phi = U\, .\, x + C_2 \qquad \ldots(5.32)$$

Let $\phi = 0$, where $x = 0$. Substituting these values in the above equation, we get $C_2 = 0$.

Hence equation of potential lines becomes as

$$\phi = U.\, x$$

The above equation shows that potential lines are straight lines parallel to y-axis and at a distance of x from y-axis as shown in Fig. 5.29.

Fig. 5.30 shows the plot of stream lines and potential lines for uniform flow parallel to x-axis. The stream lines and potential lines intersect each other at right angles.

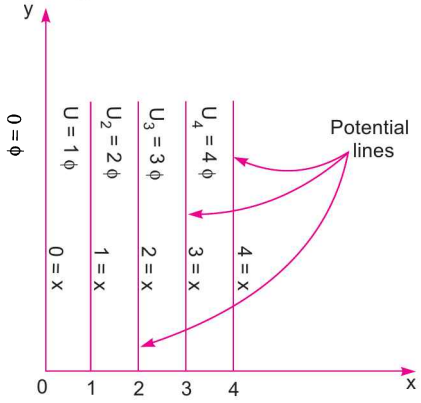


Fig. 5.29

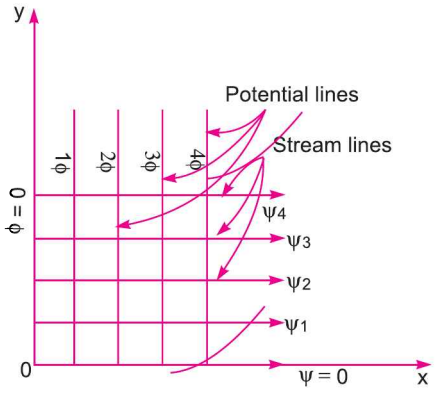


Fig. 5.30

5.13.2 Uniform Potential Flow Parallel to y-Axis. Fig. 5.31 shows the uniform potential flow parallel to y-axis in which U is the uniform velocity along y-axis.

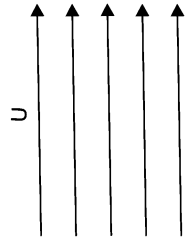


Fig. 5.31

The velocity components u, v along x-axis and y-axis are given by

$$u = 0 \text{ and } v = U \qquad \ldots(5.33)$$

These velocity components in terms of stream function (ψ) and velocity potential function (ϕ) are given as

$$u = \frac{\partial \psi}{\partial y} = \frac{\partial \phi}{\partial x} \qquad \ldots(5.34)$$

and

$$v = -\frac{\partial \psi}{\partial x} = \frac{\partial \phi}{\partial y} \qquad \ldots(5.35)$$

But from equation (5.33), $v = U$. Substituting $v = U$ in equation (5.35), we get

$$U = -\frac{\partial \psi}{\partial x} = \frac{\partial \phi}{\partial y} \quad \text{or} \quad U = -\frac{\partial \psi}{\partial x} \text{ and also } U = \frac{\partial \phi}{\partial y}$$

First part gives $d\psi = -U\,dx$ whereas second part gives $d\phi = U\,dy$.

Integration of these parts gives as

$$\psi = -U \,.\, x + C_1 \text{ and } \phi = U.y + C_2 \qquad \ldots(5.36)$$

where C_1 and C_2 are constant of integration. Let us now plot the stream lines and potential lines.

Plotting of Stream lines. For stream lines, the equation is $\psi = U.x + C_1$

Let $\psi = 0$, where $x = 0$. Then $C_1 = 0$.

Hence the equation of stream lines becomes as $\psi = -U.x$ $\qquad \ldots(5.37)$

The above equation shows that stream lines are straight lines parallel to y-axis and at a distance of x from the y-axis as shown in Fig. 5.32. The –ve sign shows that the stream lines are in the downward direction.

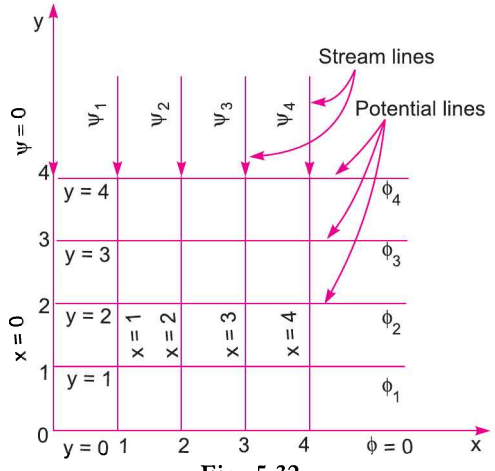


Fig. 5.32

Plotting of Potential lines. For potential lines, the equation is $\phi = U.y + C_2$

Let $\phi = 0$, where $y = 0$. Then $C_2 = 0$.

Hence equation of potential lines becomes as $\phi = U.y$ ...(5.38)

The above equation shows that potential lines are straight lines parallel to x-axis and at a distance of y from the x-axis as shown in Fig. 5.32.

▶ 5.14 SOURCE FLOW

The source flow is the flow coming from a point (source) and moving out radially in all directions of a plane at uniform rate. Fig. 5.33 shows a source flow in which the point O is the source from which the fluid moves radially outward. The strength of a source is defined as the volume flow rate per unit depth. The unit of strength of source is m^2/s. It is represented by q.

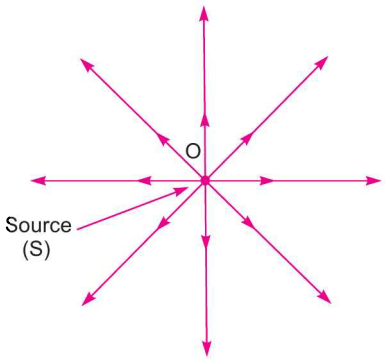


Fig. 5.33 *Source flow (Flow away from source)*

Let u_r = radial velocity of flow at a radius r from the source O

q = volume flow rate per unit depth

r = radius

The radial velocity u_r at any radius r is given by,

$$u_r = \frac{q}{2\pi r} \qquad ...(5.39)$$

The above equation shows that with the increase of r, the radial velocity decreases. And at a large distance away from the source, the velocity will be approximately equal to zero. The flow is in radial direction, hence the tangential velocity $u_\theta = 0$.

Let us now find the equation of stream function and velocity potential function for the source flow. As in this case, $u_\theta = 0$, the equation of stream function and velocity potential function will be obtained from u_r.

Equation of Stream Function

By definition, the radial velocity and tangential velocity components in terms of stream function are given by

$$u_r = \frac{1}{r}\frac{\partial \psi}{\partial \theta} \text{ and } u_\theta = -\frac{\partial \psi}{\partial r} \qquad \text{[See equation (5.12A)]}$$

But $$u_r = \frac{q}{2\pi r} \qquad \text{[See equation (5.39)]}$$

$\therefore$ $$\frac{1}{r}\frac{\partial \psi}{\partial \theta} = \frac{q}{2\pi r}$$

or $$d\psi = r.\frac{q}{2\pi r}.d\theta = \frac{q}{2\pi}\,d\theta$$

Integrating the above equation w.r.t. θ, we get

$$\psi = \frac{q}{2\pi} \times \theta + C_1, \text{ where } C_1 \text{ is constant of integration.}$$

Let $\psi = 0$, when $\theta = 0$, then $C_1 = 0$.

Hence the equation of stream function becomes as

$$\psi = \frac{q}{2\pi} . \theta \qquad \ldots(5.40)$$

In the above equation, q is constant.

The above equation shows that stream function is a function of θ. For a given value of θ, the stream function ψ will be constant. And this will be a radial line. The stream lines can be plotted by having different values of θ. Here θ is taken in radians.

Plotting of stream lines

When $\theta = 0, \psi = 0$

$$\theta = 45° = \frac{\pi}{4} \text{ radians}, \psi = \frac{q}{2\pi} . \frac{\pi}{4} = \frac{q}{8} \text{ units}$$

$$\theta = 90° = \frac{\pi}{2} \text{ radians}, \psi = \frac{q}{2\pi} . \frac{\pi}{2} = \frac{q}{4} \text{ units}$$

$$\theta = 135° = \frac{3\pi}{4} \text{ radians}, \psi = \frac{q}{2\pi} . \frac{3\pi}{4} = \frac{3q}{8} \text{ units}$$

The stream lines will be radial lines as shown in Fig. 5.34.

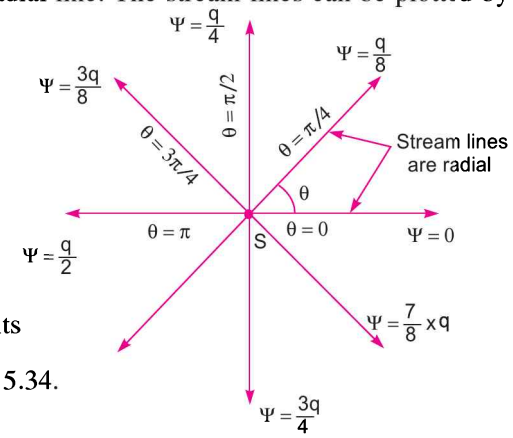


Fig. 5.34 *Stream line for source flow.*

Equation of Potential Function

By definition, the radial and tangential components in terms of velocity function are given by

$$u_r = \frac{\partial \phi}{\partial r} \text{ and } u_\theta = \frac{1}{r} . \frac{\partial \phi}{\partial \theta} \qquad \text{[See equation (5.9A)]}$$

But from equation (5.39), $u_r = \frac{q}{2\pi r}$

Equating the two values of u_r, we get

$$\frac{\partial \phi}{\partial r} = \frac{q}{2\pi r} \text{ or } d\phi = \frac{q}{2\pi r} dr$$

Integrating the above equation, we get

$$\int d\phi = \int \frac{q}{2\pi r} . dr$$

or

$$\phi = \frac{q}{2\pi} \int \frac{1}{r} dr \quad \left[\because \frac{q}{2\pi} \text{ is a constant term} \right]$$

$$= \frac{q}{2\pi} \log_e r \qquad \ldots(5.41)$$

In the above equation, q is constant.

The above equation shows, that the velocity potential function is a function of r. For a given value of r, the velocity function ϕ will be constant. Hence it will be a circle with origin at the source. The velocity potential lines will be circles with origin at the source as shown in Fig. 5.35.

Let us now find an expression for the pressure in terms of radius.

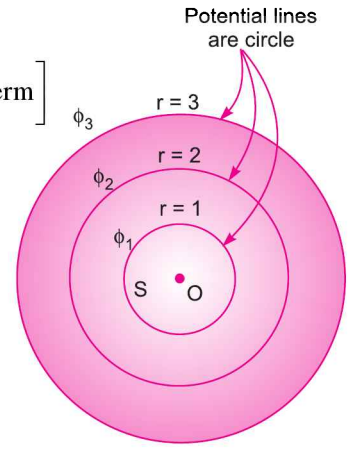


Fig. 5.35 *Potential lines for source.*

Pressure distribution in a plane source flow

The pressure distribution in a plane source flow can be obtained with the help of Bernoulli's equation. Let us assume that the plane of the flow is horizontal. In that case the datum head will be same for two points of flow.

Let p = pressure at a point 1 which is at a radius r from the source at point 1

u_r = velocity at point 1

p_0 = pressure at point 2, which is at a large distance away from the source. The velocity will be zero at point 2. [Refer to equation (5.39)]

Applying Bernoulli's equation, we get

$$\frac{p}{\rho g} + \frac{u_r^2}{2g} = \frac{p_0}{\rho g} + 0 \quad \text{or} \quad \frac{(p - p_0)}{\rho g} = -\frac{u_r^2}{2g}$$

or
$$(p - p_0) = -\frac{\rho . u_r^2}{2}$$

But from equation(5.39), $u_r = \dfrac{q}{2\pi r}$

Substituting the value of u_r in the above equation, we get

$$(p - p_0) = -\left(\frac{\rho}{2}\right) \cdot \left(\frac{q}{2\pi r}\right)^2$$

$$= -\frac{\rho q^2}{8\pi^2 r^2} \qquad \ldots(5.42)$$

In the above equation, ρ and q are constants.

The above equation shows that the pressure is inversely proportional to the square of the radius from the source.

▶ 5.15 SINK FLOW

The sink flow is the flow in which fluid moves radially inwards towards a point where it disappears at a constant rate. This flow is just opposite to the source flow. Fig. 5.36 shows a sink flow in which the fluid moves radially inwards towards point O, where it disappears at a constant rate. The pattern of stream lines and equipotential lines of a sink flow is the same as that of a source flow. All the equations derived for a source flow shall hold to good for sink flow also except that in sink flow equations, q is to be replaced by $(-q)$.

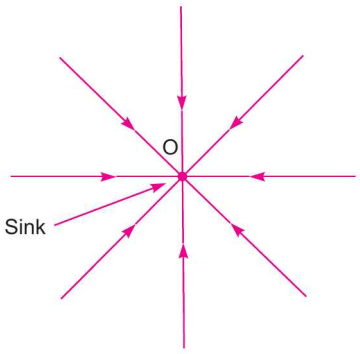


Fig. 5.36 *Sink flow (Flow toward centre)*

Problem 5.33 *Plot the stream lines for a uniform flow of : (i) 5 m/s parallel to the positive direction of the x-axis and (ii) 10 m/s parallel to the positive direction of the y-axis.*

Solution. (*i*) The stream function for a uniform flow parallel to the positive direction of the x-axis is given by equation (5.31) as

$$\psi = U \times y$$

The above equation shows that stream lines are straight lines parallel to the x-axis at a distance y from the x-axis. Here U = 5 m/s and hence above equation becomes as

$$\psi = 5y$$

For $y = 0$, stream function $\psi = 0$

For $y = 0.2$, stream function $\psi = 5 \times 0.2 = 1$ unit

For $y = 0.4$, stream function $\psi = 5 \times 0.4 = 2$ unit

The other values of stream function can be obtained by substituting the different values of y. The stream lines are horizontal as shown in Fig. 5.36 (*a*).

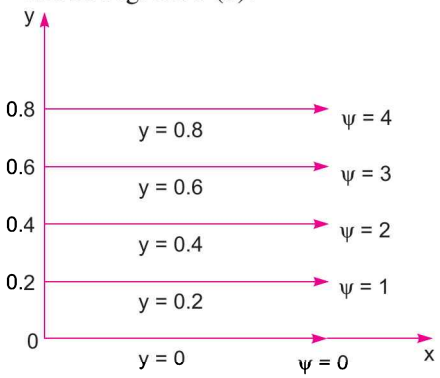


Fig. 5.36 (*a*)

(*ii*) The stream function for a uniform flow parallel to the positive direction of the y-axis is given by equation (5.37) as

$$\psi = - U \times x$$

The above equation shows that stream lines are straight lines parallel to the y-axis at a distance x from the y-axis. Here $U = 10$ m/s and hence the above equation becomes as

$$\psi = - 10 \times x$$

The negative sign shows that the stream lines are in the downward direction.

For $x = 0$, the stream function $\psi = 0$

For $x = 0.1$, the stream function $\psi = - 10 \times 0.1 = - 1.0$ unit

For $x = 0.2$, the stream function $\psi = - 10 \times 0.2 = - 2.0$ unit

For $x = 0.3$, the stream function $\psi = - 10 \times 0.3 = - 3.0$ unit

The other values of stream function can be obtained by substituting the different values of x. The stream lines are vertical as shown in Fig. 5.36 (*b*).

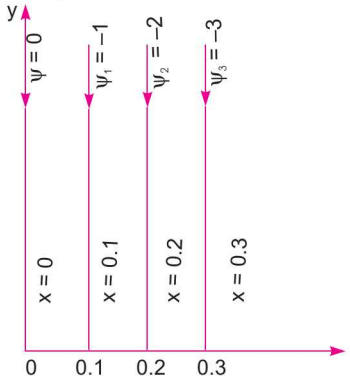


Fig. 5.36 (*b*)

Problem 5.34 *Determine the velocity of flow at radii of 0.2 m, 0.4 m and 0.8 m, when the water is flowing radially outward in a horizontal plane from a source at a strength of 12 m^2/s.*

Solution. Given :

Strength of source, $q = 12\ m^2/s$

The radial velocity u_r at any radius r is given by equation (5.39) as

$$u_r = \frac{q}{2\pi r}$$

When $r = 0.2$ m, $\quad u_r = \dfrac{12}{2\pi \times 0.2} =$ **9.55 m/s. Ans.**

When $r = 0.4$ m, $\quad u_r = \dfrac{12}{2\pi \times 0.4} =$ **4.77 m/s. Ans.**

When $r = 0.8$ m, $\quad u_r = \dfrac{12}{2\pi \times 0.8} =$ **2.38 m/s. Ans.**

Problem 5.35 *Two discs are placed in a horizontal plane, one over the other. The water enters at the centre of the lower disc and flows radially outward from a source of strength 0.628 m^2/s. The pressure, at a radius 50 mm, is 200 kN/m^2. Find :*

(i) pressure in kN/m^2 at a radius of 500 mm and

(ii) stream function at angles of 30° and 60° if $\psi = 0$ at $\theta = 0°$.

Solution. Given :

Source strength, $\quad q = 0.628\ m^2/s$

Pressure at radius 50 mm, $p_1 = 200\ kN/m^2 = 200 \times 10^3\ N/m^2$

(i) Pressure at a radius 500 mm

Let $\quad p_2$ = pressure at radius 500 mm

$(u_r)_1$ = velocity at radius 50 mm

$(u_r)_2$ = velocity at radius 500 mm

The radial velocity at any radius r is given by equation (5.39) as

$$u_r = \frac{q}{2\pi r}$$

When $r = 50$ mm $= 0.05$ m, $\quad (u_r)_1 = \dfrac{0.628}{2\pi \times 0.05} = 1.998$ m/s $\simeq 2$ m/s

When $r = 500$ mm $= 0.5$ m, $\quad (u_r)_2 = \dfrac{0.628}{2\pi \times 0.5} = 0.2$ m/s

Applying Bernoulli's equation at radius 0.05 m and at radius 0.5 m,

$$\frac{p_1}{\rho g} + \frac{(u_r)_1^2}{2g} = \frac{p_2}{\rho g} + \frac{(u_r)_2^2}{2g}$$

or

$$\frac{p_1}{\rho} + \frac{(u_r)_1^2}{2} = \frac{p_2}{\rho} + \frac{(u_r)_2^2}{2}$$

or $$\frac{200\times10^3}{1000}+\frac{2^2}{2}=\frac{p_2}{1000}+\frac{0.2^2}{2}$$

or $$200+2=\frac{p_2}{1000}+0.02$$

or $$\frac{p_2}{1000}=202-0.02=201.98$$

$\therefore$ $$p_2=201.98\times1000\ \text{N/m}^2=\mathbf{201.98\ kN/m^2.\ Ans.}$$

(*ii*) *Stream functions at $\theta = 30°$ and $\theta = 60°$*

For the source flow, the equation of stream function is given by equation (5.40) as

$$\psi=\frac{q}{2\pi}.\theta,\ \text{where }\theta\text{ is in radians}$$

When $\theta = 30°$, $$\psi=\frac{0.628}{2\pi}\times\frac{30\times\pi}{180}\qquad\left(\because\ \theta=30°=\frac{30\times\pi}{180}\text{ radians}\right)$$

$$=\mathbf{0.0523\ m^2/s.\ Ans.}$$

When $\theta = 60°$, $$\psi=\frac{0.628}{2\pi}\times\frac{60\pi}{180}=\mathbf{0.1046\ m^2/s.\ Ans.}$$

▶ 5.16 FREE-VORTEX FLOW

Free-vortex flow is a circulatory flow of a fluid such that its stream lines are concentric circles.

For a free-vortex flow, $u_\theta \times r$ = constant (say C)

Also, circulation around a stream line of an irrotation vortex is

$$\Gamma=2\pi r\times u_\theta=2\pi\times C\qquad(\because\ r\times u_\theta=C)$$

where u_θ = tangential velocity at any radius r from the centre.

$\therefore$ $$u_\theta=\frac{\Gamma}{2\pi r}$$

The circulation Γ is taken positive if the free vortex is anticlockwise.

For a free-vortex flow, the velocity components are

$$u_\theta=\frac{\Gamma}{2\pi r}\quad\text{and}\quad u_r=0$$

Equation of Stream Function

By definition, the stream function is given by

$$u_\theta=\frac{-\partial\psi}{\partial r}\quad\text{and}\quad u_r=\frac{1}{r}\frac{\partial\psi}{\partial\theta}\qquad\text{[See equation (5.12}A\text{)]}$$

In case of free-vortex flow, the radial velocity (u_r) is zero. Hence equation of stream function will be obtained from tangential velocity, u_θ. The value of u_θ is given by

$$u_\theta=\frac{\Gamma}{2\pi r}$$

Equating the two values of u_θ, we get

$$-\frac{\partial \psi}{\partial r} = \frac{\Gamma}{2\pi r} \quad \text{or} \quad d\psi = -\frac{\Gamma}{2\pi r} dr$$

Integrating the above equation, we get

$$\int d\psi = \int -\frac{\Gamma}{2\pi r} dr = \left(-\frac{\Gamma}{2\pi}\right) \int \frac{1}{r} dr$$

or
$$\psi = \left(-\frac{\Gamma}{2\pi}\right) \log_e r \qquad \left(\because \frac{\Gamma}{2\pi} \text{ is a constant term}\right) \quad \ldots(5.43)$$

The above equation shows that stream function is a function of radius. For a given value of r, the stream function is constant. Hence the stream lines are concentric circles as shown in Fig. 5.37.

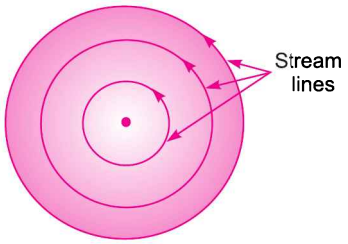


Fig. 5.37

Equation of potential function. By definition, the potential function is given by,

$$u_\theta = \frac{1}{r}\frac{\partial \phi}{\partial \theta} \quad \text{and} \quad u_r = \frac{\partial \phi}{\partial r} \quad \text{[See equation (5.9}A\text{)]}$$

Here $u_r = 0$ and $u_\theta = \dfrac{\Gamma}{2\pi r}$. Hence, the equation of potential function will be obtained from u_θ.

Equating the two values of u_θ, we get

$$\frac{1}{r}\frac{\partial \phi}{\partial \theta} = \frac{\Gamma}{2\pi r} \quad \text{or} \quad d\phi = r.\frac{\Gamma}{2\pi r}.d\theta = \frac{\Gamma}{2\pi} d\theta$$

Integrating the above equation, we get

$$\int d\phi = \int \frac{\Gamma}{2\pi} d\theta \quad \text{or} \quad \phi = \frac{\Gamma}{2\pi}\int d\theta = \frac{\Gamma}{2\pi}.\theta \qquad \ldots(5.44)$$

The above equation shows that velocity potential function is a function of θ. For a given value of θ, potential function is a constant. Hence equipotential lines are radial as shown in Fig. 5.38.

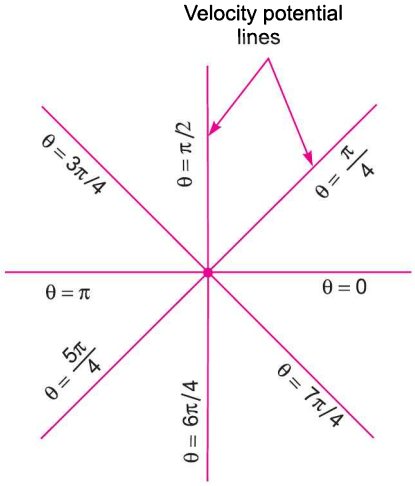


Fig. 5.38 *Potential lines are radial.*

▶ 5.17 SUPER-IMPOSED FLOW

The flow patterns due to uniform flow, a source flow, a sink flow and a free vortex flow can be super-imposed in any linear combination to get a resultant flow which closely resembles the flow around bodies. The resultant flow will still be potential and ideal. The following are the important super-imposed flow :

(*i*) Source and sink pair
(*ii*) Doublet (special case of source and sink combination)
(*iii*) A plane source in a uniform flow (flow past a half body)
(*iv*) A source and sink pair in a uniform flow
(*v*) A doublet in a uniform flow.

5.17.1 Source and Sink Pair. Fig. 5.39 shows a source and a sink of strength q and $(-q)$ placed at A and B respectively at equal distance from the point O on the x-axis. Thus the source and sink are placed symmetrically on the x-axis. The source of strength q is placed at A and sink of strength $(-q)$ is placed at B. The combination of the source and the sink would result in a flownet where stream lines will be circular arcs starting from point A and ending at point B as shown in Fig. 5.40.

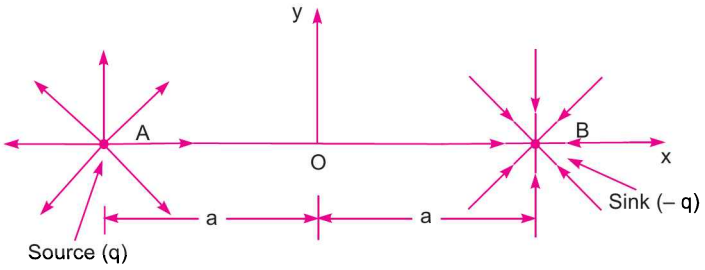


Fig. 5.39 *Source and sink pair.*

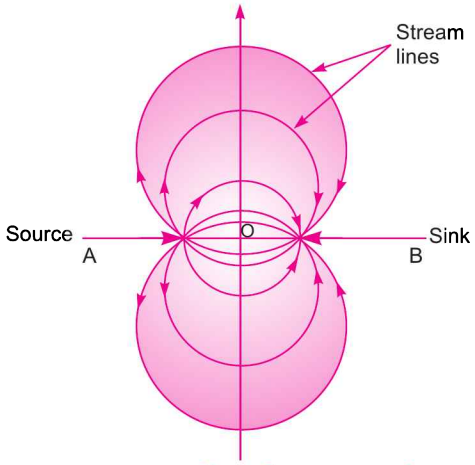


Fig. 5.40 *Stream lines for source-sink pair.*

Equation of stream function and potential function

Let P be any point in the resultant flownet of source and sink as shown in Fig. 5.41.

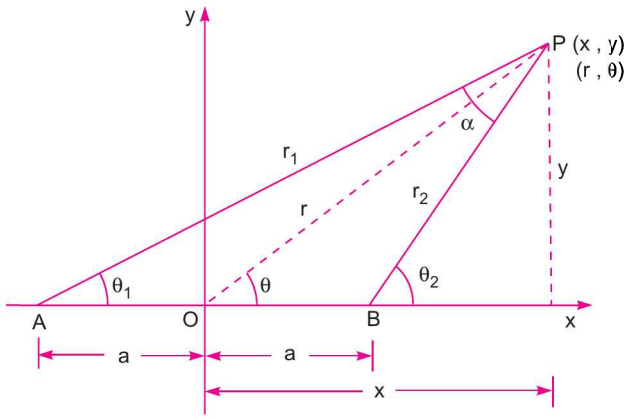


Fig. 5.41

Let r, θ = Cylindrical co-ordinates of point P with respect to origin O

x, y = Corresponding co-ordinates of point P

r_1, θ_1 = Position of point P with respect to source placed at A

r_2, θ_2 = Position of point P with respect to sink placed at B

α = Angle subtended at P by the join of source and sink *i.e.*, angle APB.

Let us find the equation for the resultant stream function and velocity potential function. The equation for stream function due to source is given by equation (5.40) as $\psi_1 = \dfrac{q \cdot \theta_1}{2\pi}$ whereas due to sink it is given by $\psi_2 = \dfrac{(-q\theta_2)}{2\pi}$. The equation for resultant stream function (ψ) will be the sum of these two stream function.

$\therefore$ $$\psi = \psi_1 + \psi_2$$

$$= \frac{q\theta_1}{2\pi} + \left(\frac{-q\theta_2}{2\pi}\right) = \frac{-q}{2\pi}(\theta_2 - \theta_1)$$

$$= \frac{-q}{2\pi} \cdot \alpha \quad [\because \quad \alpha = \theta_2 - \theta_1. \text{ In triangle } ABP, \theta_1 + \alpha + (180^\circ - \theta_2)$$

$$= 180^\circ \quad \therefore \quad \alpha = \theta_2 - \theta_1]$$

$$= \frac{-q \cdot \alpha}{2\pi} \qquad \ldots(5.45)$$

The equation for potential function due to source is given by equation (5.41) as $\phi_1 = \dfrac{q}{2\pi} \log_e r_1$ and due to sink it is given as $\phi_2 = \dfrac{-q}{2\pi} \log_e r_2$. The equation for resultant potential function (ϕ) will be the sum of these two potential function.

$\therefore$ $$\phi = \phi_1 + \phi_2$$

$$= \frac{q}{2\pi} \log_e r_1 + \left(\frac{-q}{2\pi}\right) \log_e r_2$$

$$= \frac{q}{2\pi} [\log_e r_1 - \log_e r_2] = \frac{q}{2\pi} \log_e \left(\frac{r_1}{r_2}\right) \qquad ...(5.46)$$

To prove that resultant stream lines will be circular arc passing through source and sink

The resultant stream function is given by equation (5.45) as

$$\psi = \frac{-q \cdot \alpha}{2\pi} \quad \text{or} \quad \frac{-q}{2\pi} (\theta_2 - \theta_1) \qquad (\because \alpha = \theta_2 - \theta_1)$$

For a given stream line ψ = constant. In the above equation the term $\frac{q}{2\pi}$ is also constant. This means that $(\theta_2 - \theta_1)$ or angle α will also be constant for various positions of P in the plane.

To satisfy this, the locus of P must be a circle with AB as chord, having its centre on y-axis, as shown in Fig. 5.40.

Consider the equation (5.45) again as

$$\psi = \frac{-q}{2\pi} \alpha = \frac{-q}{2\pi} (\theta_2 - \theta_1) \qquad (\because \alpha = \theta_2 - \theta_1)$$

$$= \frac{q}{2\pi} (\theta_1 - \theta_2)$$

or $$(\theta_1 - \theta_2) = \frac{2\pi\psi}{q}$$

Taking tangent to both sides, we get

$$\tan (\theta_1 - \theta_2) = \tan \left(\frac{2\pi\psi}{q}\right) \quad \text{or} \quad \frac{\tan \theta_1 - \tan \theta_2}{1 + \tan \theta_1 . \tan \theta_2} = \tan \left(\frac{2\pi\psi}{q}\right) \qquad ...(i)$$

But $$\tan \theta_1 = \frac{y}{x + a} \quad \text{and} \quad \tan \theta_2 = \frac{y}{x - a} \qquad ...(5.46A)$$

Substituting the values of $\tan \theta_1$ and $\tan \theta_2$ in equation (i),

$$\frac{\frac{y}{(x+a)} - \frac{y}{(x-a)}}{1 + \frac{y}{(x+a)} \cdot \frac{y}{(x-a)}} = \tan \left(\frac{2\pi\psi}{q}\right)$$

or $$\frac{y(x-a) - y(x+a)}{x^2 - a^2 + y^2} = \tan \left(\frac{2\pi\psi}{q}\right)$$

or $$\frac{-2ay}{x^2 - a^2 + y^2} = \tan \left(\frac{2\pi\psi}{q}\right)$$

or $$\frac{-2ay}{x^2 - a^2 + y^2} = \frac{1}{\cot \left(\frac{2\pi\psi}{q}\right)}$$

or $$x^2 - a^2 + y^2 = -2ay \cot \left(\frac{2\pi\psi}{q}\right)$$

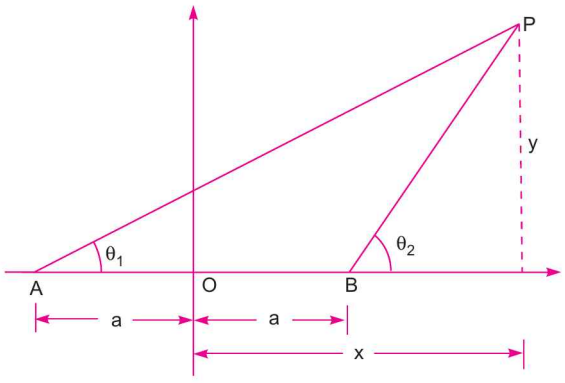


Fig. 5.41 (*a*)

or $$x^2 - a^2 + y^2 + 2ay \cot\left(\frac{2\pi\psi}{q}\right) = 0$$

or $$x^2 + y^2 + 2ay \cot\left(\frac{2\pi\psi}{q}\right) - a^2 = 0$$

or $$x^2 + y^2 + 2ay \cot\left(\frac{2\pi\psi}{q}\right) + a^2 \cot^2\left(\frac{2\pi\psi}{q}\right) - a^2 \cot^2\left(\frac{2\pi\psi}{q}\right) - a^2 = 0$$

$$\left[\text{Adding and subtracting } a^2 \cot^2\left(\frac{2\pi\psi}{q}\right)\right]$$

or $$x^2 + \left[y + a \cot\left(\frac{2\pi\psi}{q}\right)\right]^2 = a^2 + a^2 \cot^2\left(\frac{2\pi\psi}{q}\right)$$

$$= a^2\left[1 + \cot^2\left(\frac{2\pi\psi}{q}\right)\right]$$

$$= a^2 \operatorname{cosec}^2\left(\frac{2\pi\psi}{q}\right) \qquad \left[\because\ 1 + \cot^2\left(\frac{2\pi\psi}{q}\right) = \operatorname{cosec}^2\left(\frac{2\pi\psi}{q}\right)\right]$$

or $$x^2 + \left[y + a \cot\left(\frac{2\pi\psi}{q}\right)\right]^2 = \left[a \operatorname{cosec}\left(\frac{2\pi\psi}{q}\right)\right]^2 \qquad \ldots(5.47)$$

The above is the equation of a circle* with centre on y-axis at a distance of $\pm a \cot\left(\frac{2\pi\psi}{q}\right)$ from the origin. The radius of the circle will be $a \operatorname{cosec}\left(\frac{2\pi\psi}{q}\right)$.

Similarly, it can be shown that the potential lines for the source-sink pair will be eccentric non-intersecting circles with their centres on the x-axis as shown in Fig. 5.41 (b).

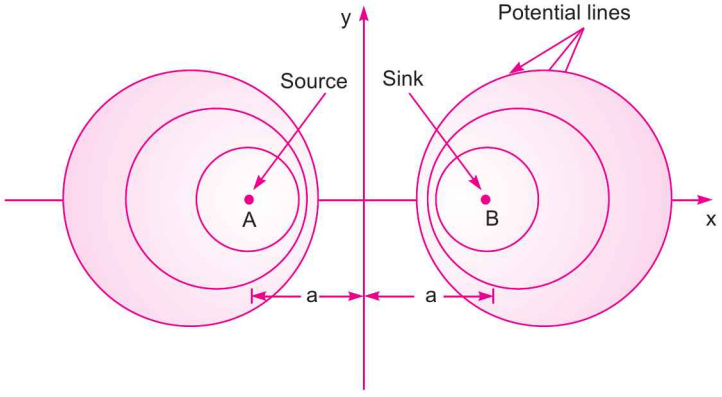


Fig. 5.41 (b) *Potential lines for source sink pair (Potential lines are eccentric non-intersecting circles with their centres on x-axis).*

*The equation $x^2 + y^2 = a^2$ is the equation of a circle with centre at origin and of radius 'a'.

Problem 5.36 *A source and a sink of strength 4 m^2/s and 8 m^2/s are located at (– 1, 0) and (1, 0) respectively. Determine the velocity and stream function at a point P (1, 1) which is lying on the flownet of the resultant stream line.*

Solution. Given :

Source strength, $q_1 = 4\ m^2/s$

Sink strength, $q_2 = 8\ m^2/s$

Distance of the source and sink from origin, $a = 1$ unit.

The position of the source, sink and point P in the flow field is shown in Fig. 5.42.

From Fig. 5.42, it is clear that angle θ_2 will be 90° and angle θ_1 can be calculated from right angled triangle ABP.

The equation for stream function due to source is given by equation (5.40) as $\psi_1 = \dfrac{q_1 \times \theta_1}{2\pi}$, whereas due to sink it is given by $\psi_2 = \dfrac{-q_2 \times \theta_2}{2\pi}$. The resultant stream function ψ is given as

$$\psi = \psi_1 + \psi_2$$

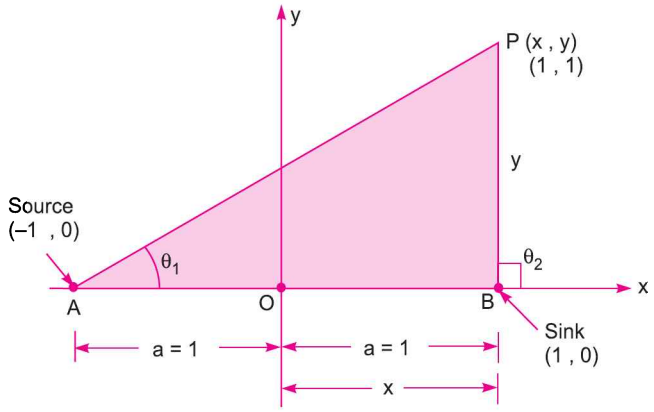


Fig. 5.42

$$= \frac{q_1 \times \theta_1}{2\pi} + \left(\frac{-q_2 \times \theta_2}{2\pi}\right) = \frac{q_1 \times \theta_1}{2\pi} - \frac{q_2 \times \theta_2}{2\pi} \qquad ...(i)$$

Let us find the values of θ_1 and θ_2 in radians. From the geometry, it is clear that the traingle ABP is a right angled triangle with angle $\theta_2 = 90° = \dfrac{90}{180} \times \pi = \dfrac{\pi}{2}$ radians.

Also $$\tan \theta_1 = \frac{BP}{AB} = \frac{1}{2} = 0.5$$

or $$\theta_1 = \tan^{-1} 0.5 = 26.56° = 26.56 \times \frac{\pi}{180} \text{ radians} = 0.463$$

Substituting these values in equation (i),

$$\psi = \frac{q_1}{2\pi} \times 0.463 - \frac{q_2}{2\pi} \times \frac{\pi}{2}$$

$$= \frac{\pi}{2\pi} \times 0.463 - \frac{8}{2\pi} \times \frac{\pi}{2} \qquad (\because\ q_1 = 4\ \text{m}^2/\text{s},\ q_2 = 8\ \text{m}^2/\text{s})$$

$$= 0.294 - 2.0 = \mathbf{-\ 1.706\ m^2/s.\ Ans.}$$

To find the velocity at the point P, let us first find the stream function in terms of x and y co-ordinates. The stream function in terms of θ_1 and θ_2 is given by equation (i) above as

$$\psi = \frac{q_1 \times \theta_1}{2\pi} - \frac{q_2 \times \theta_2}{2\pi}$$

The values of θ_1 and θ_2 in terms of x, y and a are given by equation (5.46A) as

$$\tan \theta_1 = \frac{y}{x + a} \quad \text{and} \quad \tan \theta_2 = \frac{y}{(x - a)}$$

or

$$\theta_1 = \tan^{-1} \frac{y}{x + a} \quad \text{and} \quad \theta_2 = \tan^{-1} \frac{y}{(x - a)}$$

Substituting these values of θ_1 and θ_2 in equation (i), we get

$$\psi = \frac{q_1}{2\pi} \tan^{-1} \frac{y}{x + a} - \frac{q_2}{2\pi} \tan^{-1} \frac{y}{x - a}$$

The velocity component $u = \dfrac{\partial \psi}{\partial y}$ and $v = -\dfrac{\partial \psi}{\partial x}$.

$\therefore$

$$u = \frac{\partial \psi}{\partial y}$$

$$= \frac{\partial}{\partial y}\left[\frac{q_1}{2\pi} \tan^{-1} \frac{y}{x + a} - \frac{q_2}{2\pi} \tan^{-1} \frac{y}{x - a}\right]$$

$$= \frac{q_1}{2\pi} \times \frac{1}{1 + \left(\dfrac{y}{x + a}\right)^2} \times \frac{1}{(x + a)} - \frac{q_2}{2\pi} \times \frac{1}{1 + \left(\dfrac{y}{x - a}\right)^2} \times \frac{1}{(x - a)}$$

$$= \frac{q_1}{2\pi} \frac{(x + a)^2}{(x + a)^2 + y^2} \times \frac{1}{(x + a)} - \frac{q_2}{2\pi} \times \frac{(x - a)^2}{(x - a)^2 + y^2} \times \frac{1}{(x - a)}$$

$$= \frac{q_1}{2\pi} \frac{(x + a)}{(x + a)^2 + y^2} - \frac{q_2}{2\pi} \frac{(x - a)}{(x - a)^2 + y^2}$$

At the point $P(1, 1)$, the component u is obtained by substituting $x = 1$ and $y = 1$ in the above equation. The value of a is also equal to one.

$\therefore$

$$u = \frac{q_1}{2\pi} \frac{1 + 1}{(1 + 1)^2 + 1^2} - \frac{q_2}{2\pi} \frac{(1 - 1)}{(1 - 1)^2 + 1^2}$$

$$= \frac{q_1}{2\pi} \frac{2}{5} - \frac{q_2}{2\pi} \times 0 = \frac{q_1}{2\pi} \times \frac{2}{5} = \frac{4}{2\pi} \times \frac{2}{5} = 0.2544\ \text{m/s}$$

Now
$$v = -\frac{\partial \psi}{\partial x}$$
$$= -\frac{\partial}{\partial x}\left[\frac{q_1}{2\pi}\tan^{-1}\frac{y}{x+a} - \frac{q_2}{2\pi}\tan^{-1}\frac{y}{x-a}\right]$$
$$= -\left[\frac{q_1}{2\pi}\frac{1}{1+\left(\frac{y}{x+a}\right)^2}\times\frac{y(-1)}{(x+a)^2}\times 1 - \frac{q_2}{2\pi}\times\frac{1}{1+\left(\frac{y}{x-a}\right)^2}\times\frac{y(-1)}{(x-a)^2}\times 1\right]$$
$$= -\left[\frac{q_1}{2\pi}\frac{(x+a)^2}{(x+a)^2+y^2}\times\frac{(-y)}{(x+a)^2} - \frac{q_2}{2\pi}\frac{(x-a)^2}{(x-a)^2+y^2}\times\frac{(-y)}{(x-a)^2}\right]$$
$$= \frac{q_1}{2\pi}\frac{y}{(x+a)^2+y^2} - \frac{q_2}{2\pi}\frac{y}{(x-a)^2+y^2}$$

At the point $P(1, 1)$,
$$v = \frac{q_1}{2\pi}\times\frac{1}{(1+1)^2+1^2} - \frac{q_2}{2\pi}\times\frac{1}{(1-1)^2+1^2} \qquad (\because a = 1)$$
$$= \frac{q_1}{2\pi}\times\frac{1}{5} - \frac{q_2}{2\pi}\times\frac{1}{1}$$
$$= \frac{q_1}{2\pi}\times\frac{1}{5} - \frac{q_2}{2\pi} = \frac{4}{2\pi}\times\frac{1}{5} - \frac{8}{2\pi} = 0.1272 - 1.272 = -1.145 \text{ m/s}^2$$

$\therefore$ The resultant velocity, $V = \sqrt{u^2+v^2} = \sqrt{0.2544^2 + (-1.145)^2}$ = **1.174 m/s. Ans.**

Problem 5.37 *For the above problem, determine the pressure at P(1, 1) if the pressure at infinity is zero and density of fluid is 1000 kg/m^3.*

Solution. Given :

Pressure at infinity, $p_0 = 0$

Density of fluid, $\rho = 1000$ kg/m^3

The velocity* of fluid at infinity will be zero. If V_0 = velocity at infinity, then $V_0 = 0$.

The resultant velocity of fluid at $P(1, 1) = 1.174$ m/s (calculated above)

or $V = 1.174$ m/s.

Let p = pressure at $P(1, 1)$

Applying Bernoulli's theorem at point at infinity and at point P, we get
$$\frac{p_0}{\rho g} + \frac{V_0^2}{2g} = \frac{p}{\rho g} + \frac{V^2}{2g}$$
or
$$0 + 0 = \frac{p}{\rho g} + \frac{V^2}{2g} \quad \text{or} \quad 0 = \frac{p}{\rho g} + \frac{V^2}{2g} \quad \text{or} \quad 0 = \frac{p}{\rho} + \frac{V^2}{2}$$
or
$$\frac{p}{\rho} = -\frac{V^2}{2} = -\frac{1.174^2}{2} \qquad (\because V = 1.174 \text{ m/s})$$

* From equation (5.39), the velocity at a distance 'r' from source or sink is given by $u_r = \frac{q}{2\pi r}$. At infinity, r is very very large hence velocity is zero.

or $$p = -\frac{1.174^2}{2} \times \rho = -\frac{1.174^2 \times 1000}{2} = \mathbf{-689.14\ N/m^2.\ Ans.}$$

5.17.2 Doublet. It is a special case of a source and sink pair (both of them are of equal strength) when the two approach each other in such a way that the distance $2a$ between them approaches zero and the product $2a \cdot q$ remains constant. This product $2a \cdot q$ is known as doublet strength and is denoted by μ.

$\therefore$ Doublet strength, $\mu = 2a \cdot q$ ...(5.48)

Let q and $(-q)$ may be the strength of the source and the sink respectively as shown in Fig. 5.43. Let $2a$ be the distance between them and P be any point in the combined field of source and sink.

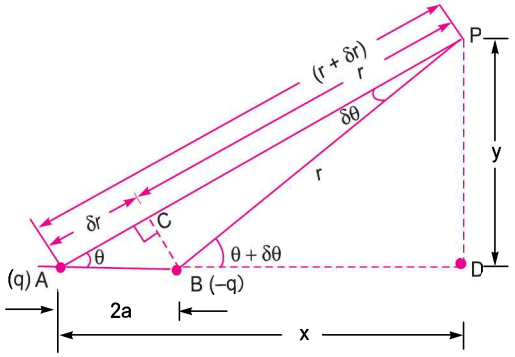


Fig. 5.43

Let θ is the angle made by P at A whereas $(\theta + \delta\theta)$ is the angle at B.

Now the stream function at P,

$$\psi = \frac{q\theta}{2\pi} - \frac{q}{2\pi}(\theta + \delta\theta) = -\frac{q}{2\pi}\delta\theta \qquad ...(5.49)$$

From B, draw $BC \perp$ on AP. Let $AC = \delta r$, $CP = r$ and $AP = r + \delta r$. Also angle $BPC = \delta\theta$. The angle $\delta\theta$ is very small. The distance BC can be taken equal to $r \times \delta\theta$. In triangle ABC, angle $BCA = 90°$ and hence distance BC is also equal to $2a \cdot \sin\theta$. Equating the two values of BC, we get

$$r \times \delta\theta = 2a \cdot \sin\theta$$

$\therefore$ $$\delta\theta = \frac{2a \cdot \sin\theta}{r}$$

Substituting the value of $\delta\theta$ in equation (5.49), we get

$$\psi = -\frac{q}{2\pi} \times \frac{2a \sin\theta}{r}$$

$$= -\frac{\mu}{2\pi} \times \frac{\sin\theta}{r} \qquad [\because \ 2a \cdot q = \mu \text{ from equation (5.48)}] \ ...(5.50)$$

In Fig. 5.43, when $2a \to 0$, the angle $\delta\theta$ subtended by point P with A and B becomes very small. Also $\delta r \to 0$ and AP becomes equal to r. Then

$$\sin\theta = \frac{PD}{AP} = \frac{y}{r}$$

Also $$AP^2 = AD^2 + PD^2 \quad \text{or} \quad r^2 = x^2 + y^2$$

Substituting the value of $\sin\theta$ in equation (5.50), we get

$$\psi = -\frac{\mu}{2\pi} \times \frac{y}{r} \times \frac{1}{r} = -\frac{\mu y}{2\pi r^2} = -\frac{\mu y}{2\pi\left(x^2 + y^2\right)} \qquad (\because \quad r^2 = x^2 + y^2)$$

...(5.50*A*)

or $$x^2 + y^2 = -\frac{\mu y}{2\pi\psi} \quad \text{or} \quad x^2 + y^2 + \frac{\mu y}{2\pi\psi} = 0$$

The above equation can be written as

$$x^2 + y^2 + 2 \times y \times \frac{\mu}{4\pi\psi} + \left(\frac{\mu}{4\pi\psi}\right)^2 - \left(\frac{\mu}{4\pi\psi}\right)^2 = 0 \qquad \left[\text{Adding and subtracting } \left(\frac{\mu}{4\pi\psi}\right)^2\right]$$

or $$x^2 + \left(y + \frac{\mu}{4\pi\psi}\right)^2 = \left(\frac{\mu}{4\pi\psi}\right)^2 \qquad \text{...(5.51)}$$

The above is the equation of a circle with centre $\left(0, \frac{\mu}{4\pi\psi}\right)$ and radius $\frac{\mu}{4\pi\psi}$. The centre of the circle lies on y-axis at a distance of $\frac{\mu}{4\pi\psi}$ from x-axis. As the radius of the circle is also equal to $\frac{\mu}{4\pi\psi}$, hence the circle will be tangent to the x-axis. Hence stream lines of the doublet will be the family of circles tangent to the x-axis as shown in Fig. 5.44.

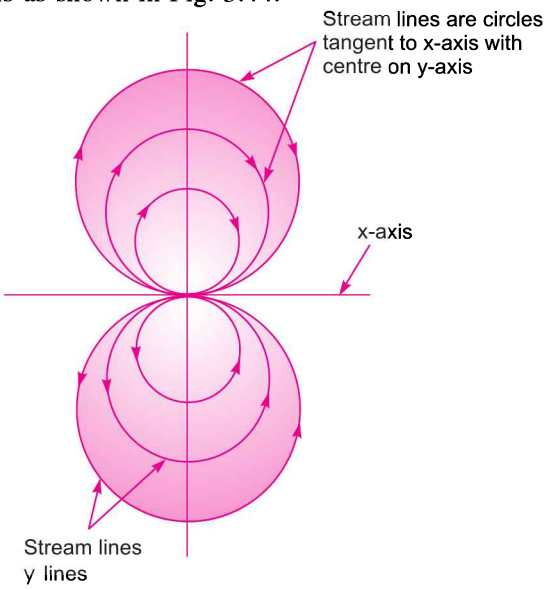


Fig. 5.44 *Stream lines for a doublet.*

Potential function at P

Refer to Fig. 5.43. The potential function at P is given by

$$\phi = \frac{q}{2\pi} \log_e (r + \delta r) + \left(-\frac{q}{2\pi}\right) \log_e r \qquad \text{[Refer to equation (5.41)]}$$

$$= \frac{q}{2\pi}\log_e(r+\delta r) - \frac{q}{2\pi}\log_e r = \frac{q}{2\pi}\log_e\left(\frac{r+\delta r}{r}\right) = \frac{q}{2\pi}\log_e\left(1+\frac{\delta r}{r}\right)^*$$

$$= \frac{q}{2\pi}\left[\frac{\delta r}{r} + \left(\frac{\delta r}{r}\right)^2 \times \frac{1}{2} + ...\right]$$

$$= \frac{q}{2\pi}.\frac{\delta r}{r} \qquad \left[\text{As } \frac{\delta r}{r} \text{ is a small quantity. Hence } \left(\frac{\delta r}{r}\right)^2 \text{ becomes negligible}\right]$$

But in Fig. 5.43, from triangle ABC, we get $\frac{\delta r}{2a} = \cos\theta$

$\therefore \qquad \delta r = 2a\cos\theta$

Substituting the value of δr, we get

$$\phi = \frac{q}{2\pi} \times \frac{2a\cos\theta}{r}$$

$$= \frac{\mu}{2\pi} \times \frac{\cos\theta}{r} \qquad [\because 2a \times q = \mu \text{ from equation } (i)] \quad ...(5.52)$$

In Fig. 5.43, when $2a \to 0$, the angle $\delta\theta$ becomes very small.

Also $\delta r \to 0$ and AP becomes equal to r. Then

$$\cos\theta = \frac{AD}{AP} = \frac{x}{r}$$

Also $AP^2 = AD^2 + PD^2$ or $r^2 = x^2 + y^2$

Substituting the value of $\cos\theta$ in equation (5.52), we get

$$\phi = \frac{\mu}{2\pi} \times \left(\frac{x}{r}\right) \times \frac{1}{r} = \frac{\mu}{2\pi} \times \frac{x}{r^2}$$

$$= \frac{\mu}{2\pi} \times \frac{x}{\left(x^2+y^2\right)} \qquad [\because r^2 = x^2 + y^2]$$

or $$x^2 + y^2 = \frac{\mu}{2\pi} \times \frac{x}{\phi} \quad \text{or} \quad x^2 + y^2 - \frac{\mu}{2\pi} \times \frac{x}{\phi} = 0$$

The above equation can be written as

$$x^2 - \frac{\mu}{2\pi}\frac{x}{\phi} + \left(\frac{\mu}{4\pi\phi}\right)^2 - \left(\frac{\mu}{4\pi\phi}\right)^2 + y^2 = 0 \qquad \left[\text{Adding and subtracting } \left(\frac{\mu}{4\pi\phi}\right)^2\right]$$

or $$\left(x - \frac{\mu}{4\pi\phi}\right)^2 + y^2 = \left(\frac{\mu}{4\pi\phi}\right)^2 \quad ...(5.53)$$

The above is the equation of a circle with centre $\left(\frac{\mu}{4\pi\phi}, 0\right)$ and radius $\left(\frac{\mu}{4\pi\phi}\right)$. The centre of the circle lies on x-axis at a distance of $\frac{\mu}{4\pi\phi}$ from y-axis. As the radius of the circle is equal to the distance of the centre of the circle from the y-axis, hence the circle will be tangent to the y-axis.

---

* Expansion of $\log_e(1+x) = x + \frac{x^2}{2} + ...$

Hence the potential lines of a doublet will be a family of circles tangent to the y-axis with their centres on the x-axis as shown in Fig. 5.45.

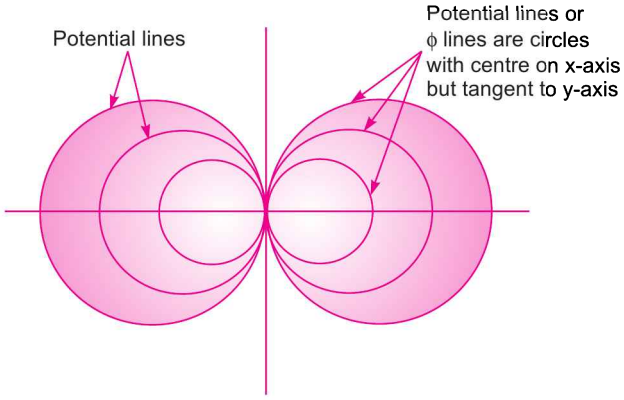


Fig. 5.45 *Potential lines for a doublet.*

Problem 5.38 *A point P(0.5, 1) is situated in the flow field of a doublet of strength 5 m^2/s. Calculate the velocity at this point and also the value of the stream function.*

Solution. Given : Point $P(0.5, 1)$. This means $x = 0.5$ and $y = 1.0$

Strength of doublet, $\mu = 5\ m^2/s$

(*i*) *Velocity at point P*

The velocity at the given point can be obtained if we know the stream function (ψ). But stream function is given by equation (5.50*A*) as

$$\psi = -\frac{\mu}{2\pi} \times \frac{y}{\left(x^2 + y^2\right)}$$

The velocity components u and v are obtained from the stream function as

$$u = \frac{\partial \psi}{\partial y} = \frac{\partial}{\partial y}\left[-\frac{\mu}{2\pi} \times \frac{y}{\left(x^2 + y^2\right)}\right]$$

$$= -\frac{\mu}{2\pi}\frac{\partial}{\partial y}\left[\frac{y}{\left(x^2 + y^2\right)}\right] \qquad \left(\because \frac{\mu}{2\pi} \text{ is a constant term}\right)$$

$$= -\frac{\mu}{2\pi}\left[\frac{x^2 - y^2}{\left(x^2 + y^2\right)^2}\right]$$

$$\left[\because \frac{\partial}{\partial y}\left[y\left(x^2 + y^2\right)^{-1}\right] = y[-1]\left[x^2 + y^2\right]^{-2}[2y] + \left(x^2 + y^2\right)^{-1}.1\right.$$

$$\left. = \frac{-2y^2}{\left(x^2 + y^2\right)^2} + \frac{1}{\left(x^2 + y^2\right)} = \frac{-2y^2 + x^2 + y^2}{\left(x^2 + y^2\right)^2} = \frac{x^2 - y^2}{\left(x^2 + y^2\right)^2}\right]$$

and

$$v = -\frac{\partial \psi}{\partial x} = -\frac{\partial}{\partial x}\left[-\frac{\mu}{2\pi} \times \frac{y}{\left(x^2 + y^2\right)}\right]$$

$$= \frac{\mu}{2\pi}\frac{\partial}{\partial x}\left[\frac{y}{\left(x^2+y^2\right)}\right] = \frac{\mu}{2\pi}\left[\frac{-2xy}{\left(x^2+y^2\right)^2}\right]$$

Substituting the values of $\mu = 5$ m^2/s, $x = 0.5$ and $y = 1.0$, we get the velocity components as

$$u = -\frac{\mu}{2\pi}\left[\frac{x^2-y^2}{\left(x^2+y^2\right)^2}\right] = -\frac{5}{2\pi}\left[\frac{0.5^2-1^2}{\left(0.5^2+1^2\right)^2}\right] = -\frac{5}{2\pi}\frac{0.75}{1.25^2} = -0.382$$

and $$v = \frac{\mu}{2\pi}\left[\frac{-2xy}{\left(x^2+y^2\right)^2}\right] = \frac{5}{2\pi}\left[\frac{-2\times0.5\times1}{\left(0.5^2+1^2\right)^2}\right] = \frac{5}{2\pi}\left[\frac{-1}{1.25^2}\right] = -0.509$$

$\therefore$ Resultant velocity, $V = \sqrt{u^2+v^2} = \sqrt{(-0.382)^2+(-0.509)^2}$ = **0.636 m/s. Ans.**

(*ii*) *Value of stream function at point P*

$$\psi = -\frac{\mu}{2\pi}\frac{y}{\left(x^2+y^2\right)} = -\frac{5}{2\pi}\times\frac{1.0}{\left(0.5^2+1^2\right)} = -\frac{5}{2\pi}\times\frac{1}{1.25}$$

$$= \mathbf{-\ 0.636\ m^2/s.\ Ans.}$$

Solution in polar co-ordinates

The above question can also be done in r, θ (*i.e.,* polar) co-ordinates. The stream function in r, θ co-ordinates is given by equation (5.50) as

$$\psi = -\frac{\mu}{2\pi}\times\frac{\sin\theta}{r} \qquad ...(i)$$

and velocity components in radial and tangential directions are given as

$$u_r = \frac{1}{r}\times\frac{\partial\psi}{\partial\theta} = \frac{1}{r}\frac{\partial}{\partial\theta}\left[-\frac{\mu}{2\pi}\frac{\sin\theta}{r}\right]$$

$$= \frac{1}{r}\times\left(-\frac{\mu}{2\pi}\right)\times\frac{1}{r}\frac{\partial}{\partial\theta}(\sin\theta)$$

$$\left[\because\ \frac{\mu}{2\pi}\text{ is a constant term and also } r \text{ is constant w. r. t. } \theta\right]$$

$$= -\frac{\mu}{2\pi}\times\frac{1}{r^2}\cos\theta \qquad ...(ii)$$

and $$u_\theta = -\frac{\partial\psi}{\partial r} = -\frac{\partial}{\partial r}\left[-\frac{\mu}{2\pi}\frac{\sin\theta}{r}\right]$$

$$= -\left(-\frac{\mu}{2\pi}\sin\theta\right)\frac{\partial}{\partial r}\left[\frac{1}{r}\right] = \frac{\mu}{2\pi}\sin\theta\,(-1)\,.\,\frac{1}{r^2}$$

$$\left[\because\ \frac{\mu\sin\theta}{2\pi}\text{ is a constant w. r. t. } r\right]$$

$$= -\frac{\mu}{2\pi}\times\frac{\sin\theta}{r^2} \qquad ...(iii)$$

Now $$r = \sqrt{x^2 + y^2} = \sqrt{0.5^2 + 1^2} = \sqrt{1.25}$$

$\therefore$ $$\sin\theta = \frac{y}{r} = \frac{1}{\sqrt{1.25}} = 0.894 \text{ and } \cos\theta = \frac{x}{r} = \frac{0.5}{\sqrt{1.25}} = 0.447$$

Substituting the values of r, $\sin\theta$ and $\cos\theta$ in above equations (i), (ii) and (iii), we get

$$\psi = -\frac{\mu}{2\pi}\frac{\sin\theta}{r} = -\frac{5}{2\pi} \times \frac{0.894}{\sqrt{1.25}} = \mathbf{-0.636\ m^2/s.\ Ans.}$$

$$u_r = -\frac{\mu}{2\pi} \times \frac{1}{r^2} \times \cos\theta = -\frac{5}{2\pi} \times \frac{1}{(1.25)} \times 0.447 = -0.2845 \text{ m/s}$$

and $$u_\theta = -\frac{\mu}{2\pi} \times \frac{\sin\theta}{r^2} = -\frac{5}{2\pi} \times \frac{0.894}{1.25} = -0.569 \text{ m/s}$$

$\therefore$ Resultant velocity, $$V = \sqrt{u_r^2 + u_\theta^2}$$

$$= \sqrt{(-0.2845)^2 + (-0.569)^2} = \mathbf{0.636\ m/s.\ Ans.}$$

5.17.3 A Plane Source in a Uniform Flow (Flow Past a Half-Body). Fig. 5.46 (a) shows a uniform flow of velocity U and Fig. 5.46 (b) shows a source flow of strength q. When this uniform flow is flowing over the source flow, a resultant flow will be obtained as shown in Fig. 5.46. This resultant flow is also known as the flow past a half-body. Let the source is placed on the origin O. Consider a point $P(x, y)$ lying in the resultant flow field with polar co-ordinates r and θ as shown in Fig. 5.46.

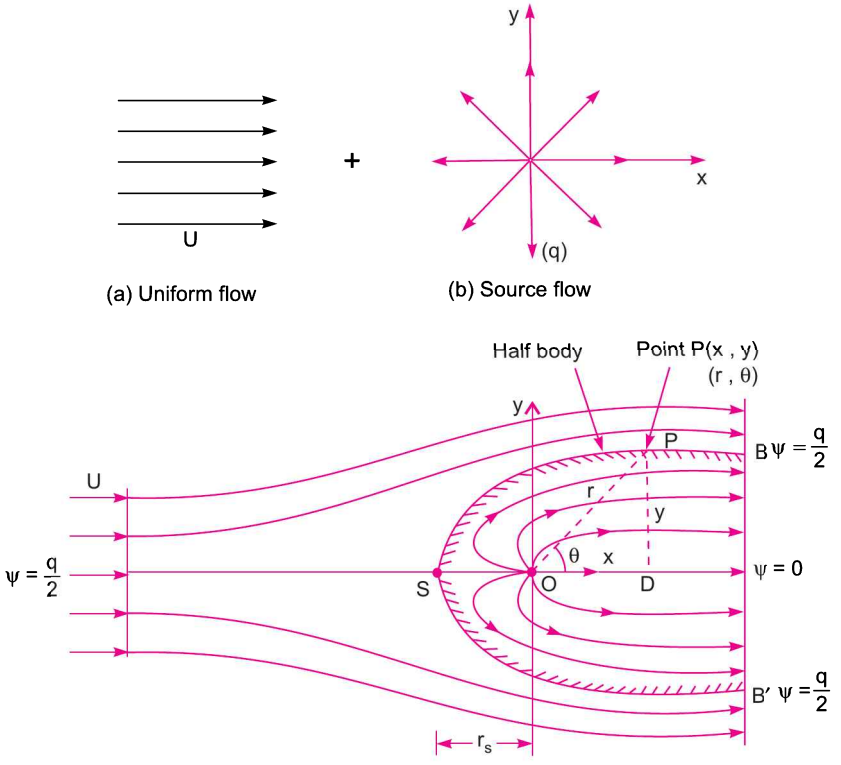


Fig. 5.46 *Flow pattern resulting from the combination of a uniform flow and a source.*

The stream function (ψ) and potential function (ϕ) for the resultant flow are obtained as given below :

ψ = Stream function due to uniform flow + stream function due to source

$$= U \,.\, y + \frac{q}{2\pi}\,\theta \qquad \ldots(5.54)$$

$$= U \,.\, r \sin\theta + \frac{q}{2\pi}\,\theta \qquad (\because \quad y = r \sin\theta) \ \ldots(5.54A)$$

and ϕ = Velocity potential function due to uniform flow + Velocity potential function due to source

$$= U \,.\, x + \frac{q}{2\pi}\log_e r = U \,.\, r\cos\theta + \frac{q}{2\pi}\log_e r \qquad \ldots(5.54B)$$

The following are the important points for the resultant flow pattern :

(*i*) *Stagnation point.* On the left side of the source, at the point S lying on the x-axis, the velocity of uniform flow and that due to source are equal and opposite to each other. Hence the net velocity of the combined flow field is zero. This point is known as stagnation point and is denoted by S. The polar co-ordinates of the stagnation point S are r_S and π, where r_S is radial distance of point S from O.

The net velocity (or resultant velocity) is zero at the stagnation point S.

$$u_r = \frac{1}{r}\frac{\partial\psi}{\partial\theta} = \frac{1}{r}\frac{\partial}{\partial\theta}\left(U.r\sin\theta + \frac{q}{2\pi}\theta\right) \qquad \left[\because \quad \psi = U.r\sin\theta + \frac{q}{2\pi}\theta\right]$$

$$\therefore \qquad = \frac{1}{r}\left[U.r\cos\theta + \frac{q}{2\pi}\right] = U\,.\cos\theta + \frac{q}{2\pi r}$$

At the stagnation point, $\theta = \pi$ radians (180°) and $r = r_S$ and net velocity is zero. This means $u_r = 0$ and $v_\theta = 0$. Substituting these values in the above equation, we get

$$0 = U\,.\cos 180^\circ + \frac{q}{2\pi r_S} \qquad [\because\ u_r = 0,\ \theta = 180^\circ \text{ and } r = r_S]$$

$$= -U + \frac{q}{2\pi r_S} \quad \text{or} \quad U = \frac{q}{2\pi r_S}$$

or $$r_S = \frac{q}{2\pi U} \qquad \ldots(5.55)$$

From the above equation it is clear that position of stagnation point depends upon the free stream velocity U and source strength q. At the stagnation point, the value of stream function is obtained from equation (5.54*A*) as

$$\psi = U\,.\,r\sin\theta + \frac{q}{2\pi}\,.\,\theta$$

For the stagnation point, the above equation becomes as

$$\therefore \qquad \psi_s = U.r_s\sin 180^\circ + \frac{q}{2\pi}\times\theta$$

[$\because$ At stagnation point, $\theta = \pi$ radians = 180° and $r = r_S$]

$$= 0 + \frac{q}{2} = \frac{q}{2} \qquad \ldots(5.56)$$

The above relation gives the equation of stream line passing through stagnation point. We know that no fluid mass crosses a stream line. Hence a stream line is a *virtual solid surface.*

(*ii*) *Shape of resultant flow.* At the stagnation point S, the net velocity is zero. The fluid particles that issue from the source cannot proceed further to the left of stagnation point. They are carried along the contour BSB' that separates the source flow from uniform flow. The curve BSB' can be regarded as the **solid boundary** of a round nosed body such as a bridge pier around which the uniform flow is forced to pass. The contour BSB' is called the half body, because it has only the leading point, it trails to infinity at down stream end.

The value of stream function of the stream line passing through stagnation point S and passing over the solid boundary (*i.e.*, curve BSB') is $\psi_S = \frac{q}{2}$.

Thus the composite flow consists of :

(1) flow over a plane half-body (*i.e.*, flow over curve BSB') outside $\psi = \frac{q}{2}$ and

(2) source flow within the plane half-body.

The plane half-body is described by the dividing stream line, $\psi = \frac{q}{2}$.

But the stream function at any point in the combined flow field is given by equation (5.54) as

$$\psi = U \,.\, y + \frac{q}{2\pi}\,\theta$$

If we take $\psi = \frac{q}{2}$ in the above equation, we will get the equation of the dividing stream line.

$\therefore$ Equation of the dividing stream line (*i.e.*, equation of curve BSB') will be

$$\frac{q}{2} = U \,.\, y + \frac{q}{2\pi} \,.\, \theta \text{ or } U \,.\, y = \frac{q}{2} - \frac{q}{2\pi}\,\theta = \frac{q}{2}\left(1 - \frac{\theta}{\pi}\right)$$

or
$$y = \frac{q}{2U}\left(1 - \frac{\theta}{\pi}\right) \qquad \ldots(5.57)$$

From the above equation, the main dimensions of the plane half-body may be obtained. From this equation, it is clear that y is maximum, when $\theta = 0$.

Hence At $\theta = 0$, y is maximum and $y_{max} = \frac{q}{2U}$ ... the maximum ordinate

At $\theta = \frac{\pi}{2}$, $y = \frac{q}{2U}\left(1 - \frac{\pi}{2} \,.\, \frac{1}{\pi}\right) = \frac{q}{4U}$ ... the ordinate above the origin

At $\theta = \pi$, $y = \frac{q}{2U}\left(1 - \frac{\pi}{\pi}\right) = 0$ ... the leading point of the half-body

At $\theta = \frac{3\pi}{2}$, $y = \frac{q}{2U}\left(1 - \frac{3\pi}{2\pi}\right) = -\frac{q}{4U}$ ... the ordinate below the origin.

The main dimensions are shown in Fig. 5.47.

(*iii*) *Resultant velocity at any point*

The velocity components at any point in the flow field are given by

$$u_r = \frac{1}{r}\frac{\partial \psi}{\partial \theta} = \frac{1}{r}\frac{d}{d\theta}\left[U.r\sin\theta + \frac{q}{2\pi}\theta\right]$$

$$= \frac{1}{r}\left[U.r\cos\theta + \frac{q}{2\pi}\right] = U.\cos\theta + \frac{q}{2\pi r}$$

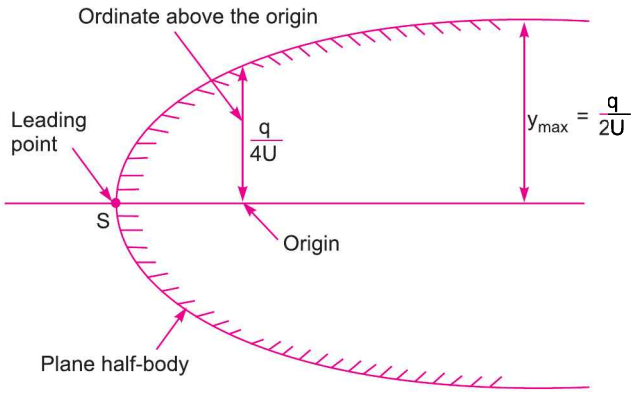


Fig. 5.47

The above equation gives the radial velocity at any point in the flow field. This radial velocity is due to uniform flow and due to source. Due to source the radial velocity is $\frac{q}{2\pi r}$. Hence the velocity due to source diminishes with increase in radial distance from the source. At large distance from the source the contribution of source is negligible and hence free stream uniform flow is not influenced by the presence of source.

$$u_\theta = -\frac{\partial\psi}{\partial r} = -\frac{\partial}{\partial r}\left[U.r\sin\theta + \frac{q}{2\pi}\theta\right]$$

$$= -[U\ .\ \sin\theta + 0] = -U\sin\theta \qquad \left[\because \frac{q}{2\pi}\theta \text{ is constant w. r. t. } r\right]$$

$\therefore$ Resultant velocity, $V = \sqrt{u_r^2 + u_\theta^2}$

(*iv*) *Location of stagnation point*

At the stagnation point, the velocity components are zero. Hence equating the radial and tangential velocity components to zero, we get

$$u_r = 0 \quad \text{or} \quad U\cos\theta + \frac{q}{2\pi r} = 0 \quad \text{or} \quad U\cos\theta = -\frac{q}{2\pi r}$$

or $$r\cos\theta = -\frac{q}{2\pi U} \quad \text{But} \quad r\cos\theta = x$$

$\therefore$ $$x = -\frac{q}{2\pi U}$$

When $u_\theta = 0$ or $-U\sin\theta = 0$ or $\sin\theta = 0$ as U cannot be zero

or $\theta = 0$ or π But $y = r\sin\theta$ $\therefore$ $y = 0$

Hence stagnation point is at $\left(-\frac{q}{2\pi U}, 0\right)$, the leading point of the half-body.

(*v*) *Pressure at any point in flow field*

Let p_0 = pressure at infinity where velocity is U

p = pressure at any point P in the flow field, where velocity is V

Now applying the Bernoulli's equation at a point at infinity and at a point P in the flow field, we get

$$\frac{p_0}{\rho g}+\frac{U^2}{2g}=\frac{p}{\rho g}+\frac{V^2}{2g} \quad \text{or} \quad \frac{U^2}{2g}-\frac{V^2}{2g}=\frac{p}{\rho g}-\frac{p_0}{\rho g}=\frac{p-p_0}{\rho g}$$

The pressure co-efficient is defined as

$$C_p=\frac{p-p_0}{\frac{1}{2}\rho U^2}$$

$$=\frac{\rho g\left[\frac{U^2}{2g}-\frac{V^2}{2g}\right]}{\frac{1}{2}\rho U^2} \qquad \left[\because \; p-p_0=\rho g\left(\frac{U^2}{2g}-\frac{V^2}{2g}\right)\right]$$

$$=\frac{U^2-V^2}{U^2}=1-\left(\frac{V}{U}\right)^2 \qquad \ldots(5.58)$$

Problem 5.39 *A uniform flow with a velocity of 3 m/s is flowing over a plane source of strength 30 m^2/s. The uniform flow and source flow are in the same plane. A point P is situated in the flow field. The distance of the point P from the source is 0.5 m and it is at an angle of 30° to the uniform flow. Determine : (i) stream function at point P, (ii) resultant velocity of flow at P and (iii) location of stagnation point from the source.*

Solution. Given : Uniform velocity, U = 3 m/s ; source strength, q = 30 m^2/s ; co-ordinates of point P are r = 0.5 m and $\theta = 30°$.

(*i*) *Stream function at point P*

The stream function at any point in the combined flow field is given by equation (5.54*A*)

$$\psi = U \,.\, r \sin\theta + \frac{q}{2\pi}\theta$$

at point P, r = 0.5 m and $\theta = 30°$ or $\frac{30}{180}\times\pi$ radians.

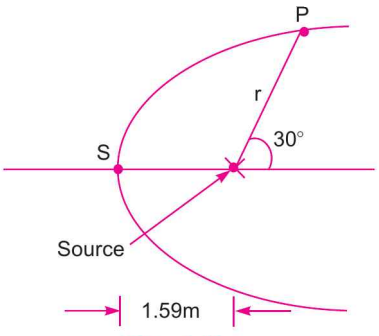


Fig. 5.48

∴ Stream function at point P,

$$\psi = 3\times 0.5\times \sin 30° + \frac{30}{2\pi}\times\left(\frac{30}{180}\times\pi\right)$$

$$= 0.75 + 2.5 = \mathbf{3.25\ m^2/s.\ Ans.}$$

(*ii*) *Resultant velocity at P*

The velocity components anywhere in the flow are given by

$$u_r = \frac{1}{r}\frac{\partial \psi}{\partial \theta} = \frac{1}{r}\frac{\partial}{\partial \theta}\left[U.r\sin\theta + \frac{q}{2\pi}\theta\right]$$

$$= \frac{1}{r}\left[U.r\cos\theta + \frac{q}{2\pi}\right] = U.\cos\theta + \frac{q}{2\pi r}$$

$$= 3 \times \cos 30° + \frac{30}{2\pi \times 0.5} \qquad (\because \text{ At } P,\ r = 0.5,\ \theta = 30°,\ q = 30)$$

$$= 2.598 + 9.55 = 12.14$$

and

$$u_\theta = \frac{-\partial \psi}{\partial r} = -\frac{\partial}{\partial r}\left[U.r\sin\theta + \frac{q}{2\pi}.\theta\right]$$

$$= -U\sin\theta + 0 = -U\sin\theta$$

$$= -3 \times \sin 30° = -1.5$$

$\therefore$ Resultant velocity, $V = \sqrt{u_r^2 + u_\theta^2}$

$$= \sqrt{12.14^2 + (-1.5)^2} = \mathbf{12.24\ m/s.\ Ans.}$$

(*iii*) *Location of stagnation point*

The horizontal distance of the stagnation point S from the source is given by equation (5.55) as

$$r_S = \frac{q}{2\pi U} = \frac{30}{2\pi \times 3} = \mathbf{1.59\ m.\ Ans.}$$

The stagnation point will be at a distance of 1.59 m to the left side of the source on the x-axis.

Problem 5.40 *A uniform flow with a velocity of 20 m/s is flowing over a source of strength 10 m^2/s. The uniform flow and source flow are in the same plane. Obtain the equation of the dividing stream line and sketch the flow pattern.*

Solution. Given : Uniform velocity, U = 20 m/s ; Source strength, q = 10 m^2/s

(*i*) *Equation of the dividing stream line*

The stream function at any point in the combined flow field is given by equation (5.54*A*)

$$\psi = U\ .\ r\sin\theta + \frac{q}{2\pi}\theta$$

$$= 20 \times r\sin\theta + \frac{10}{2\pi}\theta \qquad (\because \quad U = 20 \text{ m/s and } q = 10\ m^2/s)$$

The value of the stream function for the dividing stream line is $\psi = \frac{q}{2}$. Hence substituting $\psi = \frac{q}{2}$ in the above equation, we get the equation of the dividing stream line.

$\therefore$

$$\frac{q}{2} = 20r\sin\theta + \frac{10}{2\pi}\theta$$

or

$$\frac{10}{2} = 20r\sin\theta + \frac{10}{2\pi}\theta \qquad (\because \quad q = 10)$$

or $$5 = 20r \sin\theta + \frac{10}{2\pi}\theta = 20y + \frac{10}{2\pi}\theta \qquad (\because \quad r\sin\theta = y)$$

$$\therefore \qquad 20y = 5 - \frac{10}{2\pi}\theta$$

or $$y = \frac{5}{20} - \frac{10}{2\pi} \times \frac{\theta}{20} = 0.25 - \frac{\theta}{4\pi} \qquad ...(i)$$

The above relation gives the equation of the dividing stream line.

From the above equation, for different values of θ the value of y is obtained as :

Value of θ	*Value of y from (i)*	*Remarks*
0	0.25 m	Max. half width of body
$\frac{\pi}{2}$	0.125 m	The +ve ordinate above the origin
π	0	The leading point
$\frac{3\pi}{2}$	– 0.125 m	The –ve ordinate below the origin
2π	– 0.25 m	The max. –ve ordinate

(*ii*) *Sketch of flow pattern*

For sketching the flow pattern, let us first find the location of the stagnation point. The horizontal distance of the stagnation point S from the source is given by the equation,

$$r_s = \frac{q}{2\pi U} = \frac{10}{2\pi \times 20} = 0.0795 \text{ m}$$

Hence the stagnation point lies on the x-axis at a distance of 0.0795 m or 79.5 mm from the source towards left of the source. The flow pattern is shown in Fig. 5.49.

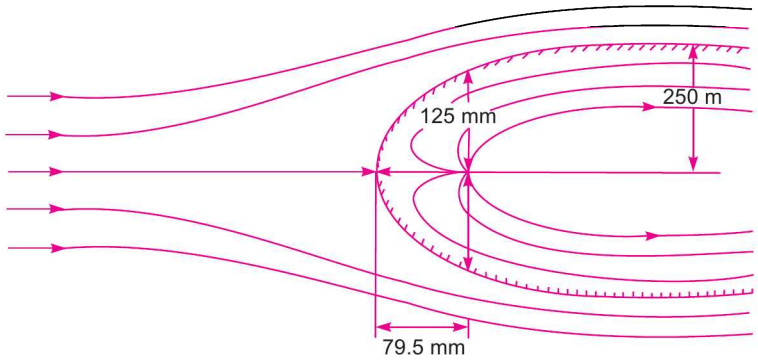


Fig. 5.49

Problem 5.41 *A uniform flow with a velocity of 2 m/s is flowing over a source placed at the origin. The stagnation point occurs at (– 0.398, 0). Determine :*

(i) Strength of the source, (ii) Maximum width of Rankine half-body and

(iii) Other principal dimensions of the Rankine half-body.

Solution. Given :

Uniform velocity, $U = 2$ m/s

Co-ordinates of stagnation point = (– 0.398, 0)

This means $r_s = 0.398$ and stagnation point lies on x-axis at a distance of 0.398 m towards left of origin. The source is placed at origin.

(*i*) *Strength of the source*

Let q = strength of the source

We know that $$r_s = \frac{q}{2\pi U}$$

or $$q = 2\pi U \times r_s = 2\pi \times 2 \times 0.398 = 5.0014 \text{ m}^2/\text{s} \simeq \mathbf{5\ m^2/s.\ Ans.}$$

(*ii*) *Maximum width of Rankine half-body*

The main dimensions of the Rankine half-body are obtained from equation (5.57) as

$$y = \frac{q}{2U}\left(1 - \frac{\theta}{\pi}\right) \qquad ...(i)$$

The value of y is maximum, when $\theta = 0$.

$$\therefore \quad y_{max} = \frac{q}{2U}\left(1 - \frac{0}{\pi}\right) = \frac{q}{2U} = \frac{5}{2 \times 2} = 1.25 \text{ m}$$

$\therefore$ Maximum width of Rankine body = $2 \times y_{max} = 2 \times 1.25 = \mathbf{2.5\ m.\ Ans.}$

(*iii*) *Other Principal dimensions of Rankine half-body*

Using equation (5.57), we get

$$y = \frac{q}{2U}\left(1 - \frac{\theta}{\pi}\right)$$

At $\theta = \frac{\pi}{2}$, $$y = \frac{q}{2U}\left[1 - \frac{\left(\frac{\pi}{2}\right)}{\pi}\right] = \frac{q}{2U}\left[1 - \frac{1}{2}\right] = \frac{q}{4U} = \frac{5}{4 \times 2} = 0.625 \text{ m}$$

The above value gives the upper ordinate at the origin, where source is placed.

$\therefore$ Width of body at origin = $2 \times 0.625 = 1.25$ m

At the stagnation point, the width of the body is zero.

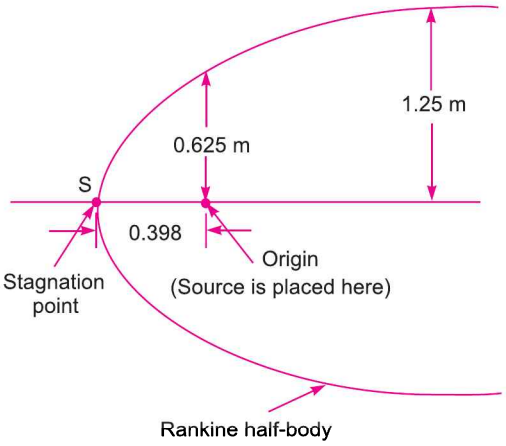


Fig. 5.50

5.17.4 A Source and Sink Pair in a Uniform Flow (Flow Past a Rankine Oval Body). Fig. 5.51 (*a*) shows a uniform flow of velocity U and Fig. 5.51 (*b*) shows a source sink pair of equal strength. When this uniform flow is flowing over the source sink pair, a resultant flow will be obtained as shown in Fig. 5.51 (*c*). This resultant flow is also known as the flow past a Rankine oval body.

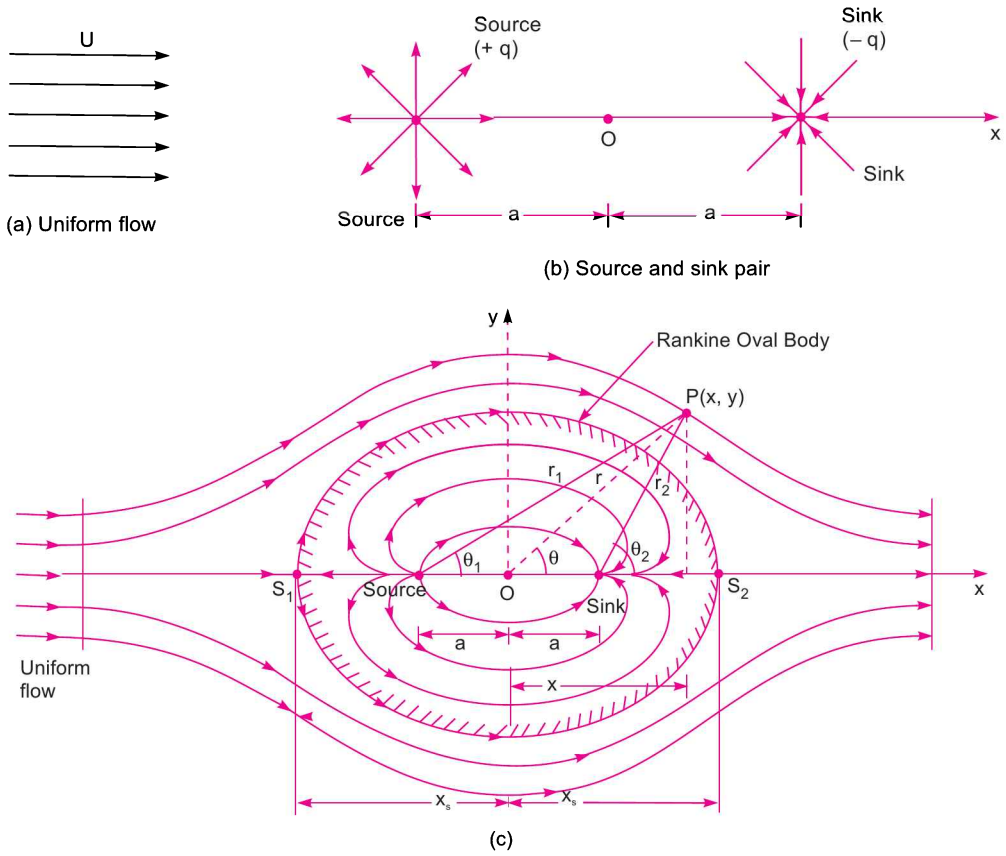


Fig. 5.51

Let U = Velocity of uniform flow along x-axis

q = Strength of source

$(-q)$ = Strength of sink

$2a$ = Distance between source and sink which is along x-axis.

The origin O of the x-y co-ordinates is mid-way between source and sink. Consider a point $P(x, y)$ lying in the resultant flow field. The stream function (ψ) and velocity potential function (ϕ) for the resultant flow field are obtained as given below :

ψ = Stream function due to uniform flow + stream function due to source + stream function due to sink

$$= \psi_{\text{uniform flow}} + \psi_{\text{source}} + \psi_{\text{sink}}$$

$$= U \times y + \frac{q}{2\pi}\theta_1 + \frac{(-q)}{2\pi} \times \theta_2$$

(where θ_1 is the angle made by P with source along x-axis and θ_2 with sink)

$$= U \times y + \frac{q\theta_1}{2\pi} - \frac{q\theta_2}{2\pi} = U \times y + \frac{q}{2\pi}(\theta_1 - \theta_2)$$

$$= U \times r \sin\theta + \frac{q}{2\pi}(\theta_1 - \theta_2) \qquad (\because \quad y = r \sin\theta) \quad ...(5.59)$$

and ϕ = potential function due to uniform flow + potential function due to source + potential function due to sink

$$= \phi_{\text{uniform flow}} + \phi_{\text{source}} + \phi_{\text{sink}}$$

$$= U \times x + \frac{q}{2\pi}\log_e r_1 + \frac{(-q)}{2\pi}\log_e r_2$$

$$= U \times r\cos\theta + \frac{q}{2\pi}[\log_e r_1 - \log_e r_2] \qquad (\because \quad x = r\cos\theta)$$

$$= U \times r\cos\theta + \frac{q}{2\pi}\left[\log_e \frac{r_1}{r_2}\right] \qquad ...(5.60)$$

The following are the important points for the resultant flow pattern :

(*a*) There will be two stagnation points S_1 and S_2, one to the left of the source and other to the right of the sink. At the stagnation points, the resultant velocity (*i.e.*, velocity due to uniform flow, velocity due to source and velocity due to sink) will be zero. The stagnation point S_1 is to the left of the source and stagnation point S_2 will be to the right of the sink on the x-axis.

Let x_S = Distance of the stagnation points from origin O along x-axis.

Let us calculate this distance x_S.

For the stagnation point S_1,

(*i*) Velocity due to uniform flow = U

(*ii*) Velocity due to source = $\dfrac{q}{2\pi(x_S - a)}$ $\left[\because \text{ The velocity at any radius due to source} = \dfrac{q}{2\pi r}, \text{ For } S_1, \text{ the radius from source} = (x_S - a)\right]$

(*iii*) Velocity due to sink = $\dfrac{-q}{2\pi(x_S + a)}$ [$\because$ At S_1, the radius from sink = $(x_S + a)$]

At point S_1, the velocity due to uniform flow is in the positive x-direction whereas due to source and sink are in the –ve x-direction.

$\therefore$ The resultant velocity at $S_1 = U - \dfrac{q}{2\pi(x_S - a)} - \dfrac{(-q)}{2\pi(x_S + a)}$

But the resultant velocity at stagnation point S_1 should be zero.

$\therefore$ $$U - \frac{q}{2\pi(x_S - a)} + \frac{q}{2\pi(x_S + a)} = 0$$

or $$U = \frac{q}{2\pi(x_S - a)} - \frac{q}{2\pi(x_S + a)}$$

$$= \frac{q}{2\pi}\left[\frac{1}{(x_S - a)} - \frac{1}{(x_S + a)}\right] = \frac{q}{2\pi}\left[\frac{(x_S + a) - (x_S - a)}{(x_S - a)(x_S + a)}\right] = \frac{q}{2\pi}\frac{2a}{(x_S^2 - a^2)}$$

or $$x_S^2 - a^2 = \frac{q \,.\, a}{\pi U}$$

or $$x_S^2 = a^2 + \frac{qa}{\pi U} = a^2\left[1 + \frac{q}{\pi a U}\right]$$

$\therefore$ $$x_S = a\sqrt{\left(1 + \frac{q}{\pi a U}\right)} \qquad ...(5.61)$$

The above equation gives the location of the stagnation point on the x-axis.

(b) The stream line passing through the stagnation points is having zero velocity and hence can be replaced by a solid body. This solid body is having a shape of oval as shown in Fig. 5.51. There will be two flow fields, one within the oval contour and the other outside the solid body. The flow field within the oval contour will be due to source and sink whereas the flow field outside the body will be due to uniform flow only.

The shape of solid body is obtained from the stream line having stream function equal to zero. But the stream function is given by equation as

$$\psi = U \times r \sin\theta + \frac{q}{2\pi}(\theta_1 - \theta_2)$$

For the shape of solid body, $\psi = 0$

$\therefore$ $$0 = U \times r \sin\theta + \frac{q}{2\pi}(\theta_1 - \theta_2)$$

or $$U \times r \sin\theta = -\frac{q}{2\pi}(\theta_1 - \theta_2) = \frac{q}{2\pi}(\theta_2 - \theta_1)$$

$\therefore$ $$r = \frac{q}{2\pi}\frac{(\theta_2 - \theta_1)}{U \sin\theta} \qquad ...(5.62)$$

From the above equation, the distances of the surface of the solid body from the origin can be obtained or the shape of the solid body can be obtained. The maximum width of the body (y_{max}) will be equal to OM as shown in Fig. 5.52.

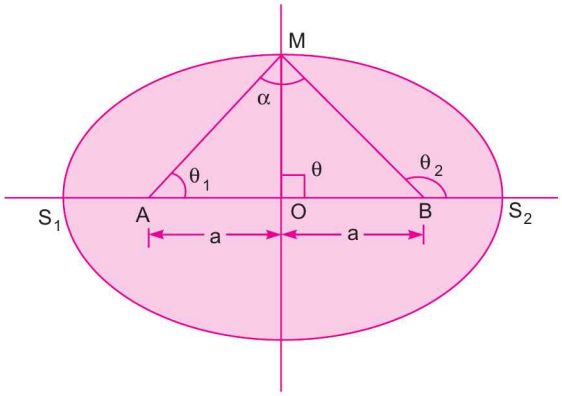


Fig. 5.52

From triangle AOM, we have

$$\tan \theta_1 = \frac{OM}{AO}$$

or $$OM = AO \tan \theta_1 = a \tan \theta_1$$

or $$y_{max} = a \tan \theta_1 \qquad (\because\ OM = y_{max}) \quad ...(5.63)$$

Let us find the value of θ_1.

When the point P lies on M, then $r = OM$, $\theta = 90° = \frac{\pi}{2}$

and $$\theta_2 = 180° - \theta_1 = \pi - \theta_1$$ [Refer to Fig. 5.52]

$[\because \quad AM = BM \quad \therefore \quad \text{Angle } ABM = \text{Angle } BAM = \theta_1]$

Substituting these values in equation (5.62), we get

$$OM = \frac{q}{2\pi} \frac{\left((\pi - \theta_1) - \theta_1\right)}{U \sin \frac{\pi}{2}} = \frac{q}{2\pi} \frac{(\pi - 2\theta_1)}{U}$$

or $$y_{max} = \frac{q(\pi - 2\theta_1)}{2\pi U}$$ [where $OM = y_{max}$]

or $$2\pi U y_{max} = q(\pi - 2\theta_1) \qquad \text{or} \qquad \frac{2\pi U y_{max}}{q} = \pi - 2\theta_1$$

or $$2\theta_1 = \pi - \frac{2\pi U y_{max}}{q} \qquad \text{or} \qquad \theta_1 = \frac{\pi}{2} - \frac{\pi U y_{max}}{q}$$

Substituting this value of θ_1 in equation (5.63), we get

$$y_{max} = a \tan \left[\frac{\pi}{2} - \frac{\pi U y_{max}}{q}\right] = a \cot \left[\frac{\pi U y_{max}}{q}\right] \quad ...(5.64)$$

From the above equation, the value of y_{max} is obtained by hit and trial method till L.H.S. = R.H.S. In this equation $\left(\frac{\pi U y_{max}}{q}\right)$ is in radians.

The length and width of the Rankine oval is obtained as :

Length, $$L = 2 \times x_S$$

$$= 2 \times a \sqrt{\left(1 + \frac{q}{\pi a U}\right)} \qquad \left[\because\ x_S = a \sqrt{\left(1 + \frac{q}{\pi a U}\right)}\right] \quad ...(5.65)$$

and Width, $$B = 2 \times y_{max}$$

$$= 2a \cot \left(\frac{\pi U y_{max}}{q}\right). \quad ...(5.66)$$

Problem 5.42 *A uniform flow of velocity 6 m/s is flowing along x-axis over a source and a sink which are situated along x-axis. The strength of source and sink is 15 m^2/s and they are at a distance of 1.5 m apart. Determine :*

(i) Location of stagnation points, *(ii) Length and width of the Rankine oval*
(iii) Equation of profile of the Rankine body.

Solution. Given : Uniform flow velocity, $U = 6$ m/s
Strength of source and sink, $q = 15$ m^2/s
Distance between source and sink, $2a = 1.5$ m

$$\therefore \quad a = \frac{1.5}{2} = 0.75 \text{ m}$$

(*i*) *Location of stagnation points* (Refer to Fig. 5.51)
For finding the location of the stagnation points, the equation (5.61) is used.

$$\therefore \quad x_s = a\sqrt{\left(1 + \frac{q}{\pi a U}\right)} = 0.75\sqrt{\left(1 + \frac{15}{\pi \times 0.75 \times 6}\right)} = 1.076 \text{ m}$$

The above equation gives the distance of the stagnation points from the origin. There will be two stagnation points.
The distance of stagnation points from the source and sink = $x_s - a = 1.076 - 0.75 =$ **0.326 m. Ans.**
(*ii*) *Length and width of the Rankine oval*
Length, $L = 2 \times x_s = 2 \times 1.076 = 2.152$ m.
Width, $B = 2 \times y_{max}$ ...(*i*)
Let us now find the value of y_{max}
Using equation (5.64), we get

$$y_{max} = a \cot\left(\frac{\pi U y_{max}}{q}\right) = 0.75 \cot\left(\frac{\pi \times 6 \times y_{max}}{15}\right) = 0.75 \cot (0.4\pi\, y_{max})$$

$$= 0.75 \cot\left(0.4\pi\, y_{max} \times \frac{180}{\pi}\right)^\circ$$

$$\left[\because \ (0.4\pi\, y_{max}) \text{ is in radians and hence } (0.4\pi\, y_{max}) \times \frac{180}{\pi} \text{ will be in degrees}\right]$$

$$= 0.75 \cot (72 \times y_{max})^\circ$$

The above equation will be solved by hit and trial method. The value of $x_s = 1.076$. But x_s is equal to length of major axis of Rankine body and y_{max} is the length of minor axis of the Rankine body. The length of minor axis will be less than length of major axis. Let us first assume $y_{max} = 0.8$ m. Then

y_{max}	*L.H.S.*	*R.H.S.*
0.8	0.8	$0.75 \cot (72 \times 0.8)^\circ = 0.75 \cot 51.6^\circ = 0.475$
0.7	0.7	$0.75 \cot (72 \times 0.7)^\circ = 0.75 \cot 50.4^\circ = 0.577$
0.6	0.6	$0.75 \cot (72 \times 0.6)^\circ = 0.75 \cot 43.2^\circ = 0.798$
0.65	0.65	$0.75 \cot (72 \times 0.65)^\circ = 0.75 \cot 46.8^\circ = 0.704$
0.67	0.67	$0.75 \cot (72 \times 0.67)^\circ = 0.75 \cot 48.24^\circ = 0.669 \simeq 0.67$

From above it is clear that, when $y_{max} = 0.67$, then L.H.S. = R.H.S.

$$\therefore \quad y_{max} = 0.67 \text{ m}$$

Substituting this value in equation (*i*), we get

Width, $B = 2 \times y_{max} = 2 \times 0.67 =$ **1.34 m. Ans.**

(*iii*) *Equation of profile of the Rankine body*
The equation of profile of the Rankine body is given by equation (5.62) as

$$r = \frac{q}{2\pi}\frac{(\theta_2 - \theta_1)}{U \sin\theta} = \frac{15}{2\pi}\frac{(\theta_2 - \theta_1)}{6 \times \sin\theta} = \frac{\mathbf{0.398\,(\theta_2 - \theta_1)}}{\mathbf{\sin\theta}}.$$ **Ans.**

5.17.5 A Doublet in a Uniform Flow (Flow Past a Circular Cylinder). Fig. 5.53 (*a*) shows a uniform flow of velocity U in the positive x-direction and Fig. 5.53 (*b*) shows a doublet at the origin. Doublet is a special case of a source and a sink combination in which both of equal strength approach each other such that distance between them tends to be zero. When the uniform flow is flowing over the doublet, a resultant flow will be obtained as shown in Fig. 5.53 (*c*). This resultant flow is **known as the flow past a Rankine oval of equal axes or flow past a circular cylinder.**

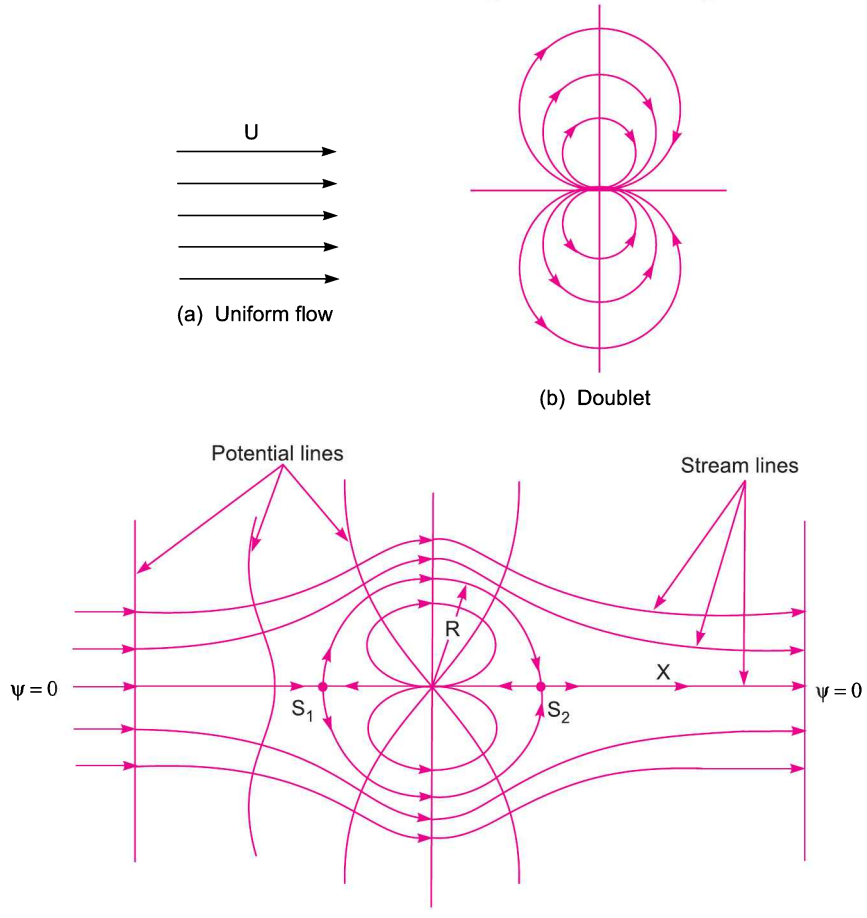


Fig. 5.53

The stream function (ψ) and velocity potential function (ϕ) for the resultant flow is obtained as given below :

ψ = stream function due to uniform flow + stream function due to doublet

$$= U \times y + \left(\frac{-\mu}{2\pi r}\sin\theta\right)$$

$$\left[\text{Stream function due to doublet is given by equation (5.50) as} = -\frac{\mu}{2\pi r}\sin\theta\right]$$

$$= U \times r \times \sin\theta - \frac{\mu}{2\pi r} \sin\theta \qquad (\because \quad y = r \sin\theta)$$

$$= \left(U \times r - \frac{\mu}{2\pi r}\right) \sin\theta \qquad ...(5.67)$$

and ϕ = Potential function due to uniform flow + potential function due to doublet

$$= U \times x + \frac{\mu}{2\pi} \times \frac{\cos\theta}{r}$$

$$\left[\text{From equation (5.52), potential function due to doublet} = \frac{\mu}{2\pi} \times \frac{\cos\theta}{r}\right]$$

$$= U \times r \cos\theta + \frac{\mu}{2\pi} \times \frac{\cos\theta}{r} \qquad (\because \; x = r\cos\theta)$$

$$= \left(U \times r + \frac{\mu}{2\pi r}\right) \cos\theta \qquad ...(5.68)$$

Shape of Rankine oval of equal axes

To get the profile of the Rankine oval of equal axes, the stream line ψ is taken as zero. Hence substituting $\psi = 0$ in equation (5.67), we get

$$0 = \left(U \times r - \frac{\mu}{2\pi r}\right) \sin\theta$$

This means either $\sin\theta = 0$ or $U \times r - \frac{\mu}{2\pi r} = 0$

(*i*) If $\sin\theta = 0$, then $\theta = 0$ and $\pm\pi$ *i.e.*, a horizontal line through the origin of the doublet. This horizontal line is the x-axis.

(*ii*) If $U \times r - \frac{\mu}{2\pi r} = 0$, then $U \times r = \frac{\mu}{2\pi r}$ or $r^2 = \frac{\mu}{2\pi U}$

or $$r = \sqrt{\frac{\mu}{2\pi U}} = \text{a constant as } \mu \text{ and } U \text{ are constant.}$$

Let this constant is equal to R.

$\therefore$ $$r = \sqrt{\frac{\mu}{2\pi U}} = R$$

This gives that the closed body profile is a circular cylinder of radius R with centre on doublet. The dividing stream line corresponds to $\psi = 0$. This stream line is a circle of radius R. The stream line $\psi = 0$ has two stagnation points S_1 and S_2. At S_1, the uniform flow splits into two streams that flow along the circle with radius $R = \sqrt{\frac{\mu}{2\pi U}}$, the two branches meet again at the stagnation point S_2 and the flow continues in the downward direction. The uniform flow occurs outside the circle whereas the flow field due to doublet lies entirely within the circle. The stream function for the composite flow is given by equation (5.67) as

$$\psi = \left(U \times r - \frac{\mu}{2\pi r}\right) \sin\theta = U\left(r - \frac{\mu}{2\pi U r}\right) \sin\theta$$

$$= U\left(r - \frac{R^2}{r}\right) \sin\theta \qquad \left(\because \frac{\mu}{2\pi U} = R^2\right) \quad ...(5.69)$$

Velocity Components (u_r and u_θ)

The velocity components at any point in the flow field are given by,

$$u_r = \frac{1}{r}\frac{\partial\psi}{\partial\theta} = \frac{1}{r}\frac{\partial}{\partial\theta}\left[U\left(r - \frac{R^2}{r}\right)\sin\theta\right] = \frac{1}{r}U\left(r - \frac{R^2}{r}\right)\cos\theta$$

$$= U\left(1 - \frac{R^2}{r^2}\right)\cos\theta \quad ...(5.70)$$

and

$$u_\theta = -\frac{\partial\psi}{\partial r} = -\frac{\partial}{\partial r}\left[U\left(r - \frac{R^2}{r}\right)\sin\theta\right] = -U\left(1 + \frac{R^2}{r^2}\right)\sin\theta$$

$$= -U\left(1 + \frac{R^2}{r^2}\right)\sin\theta \quad ...(5.71)$$

$\therefore$ Resultant velocity, $V = \sqrt{u_r^2 + u_\theta^2}$ ...(5.72)

On the surface of the cylinder, $r = R$

$$u_r = U\left[1 - \frac{R^2}{R^2}\right]\cos\theta \qquad [\because \text{ In equation (5.70), } r = R]$$

$$= 0$$

and

$$u_\theta = -U\left[1 + \frac{R^2}{R^2}\right]\sin\theta = -2U\sin\theta \quad ...(5.73)$$

–ve sign shows the clockwise direction of tangential velocity at that point. The value of u_θ is maximum, when $\theta = 90°$ and $270°$.

At $\theta = 0°$ or $180°$, the value of $u_\theta = 0$. Hence on the surface of the cylinder, the resultant velocity is zero, when $\theta = 0°$ or $180°$. These two points on the surface of cylinder [*i.e.,* at $\theta = 0°$ and $180°$] where resultant velocity is zero, are known as stagnation points. They are denoted by S_1 and S_2. Stagnation point S_1 corresponds to $\theta = 180°$ and S_2 corresponds to $\theta = 0°$.

Pressure distribution on the surface of the cylinder

Let p_0 = pressure at a point in the uniform flow far away from the cylinder and towards the left of the cylinder [*i.e.,* approaching uniform flow]

U = velocity of uniform flow at that point

p = pressure at a point on the surface of the cylinder

V = resultant velocity at that point on the surface of the cylinder. This velocity is equal to u_θ as u_r is zero on the surface of the cylinder.

$\therefore \qquad V = u_\theta = -2U \sin\theta$

Applying Bernoulli's equation at the above two points,

$$\frac{p_0}{\rho g} + \frac{U^2}{2g} = \frac{p}{\rho g} + \frac{V^2}{2g}$$

or $$\frac{p_0}{\rho g} + \frac{U^2}{2g} = \frac{p}{\rho g} + \frac{[-2U\sin\theta]^2}{2g} \qquad [\because \quad V = u_\theta = -2U \sin\theta]$$

or $$\frac{p_0}{\rho} + \frac{U^2}{2} = \frac{p}{\rho} + \frac{4U^2 \sin^2\theta}{2}$$

or $$\frac{p - p_0}{\rho} = \frac{U^2}{2} - \frac{4U^2 \sin^2\theta}{2} = \frac{1}{2} U^2 (1 - 4\sin^2\theta)$$

or $$\frac{p - p_0}{\frac{1}{2}\rho U^2} = 1 - 4\sin^2\theta$$

But $\dfrac{p - p_0}{\frac{1}{2}\rho U^2}$ is a dimensionless term and is known as dimensionless pressure co-efficient and is denoted by C_p.

$\therefore$ $$C_p = \frac{p - p_0}{\frac{1}{2}\rho U^2} = 1 - 4\sin^2\theta$$

Value of pressure co-efficient for different values of θ

Value of θ	*Value of C_p*
0	$1 - 4\sin^2\theta = 1 - 0 = 1$
30°	$1 - 4\sin^2 30° = 1 - 4 \times \left(\frac{1}{2}\right)^2 = 1 - \frac{4}{4} = 1 - 1 = 0$
90°	$1 - 4\sin^2 90° = 1 - 4 \times 1 = 1 - 4 = -3$
150°	$1 - 4\sin^2 150° = 1 - 4 \times \frac{1}{4} = 1 - 1 = 0$
180°	$1 - 4\sin^2 180° = 1 - 0 = 1$

At $\theta = 0$ and 180°, there are stagnation points S_2 and S_1 respectively.

At $\theta = 30°$ and 150°, the pressure co-efficient is zero.

At $\theta = 90°$, the pressure co-efficient is -3 (*i.e.*, least pressure)

The variation of pressure co-efficient along the surface of the cylinder for different values of θ are shown in Fig. 5.54.

The positive pressure is acting normal to the surface and towards the surface of the cylinder whereas the negative pressure is acting normal to the surface and away from the surface of the cylinder as shown in Fig. 5.55.

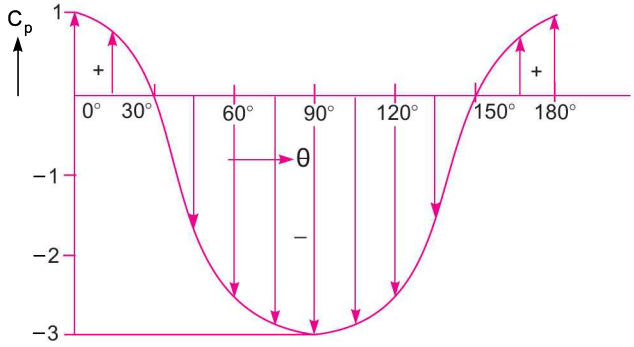


Fig. 5.54

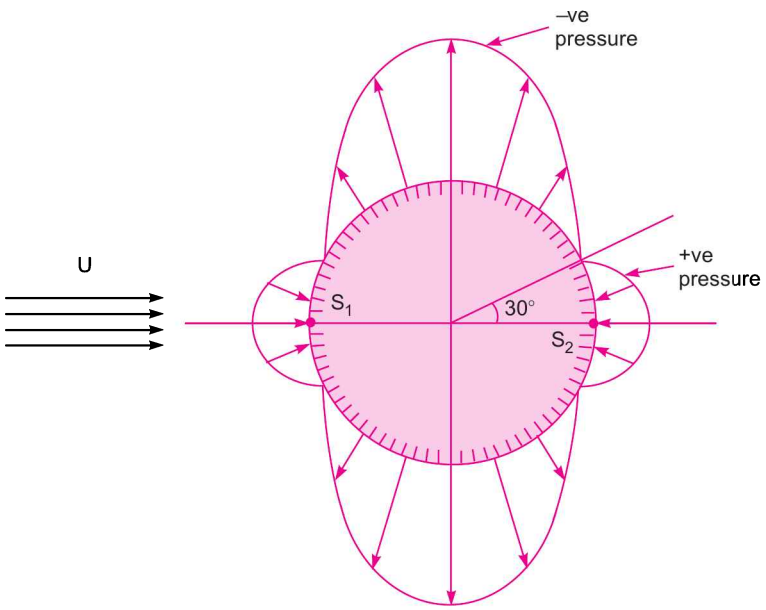


Fig. 5.55

Problem 5.43 *A uniform flow of 12 m/s is flowing over a doublet of strength 18 m^2/s. The doublet is in the line of the uniform flow. Determine :*

(i) shape of the Rankine oval (ii) radius of the Rankine circle

(iii) value of stream line function at Rankine circle

(iv) resultant velocity at a point on the Rankine circle at an angle of 30° from x-axis

(v) value of maximum velocity on the Rankine circle and location of the point where velocity is max.

Solution. Given : $U = 12$ m/s ; $\mu = 18$ m^2/s

(i) Shape of the Rankine oval

When a uniform flow is flowing over a doublet and doublet and uniform flow are in line, then the shape of the Rankine oval will be a circle of radius = $\sqrt{\frac{\mu}{2\pi U}}$. **Ans.**

(*ii*) *Radius of the Rankine circle*

$$R = r = \sqrt{\frac{\mu}{2\pi U}} = \sqrt{\frac{18}{2\pi \times 12}} = \textbf{0.488 m. Ans.}$$

(*iii*) *Value of stream line function at the Rankine circle*

The value of stream line function (ψ) at the Rankine circle is zero *i.e.*, $\psi = 0$.

(*iv*) *Resultant velocity on the surface of the circle, when* $\theta = 30°$

On the surface of the cylinder, the radial velocity (u_r) is zero. The tangential velocity (u_θ) is given by equation (5.73) as

$$u_\theta = -2U \sin\theta = -2 \times 12 \times \sin 30° = \textbf{– 12 m/s. Ans.}$$

–ve sign shows the clockwise direction of tangential velocity at that point.

$\therefore$ Resultant velocity, $V = \sqrt{u_r^2 + u_\theta^2} = \sqrt{0^2 + (-12)^2} = \textbf{12 m/s. Ans.}$

(*v*) *Maximum velocity and its location*

The resultant velocity at any point on the surface of the cylinder is equal to u_θ. But u_θ is given by,

$$u_\theta = -2U \sin\theta$$

This velocity will be maximum, when $\theta = 90°$.

$\therefore$ Max. velocity $= -2U = -2 \times 12 = \textbf{– 24 m/s. Ans.}$

Problem 5.44 *A uniform flow of 10 m/s is flowing over a doublet of strength 15 m²/s. The doublet is in the line of the uniform flow. The polar co-ordinates of a point P in the flow field are 0.9 m and 30°. Find : (i) stream line function and (ii) the resultant velocity at the point.*

Solution. Given : $U = 10$ m/s ; $\mu = 15$ m²/s ; $r = 0.9$ m and $\theta = 30°$.

Let us first find the radius (R) of the Rankine circle. This is given by

$$R = \sqrt{\frac{\mu}{2\pi U}} = \sqrt{\frac{15}{2\pi \times 10}} = 0.488 \text{ m}$$

The polar co-ordinates of the point P are 0.9 m and 30°.

Hence $r = 0.9$ m and $\theta = 30°$.

As the value of r is more than the radius of the Rankine circle, hence point P lies outside the cylinder.

(*i*) *Value of stream line function at the point P*

The stream line function for the composite flow at any point is given by equation (5.69) as

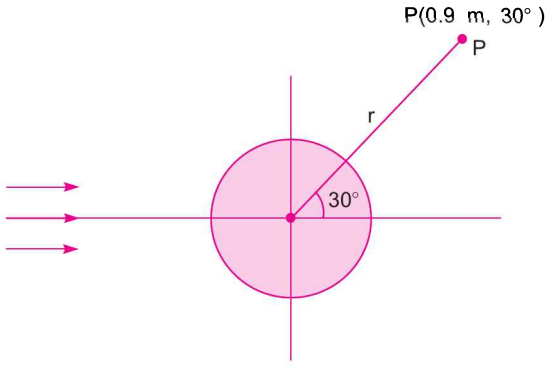


Fig. 5.56

$$\psi = U\left(r - \frac{R^2}{r}\right)\sin\theta$$

$$= 10\left(0.9 - \frac{0.488^2}{0.9}\right)\sin 30°\ (\because\ r = 0.9\text{ m}, R = 0.488 \text{ and } \theta = 30°)$$

$$= 10(0.9 - 0.2646) \times \frac{1}{2} = \mathbf{3.177\ m^2/s.\ Ans.}$$

(*ii*) *Resultant velocity at the point P*

The radial velocity and tangential velocity at any point in the flow field are given by equations (5.70) and (5.71) respectively.

$$\therefore \qquad u_r = U\left(1 - \frac{R^2}{r^2}\right)\cos\theta = 10\left(1 - \frac{0.488^2}{0.9^2}\right)\cos 30° = 611\text{ m/s}$$

+ve sign shows the radial velocity is outward.

and $$u_\theta = -U\left(1 + \frac{R^2}{r^2}\right)\sin\theta = -10\left(1 + \frac{0.488^2}{0.9^2}\right)\sin 30° = -6.47\text{ m/s}$$

–ve sign shows the clockwise direction of tangential velocity.

∴ Resultant velocity,

$$V = \sqrt{u_r^2 + u_\theta^2}$$

$$= \sqrt{6.11^2 + (-6.47)^2} = \sqrt{37.33 + 44.86}$$

$$= \mathbf{8.89\ m/s.\ Ans.}$$

HIGHLIGHTS

1. If the fluid characteristics like velocity, pressure, density etc. do not change at a point with respect to time, the fluid flow is called steady flow. If they change w.r.t. time, the fluid flow is called unsteady flow.

   Or $\left(\frac{\partial v}{\partial t}\right) = 0$ for steady flow and $\left(\frac{\partial v}{\partial t}\right) \neq 0$ for unsteady flow.

2. If the velocity in a fluid flow does not change with respect to space (length of direction of flow), the flow is said uniform otherwise non-uniform. Thus,

   $\left(\frac{\partial v}{\partial s}\right) = 0$ for uniform flow and $\left(\frac{\partial v}{\partial s}\right) \neq 0$ for non-uniform flow.

3. If the Reynolds number in a pipe is less than 2000, the flow is said to be laminar and if Reynold number is more than 4000, the flow is said to be turbulent.
4. For compressible flow, $\rho \neq$ constant
   For incompressible flow, $\rho =$ constant.
5. Rate of discharge for incompressible fluid (liquid), $Q = A \times v$.
6. Continuity equation is written as $A_1v_1 = A_2v_2 = A_3v_3$.

7. Continuity equation in differential form,

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z} = 0 \text{ for three-dimensional flow}$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y} = 0 \text{ for two-dimensional flow.}$$

8. The components of acceleration in x, y and z direction are

$$a_x = u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}+\frac{\partial u}{\partial t}$$

$$a_y = u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}+\frac{\partial v}{\partial t}$$

$$a_z = u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}+\frac{\partial w}{\partial t}.$$

9. The components of velocity in x, y and z direction in terms of velocity potential (ϕ) are

$$u = -\frac{\partial \phi}{\partial x},\ v = -\frac{\partial \phi}{\partial y} \text{ and } w = -\frac{\partial \phi}{\partial z}.$$

10. The stream function (ψ) is defined only for two-dimensional flow. The velocity components in x and y directions in terms of stream function are $u = -\frac{\partial \psi}{\partial y}$ and $v = \frac{\partial \psi}{\partial x}$.

11. Angular deformation or shear strain rate is given as

$$\text{Shear strain rate} = \frac{1}{2}\left[\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right]$$

12. Rotational components of a fluid particle are

$$\omega_z = \frac{1}{2}\left[\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right];\ \omega_x = \frac{1}{2}\left[\frac{\partial w}{\partial y}-\frac{\partial v}{\partial z}\right];\ \omega_y = \frac{1}{2}\left[\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x}\right]$$

13. Vorticity is two times the value of rotation.

14. Flow of a fluid along a curved path is known as vortex flow. If the particles are moving round in curved path with the help of some external torque the flow is called forced vortex flow. And if no external torque is required to rotate the fluid particles, the flow is called free-vortex flow.

15. The relation between tangential velocity and radius :

for forced vortex, $v = \omega \times r$,

for free vortex, $v \times r =$ constant.

16. The pressure variation along the radial direction for vortex flow along a horizontal plane, $\frac{\partial p}{\partial r} = \rho\frac{v^2}{r}$

and pressure variation in the vertical plane $\frac{\partial p}{\partial z} = -\rho g$.

17. For the forced vortex flow, $Z = \frac{v^2}{2g} = \frac{\omega^2 r^2}{2g} = \frac{\omega^2 R^2}{2g}$

where Z = height of paraboloid formed

ω = angular velocity.

18. For a forced vortex flow in a open tank.
Fall of liquid level at centre = Rise of liquid level at the ends.

19. In case of closed cylinder, the volume of air before rotation is equal to the volume of air after rotation.

20. If a close cylindrical vessel completely filled with water is rotated about its vertical axis, the total pressure forces acting on the top and bottom are

$$F_T = \frac{\rho}{4}\,\omega^2 \pi R^4$$

and $F_B = F_T$ + weight of water in cylinder

where F_T = Pressure force on top of cylinder
F_B = Pressure force on the bottom of cylinder
ω = Angular velocity
R = Radius of the vessel
ρ = Density of fluid = $\frac{w}{g}$.

21. For a free vortex flow the equation is $\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2$.

EXERCISE

(A) THEORETICAL PROBLEMS

1. What are the methods of describing fluid flow ?

2. Explain the terms :
(*i*) Path line, (*ii*) Streak line,
(*iii*) Stream line, and (*iv*) Stream tube.

3. Distinguish between :
(*i*) Steady flow and un-steady flow, (*ii*) Uniform and non-uniform flow,
(*iii*) Compressible and incompressible flow,
(*iv*) Rotational and irrotational flow, (*v*) Laminar and turbulent flow.

4. Define the following and give one practical example for each :
(*i*) Laminar flow, (*ii*) Turbulent flow,
(*iii*) Steady flow, and (*iv*) Uniform flow.

5. Define the equation of continuity. Obtain an expression for continuity equation for a three-dimensional flow. *(R.G.P.V, S 2002)*

6. What do you understand by the terms : (*i*) Total acceleration, (*ii*) Convective acceleration, and (*iii*) Local acceleration ? *(Delhi University, Dec. 2002)*

7. (*a*) Define the terms :
(*i*) Velocity potential function, and (*ii*) Stream function.
(*b*) What are the conditions for flow to be irrotational ?

8. What do you mean by equipotential line and a line of constant stream function ?

9. (*a*) Describe the use and limitations of the flow nets.
(*b*) Under what conditions can one draw flow net ?

10. Define the terms :
(*i*) Vortex flow, (*ii*) Forced vortex flow, and (*iii*) Free vortex flow.

11. Differentiate between forced vortex and free vortex flow.

12. Derive an expression for the depth of paraboloid formed by the surface of a liquid contained in a cylindrical tank which is rotated at a constant angular velocity ω about its vertical axis.

13. Derive an expression for the difference of pressure between two points in a free vortex flow. Does the difference of pressure satisfy Bernoulli's equation ? Can Bernoulli's equation be applied to a forced vortex flow ?

14. Derive, from first principles, the condition for irrotational flow. Prove that, for potential flow, both the stream function and velocity potential function satisfy the Laplace equation.

15. Define velocity potential function and stream function.

16. Under what conditions can one treat real fluid flow as irrotational (as an approximation).

17. Define the following :

(*i*) Steady flow, (*ii*) Non-uniform flow,
(*iii*) Laminar flow, and (*iv*) Two-dimensional flow.

18. (*a*) Distinguish between rotational flow and irrotational flow. Give one example of each

(*b*) Cite two examples of unsteady, non-uniform flow. How can the unsteady flow be transformed to steady flow ? *(J.N.T. University, S 2002)*

19. Explain uniform flow with source and sink. Obtain expressions for stream and velocity potential functions.

20. A point source is a point where an incompressible fluid is imagined to be created and sent out evenly in all directions. Determine its velocity potential and stream function.

21. (*i*) Explain doublet and define the strength of the doublet

(*ii*) Distinguish between a source and a sink.

22. Sketch the flow pattern of an ideal fluid flow past a cylinder with circulation.

23. Show that in case of forced vortex flow, the rise of liquid level at the ends is equal to the fall of liquid level at the axis of rotation.

24. Differentiate between :

(*i*) Stream function and velocity potential function
(*ii*) Stream line and streak line and
(*iii*) Rotational and irrotational flows.

(B) NUMERICAL PROBLEMS

1. The diameters of a pipe at the sections 1 and 2 are 15 cm and 20 cm respectively. Find the discharge through the pipe if velocity of water at section 1 is 4 m/s. Determine also the velocity at section 2.

[**Ans.** 0.07068 m^3/s, 2.25 m/s]

2. A 40 cm diameter pipe, conveying water, branches into two pipes of diameters 30 cm and 20 cm respectively. If the average velocity in the 40 cm diameter pipe is 3 m/s. Find the discharge in this pipe. Also determine the velocity in 20 cm pipe if the average velocity in 30 cm diameter pipe is 2 m/s.

[**Ans.** 0.3769 m^3/s, 7.5 m/s]

3. A 30 cm diameter pipe carries oil of sp. gr. 0.8 at a velocity of 2 m/s. At another section the diameter is 20 cm. Find the velocity at this section and also mass rate of flow of oil. [**Ans.** 4.5 m/s, 113 kg/s]

4. The velocity vector in a fluid flow is given by $V = 2x^3\mathbf{i} - 5x^2y\mathbf{j} + 4t\mathbf{k}$.
Find the velocity and acceleration of a fluid particle at (1, 2, 3) at time, $t = 1$.

[**Ans.** 10.95 units, 16.12 units]

5. The following cases represent the two velocity components, determine the third component of velocity such that they satisfy the continuity equation :

(*i*) $u = 4x^2, v = 4xyz$ (*ii*) $u = 4x^2 + 3xy, w = z^3 - 4xy - 2yz$.

[**Ans.** (*i*) $w = -8xz - 2xz^2 + f(x, y)$ (*ii*) $v = -8xy - \frac{y^2}{2} + 3yz^2 + f(x, z)$]

Calculate the unknown velocity components so that they satisfy the following equations :
(*i*) $u = 2x^2, v = 2xyz, w = ?$ (*ii*) $u = 2x^2 + 2xy, w = z^3 - 4xz + 2yz, v = ?$ [**Ans.** (*i*) $w = -1xz - x^2z$]

6. A fluid flow is given by : $V = xy^2i - 2yz^2j - \left(zy^2 - \frac{2z^3}{3}\right) k$.

Prove that it is a case of possible steady incompressible fluid flow.
Calculate the velocity and acceleration at the point [1, 2, 3]. [**Ans.** 36.7 units, 874.50 units]

7. Find the convective acceleration at the middle of a pipe which converges uniformly from 0.6 m diameter to 0.3 m diameter over 3 m length. The rate of flow is 40 lit/s. If the rate of flow changes uniformly from 40 lit/s to 80 lit/s in 40 seconds, find the total acceleration at the middle of the pipe at 20th second.
[**Ans.** .0499 m/s^2 ; .11874 m/s^2]

8. The velocity potential function, ϕ, is given by $\phi = x^2 - y^2$. Find the velocity components in x and y direction. Also show that ϕ represents a possible case of fluid flow. [**Ans.** $u = 2x$ and $v = -2y$]

9. For the velocity potential function, $\phi = x^2 - y^2$, find the velocity components at the point (4, 5).
[**Ans.** $u = 8$, $v = -10$ units]

10. A stream function is given by : $\psi = 2x - 5y$. Calculate the velocity components and also magnitude and direction of the resultant velocity at any point. [**Ans.** $u = 5$, $v = 2$, Resultant = 5.384 and $\theta = 21° 48'$]

11. If for a two-dimensional potential flow, the velocity potential is given by : $\phi = 4x(3y - 4)$, determine the velocity at the point (2, 3). Determine also the value of stream function ψ at the point (2, 3).

$\left[\text{**Ans.** } 40 \text{ units}, \psi = 6x^2 - 4\left(\frac{3}{2}y^2 - 4y\right), -18\right]$

12. The stream function for a two-dimensional flow is given by $\psi = 8xy$, calculate the velocity at the point $p(4, 5)$. Find the velocity potential function ϕ. [**Ans.** $u = -32$ units, $v = 40$ units, $\phi = 4y^2 - 4x^2$]

13. Sketch the stream lines represented by $\psi = xy$. Also find out the velocity and its direction at point (2, 3).
[**Ans.** 3.60 units and $\theta = 56° 18.6'$ or $123° 42'$]

14. For the velocity components given as : $u = ay \sin xy$, $v = ax \sin xy$.
Obtain an expression for the velocity potential function. [**Ans.** $\phi = a \cos xy$]

15. A fluid flow is given by : $V = 10x^3i - 8x^3yj$.
Find the shear strain rate and state whether the flow is rotational or irrotational. [**Ans.** $-8xy$, rotational]

16. The velocity components in a two-dimensional flow are :

$$u = 8x^2y - \frac{8}{3}y^3 \text{ and } v = -8xy^3 + \frac{8}{3}x^3.$$

Show that these velocity components represent a possible case of an irrotational flow.

$\left[\text{**Ans.** } \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0,\ \omega_z = 0\right]$

17. An open circular cylinder of 20 cm diameter and 100 cm long contains water upto a height of 80 cm. It is rotated about its vertical axis. Find the speed of rotation when :
(*i*) no water spills, (*ii*) axial depth is zero. [**Ans.** (*i*) 267.51 r.p.m., (*ii*) 422.98 r.p.m.]

18. A cylindrical vessel 15 cm in diameter and 40 cm long is completely filled with water. The vessel is open at the top. Find the quantity of water left in the vessel, when it is rotated about its vertical axis with a speed of 300 r.p.m. [**Ans.** 4566.3 cm^2]

19. An open circular cylinder of 20 cm diameter and 120 cm long contains water upto a height of 80 cm. It is rotated about its vertical axis at 400 r.p.m. Find the difference in total pressure force (*i*) at the bottom of the cylinder, and (*ii*) at the sides of the cylinder due to rotation. [**Ans.** (*i*) 14.52 N, (*ii*) 2465.45 N]

20. A closed cylindrical vessel of diameter 15 cm and length 100 cm contains water upto a height of 80 cm. The vessel is rotated at a speed of 500 r.p.m. about its vertical axis. Find the height of paraboloid formed. [**Ans.** 56.06 cm]

21. For the data given in question 20, find the speed of rotation of the vessel, when axial depth is zero. [**Ans.** 891.7 r.p.m.]

22. If the cylindrical vessel of question 20, is rotated at 950 r.p.m. about its vertical axis, find the area uncovered at the base of the tank. [**Ans.** 20.4 cm^2]

23. A closed cylindrical vessel of diameter 20 cm and height 100 cm contains water upto a height of 70 cm. The air above the water surface is at a pressure of 78.48 kN/m^2. The vessel is rotated at a speed of 300 r.p.m. about its vertical axis. Find the pressure head at the bottom of the vessel ; (*a*) at the centre, and (*b*) at the edge. [**Ans.** (*a*) 8.4485 m (*b*) 8.9515 m]

24. A closed cylinder of diameter 30 cm and height 20 cm is completely filled with water. Calculate the total pressure force exerted by water on the top and bottom of the cylinder, if it is rotated about its vertical axis at 300 r.p.m. [**Ans.** $F_T = 392.4$ N, $F_B = 531$ N]

25. In a free cylindrical vortex flow of water, at a point at a radius of 150 mm the velocity and pressure are 5 m/s and 14.715 N/cm^2. Find the pressure at a radius of 300 mm. [**Ans.** 15.65 N/cm^2]

26. Do the following velocity components represent physically possible flows ?

$$u = x^2 + z^2 + 5,\ v = y^2 + z^2,\ w = 4xyz.$$

[**Ans.** No.]

27. State if the flow represented by $u = 3x + 4y$ and $v = 2x - 3y$ is rotational or irrotational. [**Ans.** Rotational]

28. A vessel, cylindrical in shape and closed at the top and bottom, contains water upto a height of 700 mm. The diameter of the vessel is 200 mm and length of vessel is 1.1 m. Find the speed of rotation of the vessel if the axial depth of water is zero.

29. Define rotational and irrotational flow. The stream function and velocity potential for a flow are given by :

$$\psi = 2xy,\ \phi = x^2 - y^2.$$

Show that the conditions of continuity and irrotational flow are satisfied.

30. For the steady incompressible flow, are the following values of u and v possible ?

(*i*) $u = 4xy + y^2,\ v = 6xy + 3x$ and (*ii*) $u = 2x^2 + y^2,\ v = -4xy$. [**Ans.** (*i*) No, (*ii*) Yes]

31. Define two-dimensional stream function and velocity potential. Show that following stream function :

$$\psi = 6x - 4y + 7xy + 9$$

represents an irrotational flow. Find its velocity potential. [**Ans.** $\phi = 4x + 6y - 3.5x^2 + 3.5y^2 + C$]

32. Check if $\phi = x^2 - y^2 + y$ represents the velocity potential for 2-dimensional irrotational flow. If it does, then determine the stream function ψ. [**Ans.** Yes, $\psi = -2xy + x$]

33. If stream function for steady flow is given by $\psi = (y^2 - x^2)$, determine whether the flow is rotational or irrotational. Then determine the velocity potential ϕ. [**Ans.** Irrotational, $\phi = -2xy + C$]

34. A pipe (1) 450 mm in diameter branches into two pipes (2) and (3) of diameters 300 mm and 200 mm respectively as shown in Fig. 5.57. If the average velocity in 450 mm diameter pipe is 3 m/s, find : (*i*) discharge through 450 mm dia. pipe and (*ii*) velocity in 200 mm diameter pipe if the average velocity in 300 mm pipe is 2.5 m/s. *(J.N.T.U., Hyderabad, S 2002)*

[**Hint.** Given : $d_1 = 450$ mm $= 0.45$ m, $d_2 = 300$ mm $= 0.3$ m
$d_3 = 200$ mm $= 0.2$ m, $V_1 = 3$ m/s, $V_2 = 2.5$ m/s

(*i*) $Q_1 = A_1 V_1 = \frac{\pi}{4}(0.45^2) \times 3 = \mathbf{0.477\ m^3/s.}$

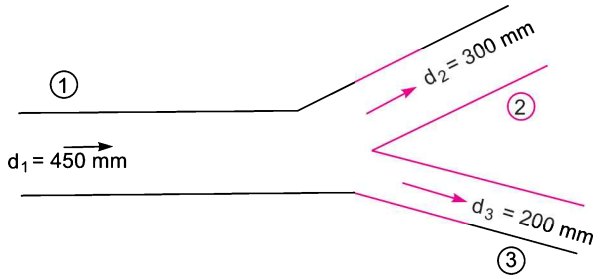


Fig. 5.57

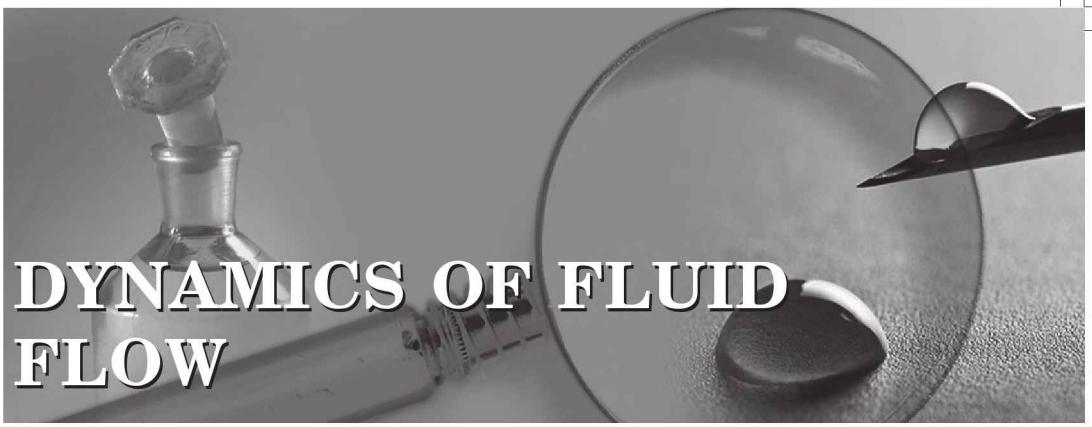
(*ii*) $Q_2 = A_2 V_2 = \frac{\pi}{4}(.3^2) \times 2.5 = 0.176\ \text{m}^3/\text{s}$

But $Q_1 = Q_2 + Q_3 \quad \therefore \quad Q_3 = Q_1 - Q_2 = 0.477 - 0.176 = 0.301$

Also $Q_3 = A_3 \times V_3 = \frac{\pi}{4}(0.2^2) \times V_3$

$\therefore$ $V_3 = \frac{Q_3}{\frac{\pi}{4}(0.2^2)} = \frac{0.301}{0.0314} = \mathbf{9.6\ m/s.}]$

6 CHAPTER



DYNAMICS OF FLUID FLOW

▶ 6.1 INTRODUCTION

In the previous chapter, we studied the velocity and acceleration at a point in a fluid flow, without taking into consideration the forces causing the flow. This chapter includes the study of forces causing fluid flow. Thus dynamics of fluid flow is the study of fluid motion with the forces causing flow. The dynamic behaviour of the fluid flow is analysed by the Newton's second law of motion, which relates the acceleration with the forces. The fluid is assumed to be incompressible and non-viscous.

▶ 6.2 EQUATIONS OF MOTION

According to Newton's second law of motion, the net force F_x acting on a fluid element in the direction of x is equal to mass m of the fluid element multiplied by the acceleration a_x in the x-direction. Thus mathematically,

$$F_x = m.a_x \qquad ...(6.1)$$

In the fluid flow, the following forces are present :

(*i*) F_g, gravity force.

(*ii*) F_p, the pressure force.

(*iii*) F_v, force due to viscosity.

(*iv*) F_t, force due to turbulence.

(*v*) F_c, force due to compressibility.

Thus in equation (6.1), the net force

$$F_x = (F_g)_x + (F_p)_x + (F_v)_x + (F_t)_x + (F_c)_x.$$

(*i*) If the force due to compressibility, F_c is negligible, the resulting net force

$$F_x = (F_g)_x + (F_p)_x + (F_v)_x + (F_t)_x$$

and equation of motions are called **Reynold's equations of motion.**

(*ii*) For flow, where (F_t) is negligible, the resulting equations of motion are known as **Navier-Stokes Equation.**

(*iii*) If the flow is assumed to be ideal, viscous force (F_v) is zero and equation of motions are known as **Euler's equation of motion.**

▶ 6.3 EULER'S EQUATION OF MOTION

This is equation of motion in which the forces due to gravity and pressure are taken into consideration. This is derived by considering the motion of a fluid element along a stream-line as :

Consider a stream-line in which flow is taking place in s-direction as shown in Fig. 6.1. Consider a cylindrical element of cross-section dA and length ds. The forces acting on the cylindrical element are:

1. Pressure force pdA in the direction of flow.
2. Pressure force $\left(p + \frac{\partial p}{\partial s} ds\right) dA$ opposite to the direction of flow.
3. Weight of element $\rho g dA ds$.

Let θ is the angle between the direction of flow and the line of action of the weight of element.

The resultant force on the fluid element in the direction of s must be equal to the mass of fluid element × acceleration in the direction s.

$$\therefore \quad pdA - \left(p + \frac{\partial p}{\partial s} ds\right) dA - \rho g dA ds \cos\theta = \rho dA ds \times a_s \qquad ...(6.2)$$

where a_s is the acceleration in the direction of s.

Now $a_s = \frac{dv}{dt}$, where v is a function of s and t.

$$= \frac{\partial v}{\partial s}\frac{ds}{dt} + \frac{\partial v}{\partial t} = \frac{v\partial v}{\partial s} + \frac{\partial v}{\partial t} \quad \left\{\because \frac{ds}{dt} = v\right\}$$

If the flow is steady, $\frac{\partial v}{\partial t} = 0$

$$\therefore \quad a_s = \frac{v\partial v}{\partial s}$$

Substituting the value of a_s in equation (6.2) and simplifying the equation, we get

$$-\frac{\partial p}{\partial s} dsdA - \rho g\, dAds \cos\theta = \rho dAds \times \frac{\partial v}{\partial s}$$

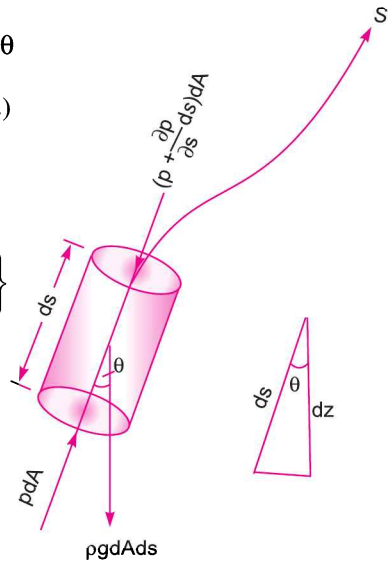


Fig. 6.1 *Forces on a fluid element.*

Dividing by $\rho dsdA$, $-\frac{\partial p}{\rho \partial s} - g\cos\theta = \frac{v\partial v}{\partial s}$

or $$\frac{\partial p}{\rho \partial s} + g\cos\theta + v\frac{\partial v}{\partial s} = 0$$

But from Fig. 6.1 (b), we have $\cos\theta = \frac{dz}{ds}$

$$\therefore \quad \frac{1}{\rho}\frac{dp}{ds} + g\frac{dz}{ds} + \frac{vdv}{ds} = 0 \quad \text{or} \quad \frac{dp}{\rho} + gdz + vdv = 0$$

or $$\frac{dp}{\rho} + gdz + vdv = 0 \qquad ...(6.3)$$

Equation (6.3) is known as Euler's equation of motion.

▶ 6.4 BERNOULLI'S EQUATION FROM EULER'S EQUATION

Bernoulli's equation is obtained by integrating the Euler's equation of motion (6.3) as

$$\int \frac{dp}{\rho} + \int g dz + \int v dv = \text{constant}$$

If flow is incompressible, ρ is constant and

$$\therefore \quad \frac{p}{\rho} + gz + \frac{v^2}{2} = \text{constant}$$

or

$$\frac{p}{\rho g} + z + \frac{v^2}{2g} = \text{constant}$$

or

$$\frac{p}{\rho g} + \frac{v^2}{2g} + z = \text{constant} \qquad ...(6.4)$$

Equation (6.4) is a Bernoulli's equation in which

$\frac{p}{\rho g}$ = pressure energy per unit weight of fluid or pressure head.

$v^2/2g$ = kinetic energy per unit weight or kinetic head.

z = potential energy per unit weight or potential head.

▶ 6.5 ASSUMPTIONS

The following are the assumptions made in the derivation of Bernoulli's equation :

(*i*) The fluid is ideal, *i.e.*, viscosity is zero (*ii*) The flow is steady

(*iii*) The flow is incompressible (*iv*) The flow is irrotational.

Problem 6.1 *Water is flowing through a pipe of 5 cm diameter under a pressure of 29.43 N/cm^2 (gauge) and with mean velocity of 2.0 m/s. Find the total head or total energy per unit weight of the water at a cross-section, which is 5 m above the datum line.*

Solution. Given :

Diameter of pipe = 5 cm = 0.5 m

Pressure, $p = 29.43 \text{ N/cm}^2 = 29.43 \times 10^4 \text{ N/m}^2$

Velocity, $v = 2.0$ m/s

Datum head, $z = 5$ m

Total head = pressure head + kinetic head + datum head

Pressure head $= \frac{p}{\rho g} = \frac{29.43 \times 10^4}{1000 \times 9.81} = 30$ m $\quad \left\{\rho \text{ for water} = 1000 \frac{\text{kg}}{\text{m}^3}\right\}$

Kinetic head $= \frac{v^2}{2g} = \frac{2 \times 2}{2 \times 9.81} = 0.204$ m

$\therefore$ Total head $= \frac{p}{\rho g} + \frac{v^2}{2g} + z = 30 + 0.204 + 5 =$ **35.204 m. Ans.**

Problem 6.2 *A pipe, through which water is flowing, is having diameters, 20 cm and 10 cm at the cross-sections 1 and 2 respectively. The velocity of water at section 1 is given 4.0 m/s. Find the velocity head at sections 1 and 2 and also rate of discharge.*

Solution. Given :

$$D_1 = 20 \text{ cm} = 0.2 \text{ m}$$

$\therefore$ Area, $A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} (.2)^2 = 0.0314 \text{ m}^2$

$V_1 = 4.0$ m/s

$D_2 = 0.1$ m

$\therefore$ $A_2 = \frac{\pi}{4} (.1)^2 = .00785 \text{ m}^2$

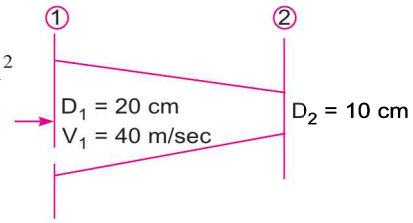


Fig. 6.2

(*i*) Velocity head at section 1

$$= \frac{V_1^2}{2g} = \frac{4.0 \times 4.0}{2 \times 9.81} = \textbf{0.815 m. Ans.}$$

(*ii*) Velocity head at section 2 = $V_2^2/2g$

To find V_2, apply continuity equation at 1 and 2

$\therefore$ $A_1V_1 = A_2V_2$ or $V_2 = \frac{A_1V_1}{A_2} = \frac{.0314}{.00785} \times 4.0 = 16.0$ m/s

$\therefore$ Velocity head at section 2 = $\frac{V_2^2}{2g} = \frac{16.0 \times 16.0}{2 \times 9.81} = \textbf{83.047 m. Ans.}$

(*iii*) Rate of discharge $= A_1V_1$ or A_2V_2

$= 0.0314 \times 4.0 = 0.1256 \text{ m}^3/\text{s}$

$= \textbf{125.6 litres/s. Ans.}$ $\{\because 1 \text{ m}^3 = 1000 \text{ litres}\}$

Problem 6.3 *State Bernoulli's theorem for steady flow of an incompressible fluid. Derive an expression for Bernoulli's equation from first principle and state the assumptions made for such a derivation.*

Solution. Statement of Bernoulli's Theorem. It states that in a steady, ideal flow of an incompressible fluid, the total energy at any point of the fluid is constant. The total energy consists of pressure energy, kinetic energy and potential energy or datum energy. These energies per unit weight of the fluid are :

Pressure energy = $\frac{p}{\rho g}$

Kinetic energy = $\frac{v^2}{2g}$

Datum energy = z

Thus mathematically, Bernoulli's theorem is written as

$$\frac{p}{\rho g} + \frac{v^2}{2g} + z = \text{Constant.}$$

Derivation of Bernoulli's theorem. For derivation of Bernoulli's theorem, Articles 6.3 and 6.4 should be written.

Assumptions are given in Article 6.5.

Problem 6.4 *The water is flowing through a pipe having diameters 20 cm and 10 cm at sections 1 and 2 respectively. The rate of flow through pipe is 35 litres/s. The section 1 is 6 m above datum and section 2 is 4 m above datum. If the pressure at section 1 is 39.24 N/cm^2, find the intensity of pressure at section 2.*

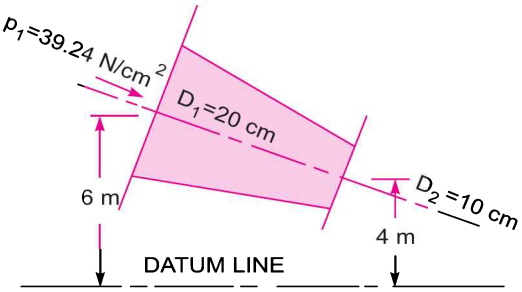


Fig. 6.3

Solution. Given :

At section 1, $D_1 = 20 \text{ cm} = 0.2 \text{ m}$

$$A_1 = \frac{\pi}{4}(.2)^2 = .0314 \text{ m}^2$$

$$p_1 = 39.24 \text{ N/cm}^2 = 39.24 \times 10^4 \text{ N/m}^2$$

$$z_1 = 6.0 \text{ m}$$

At section 2, $D_2 = 0.10 \text{ m}$

$$A_2 = \frac{\pi}{4}(0.1)^2 = .00785 \text{ m}^2$$

$$z_2 = 4 \text{ m}$$

$$p_2 = ?$$

Rate of flow, $$Q = 35 \text{ lit/s} = \frac{35}{1000} = .035 \text{ m}^3/\text{s}$$

Now $$Q = A_1V_1 = A_2V_2$$

$\therefore$ $$V_1 = \frac{Q}{A_1} = \frac{.035}{.0314} = 1.114 \text{ m/s}$$

and $$V_2 = \frac{Q}{A_2} = \frac{.035}{.00785} = 4.456 \text{ m/s}$$

Applying Bernoulli's equation at sections 1 and 2, we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2$$

or $$\frac{39.24 \times 10^4}{1000 \times 9.81} + \frac{(1.114)^2}{2 \times 9.81} + 6.0 = \frac{p_2}{1000 \times 9.81} + \frac{(4.456)^2}{2 \times 9.81} + 4.0$$

or $$40 + 0.063 + 6.0 = \frac{p_2}{9810} + 1.012 + 4.0$$

or $$46.063 = \frac{p_2}{9810} + 5.012$$

$\therefore$ $$\frac{p_2}{9810} = 46.063 - 5.012 = 41.051$$

$\therefore$ $$p_2 = 41.051 \times 9810 \text{ N/m}^2$$

$$= \frac{41.051 \times 9810}{10^4} \text{ N/cm}^2 = \mathbf{40.27 \text{ N/cm}^2}. \textbf{ Ans.}$$

Problem 6.5 *Water is flowing through a pipe having diameter 300 mm and 200 mm at the bottom and upper end respectively. The intensity of pressure at the bottom end is 24.525 N/cm^2 and the pressure at the upper end is 9.81 N/cm^2. Determine the difference in datum head if the rate of flow through pipe is 40 lit/s.*

Solution. Given :

Section 1, $D_1 = 300 \text{ mm} = 0.3 \text{ m}$

$p_1 = 24.525 \text{ N/cm}^2 = 24.525 \times 10^4 \text{ N/m}^2$

Section 2, $D_2 = 200 \text{ mm} = 0.2 \text{ m}$

$p_2 = 9.81 \text{ N/cm}^2 = 9.81 \times 10^4 \text{ N/m}^2$

Rate of flow $= 40$ lit/s

or $$Q = \frac{40}{1000} = 0.04 \text{ m}^3/\text{s}$$

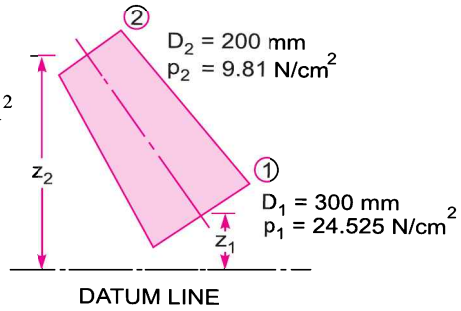


Fig. 6.4

Now $$A_1V_1 = A_2V_2 = \text{rate of flow} = 0.04$$

$\therefore$ $$V_1 = \frac{.04}{A_1} = \frac{.04}{\frac{\pi}{4}D_1^2} = \frac{0.04}{\frac{\pi}{4}(0.3)^2} = 0.5658 \text{ m/s}$$

$$\simeq 0.566 \text{ m/s}$$

$$V_2 = \frac{.04}{A_2} = \frac{.04}{\frac{\pi}{4}(D_2)^2} = \frac{0.04}{\frac{\pi}{4}(0.2)^2} = 1.274 \text{ m/s}$$

Applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2$$

or $$\frac{24.525 \times 10^4}{1000 \times 9.81} + \frac{.566 \times .566}{2 \times 9.81} + z_1 = \frac{9.81 \times 10^4}{1000 \times 9.81} + \frac{(1.274)^2}{2 \times 9.81} + z_2$$

or $$25 + .32 + z_1 = 10 + 1.623 + z_2$$

or $$25.32 + z_1 = 11.623 + z_2$$

$\therefore$ $$z_2 - z_1 = 25.32 - 11.623 = 13.697 = 13.70 \text{ m}$$

$\therefore$ Difference in datum head $= z_2 - z_1 =$ **13.70 m. Ans.**

Problem 6.6 *The water is flowing through a taper pipe of length 100 m having diameters 600 mm at the upper end and 300 mm at the lower end, at the rate of 50 litres/s. The pipe has a slope of 1 in 30. Find the pressure at the lower end if the pressure at the higher level is 19.62 N/cm^2.*

Solution. Given :

Length of pipe, $L = 100 \text{ m}$

Dia. at the upper end, $D_1 = 600 \text{ mm} = 0.6 \text{ m}$

$\therefore$ Area, $$A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} \times (.6)^2$$

$$= 0.2827 \text{ m}^2$$

$p_1 =$ pressure at upper end $= 19.62 \text{ N/cm}^2$

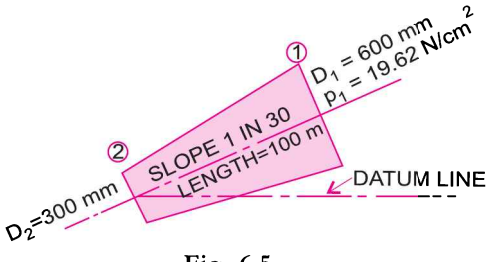


Fig. 6.5

$= 19.62 \times 10^4$ N/m^2

Dia. at lower end, $D_2 = 300$ mm $= 0.3$ m

$\therefore$ Area, $A_2 = \frac{\pi}{4} D_2^2 = \frac{\pi}{4} (.3)^2 = 0.07068$ m

Q = rate of flow = 50 litres/s $= \frac{50}{1000} = 0.05$ m^3/s

Let the datum line passes through the centre of the lower end.

Then $z_2 = 0$

As slope is 1 in 30 means $z_1 = \frac{1}{30} \times 100 = \frac{10}{3}$ m

Also we know $Q = A_1 V_1 = A_2 V_2$

$\therefore$ $V_1 = \frac{Q}{A} = \frac{0.05}{.2827} = 0.1768$ m/sec $= 0.177$ m/s

and $V_2 = \frac{Q}{A_2} = \frac{0.5}{.07068} = 0.7074$ m/sec $= 0.707$ m/s

Applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2$$

or $$\frac{19.62 \times 10^4}{1000 \times 9.81} + \frac{.177^2}{2 \times 9.81} + \frac{10}{3} = \frac{p_2}{\rho g} + \frac{.707^2}{2 \times 9.81} + 0$$

or $$20 + 0.001596 + 3.334 = \frac{p_2}{\rho g} + 0.0254$$

or $$23.335 - 0.0254 = \frac{p_2}{1000 \times 9.81}$$

or $p_2 = 23.3 \times 9810$ N/m^2 $= 228573$ N/m^2 $=$ **22.857 N/cm^2. Ans.**

▶ 6.6 BERNOULLI'S EQUATION FOR REAL FLUID

The Bernoulli's equation was derived on the assumption that fluid is inviscid (non-viscous) and therefore frictionless. But all the real fluids are viscous and hence offer resistance to flow. Thus there are always some losses in fluid flows and hence in the application of Bernoulli's equation, these losses have to be taken into consideration. Thus the Bernoulli's equation for real fluids between points 1 and 2 is given as

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2 + h_L \qquad ...(6.5)$$

where h_L is loss of energy between points 1 and 2.

Problem 6.7 *A pipe of diameter 400 mm carries water at a velocity of 25 m/s. The pressures at the points A and B are given as 29.43 N/cm^2 and 22.563 N/cm^2 respectively while the datum head at A and B are 28 m and 30 m. Find the loss of head between A and B.*

Solution. Given :

Dia. of pipe, $D = 400 \text{ mm} = 0.4 \text{ m}$

Velocity, $V = 25 \text{ m/s}$

At point *A*, $p_A = 29.43 \text{ N/cm}^2 = 29.43 \times 10^4 \text{ N/m}^2$

$z_A = 28 \text{ m}$

$v_A = v = 25 \text{ m/s}$

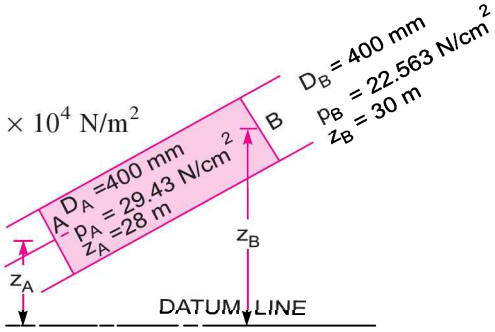


Fig. 6.6

$\therefore$ Total energy at *A*, $E_A = \dfrac{p_A}{\rho g} + \dfrac{v_A^2}{2g} + z_A$

$$= \frac{29.43 \times 10^4}{1000 \times 9.81} + \frac{25^2}{2 \times 9.81} + 28$$

$$= 30 + 31.85 + 28 = 89.85 \text{ m}$$

At point *B*, $p_B = 22.563 \text{ N/cm}^2 = 22.563 \times 10^4 \text{ N/m}^2$

$z_B = 30 \text{ m}$

$v_B = v = v_A = 25 \text{ m/s}$

$\therefore$ Total energy at *B*, $E_B = \dfrac{p_B}{\rho g} + \dfrac{v_B^2}{2g} + z_B$

$$= \frac{22.563 \times 10^4}{1000 \times 9.81} + \frac{25^2}{2 \times 9.81} + 30 = 23 + 31.85 + 30 = 84.85 \text{ m}$$

$\therefore$ Loss of energy $= E_A - E_B = 89.85 - 84.85 = \mathbf{5.0 \text{ m. Ans.}}$

Problem 6.8 *A conical tube of length 2.0 m is fixed vertically with its smaller end upwards. The velocity of flow at the smaller end is 5 m/s while at the lower end it is 2 m/s. The pressure head at the smaller end is 2.5 m of liquid. The loss of head in the tube is* $\dfrac{0.35(v_1 - v_2)^2}{2g}$, *where* v_1 *is the velocity at the smaller end and* v_2 *at the lower end respectively. Determine the pressure head at the lower end. Flow takes place in the downward direction.*

Solution. Let the smaller end is represented by (1) and lower end by (2)

Given :

Length of tube, $L = 2.0 \text{ m}$

$v_1 = 5 \text{ m/s}$

$p_1/\rho g = 2.5 \text{ m of liquid}$

$v_2 = 2 \text{ m/s}$

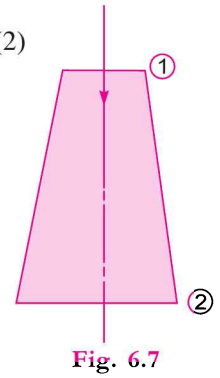


Fig. 6.7

Loss of head $= h_L = \dfrac{0.35(v_1 - v_2)^2}{2g}$

$$= \frac{0.35\,[5-2]^2}{2g} = \frac{0.35 \times 9}{2 \times 9.81} = 0.16 \text{ m}$$

Pressure head, $\frac{p_2}{\rho g} = ?$

Applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2 + h_L$$

Let the datum line passes through section (2). Then $z_2 = 0$, $z_1 = 2.0$

$$\therefore \quad 2.5 + \frac{5^2}{2 \times 9.81} + 2.0 = \frac{p_2}{\rho g} + \frac{2^2}{2 \times 9.81} + 0 + 0.16$$

$$2.5 + 1.27 + 2.0 = \frac{p_2}{\rho g} + 0.203 + .16$$

or $$\frac{p_2}{\rho g} = (2.5 + 1.27 + 2.0) - (.203 + .16)$$

$$= 5.77 - .363 = \textbf{5.407 m of fluid. Ans.}$$

Problem 6.9 *A pipeline carrying oil of specific gravity 0.87, changes in diameter from 200 mm diameter at a position A to 500 mm diameter at a position B which is 4 metres at a higher level. If the pressures at A and B are 9.81 N/cm^2 and 5.886 N/cm^2 respectively and the discharge is 200 litres/s determine the loss of head and direction of flow.*

Solution. Discharge, $Q = 200$ lit/s $= 0.2$ m^3/s

Sp. gr. of oil $= 0.87$

$\therefore$ ρ for oil $= .87 \times 1000 = 870 \frac{\text{kg}}{\text{m}^3}$

Given : **At section *A*,** $D_A = 200$ mm $= 0.2$ m

Area, $A_A = \frac{\pi}{4}(D_A)^2 = \frac{\pi}{4}(.2)^2$

$= 0.0314$ m^2

$p_A = 9.81$ N/cm^2

$= 9.81 \times 10^4$ N/m^2

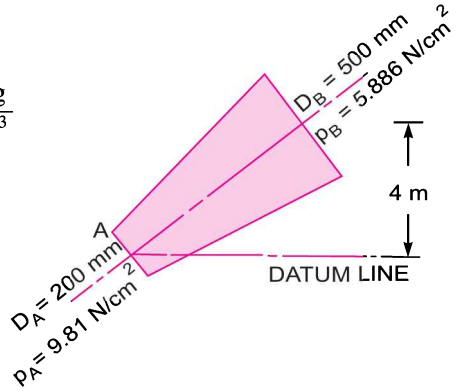


Fig. 6.8

If datum line is passing through A, then

$Z_A = 0$

$$V_A = \frac{Q}{A_A} = \frac{0.2}{0.0314} = 6.369 \text{ m/s}$$

At section *B*, $D_B = 500$ mm $= 0.50$ m

Area, $A_B = \frac{\pi}{4} D_B^2 = \frac{\pi}{4}(.5)^2 = 0.1963$ m^2

$p_B = 5.886$ N/cm^2 $= 5.886 \times 10^4$ N/m^2

$$Z_B = 4.0 \text{ m}$$

$$V_B = \frac{Q}{\text{Area}} = \frac{0.2}{.1963} = 1.018 \text{ m/s}$$

Total energy at A $$= E_A = \frac{p_A}{\rho g} + \frac{V_A^2}{2g} + Z_A$$

$$= \frac{9.81 \times 10^4}{870 \times 9.81} + \frac{(6.369)^2}{2 \times 9.81} + 0 = 11.49 + 2.067 = 13.557 \text{ m}$$

Total energy at B $$= E_B = \frac{p_B}{\rho g} + \frac{V_B^2}{2g} + Z_B$$

$$= \frac{5.886 \times 10^4}{870 \times 9.81} + \frac{(1.018)^2}{2 \times 9.81} + 4.0 = 6.896 + 0.052 + 4.0 = 10.948 \text{ m}$$

(*i*) **Direction of flow.** As E_A is more than E_B and hence flow is taking place from A to B. **Ans.**

(*ii*) Loss of head = $h_L = E_A - E_B = 13.557 - 10.948 =$ **2.609 m. Ans.**

▶ 6.7 PRACTICAL APPLICATIONS OF BERNOULLI'S EQUATION

Bernoulli's equation is applied in all problems of incompressible fluid flow where energy considerations are involved. But we shall consider its application to the following measuring devices :

1. Venturimeter.
2. Orifice meter.
3. Pitot-tube.

6.7.1 Venturimeter. A venturimeter is a device used for measuring the rate of a flow of a fluid flowing through a pipe. It consists of three parts :

(*i*) A short converging part, (*ii*) Throat, and (*iii*) Diverging part. It is based on the Principle of Bernoulli's equation.

Expression for rate of flow through venturimeter

Consider a venturimeter fitted in a horizontal pipe through which a fluid is flowing (say water), as shown in Fig. 6.9.

Let d_1 = diameter at inlet or at section (1),

p_1 = pressure at section (1)

v_1 = velocity of fluid at section (1),

a = area at section (1) = $\frac{\pi}{4} d_1^2$

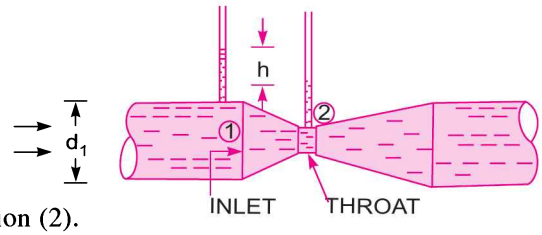


Fig. 6.9 *Venturimeter.*

and d_2, p_2, v_2, a_2 are corresponding values at section (2).

Applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2$$

As pipe is horizontal, hence $z_1 = z_2$

$$\therefore \quad \frac{p_1}{\rho g} + \frac{v_1^2}{2g} = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} \quad \text{or} \quad \frac{p_1 - p_2}{\rho g} = \frac{v_2^2}{2g} - \frac{v_1^2}{2g}$$

But $\frac{p_1 - p_2}{\rho g}$ is the difference of pressure heads at sections 1 and 2 and it is equal to h or $\frac{p_1 - p_2}{\rho g} = h$

Substituting this value of $\frac{p_1 - p_2}{\rho g}$ in the above equation, we get

$$h = \frac{v_2^2}{2g} - \frac{v_1^2}{2g} \qquad ...(6.6)$$

Now applying continuity equation at sections 1 and 2

$$a_1 v_1 = a_2 v_2 \quad \text{or} \quad v_1 = \frac{a_2 v_2}{a_1}$$

Substituting this value of v_1 in equation (6.6)

$$h = \frac{v_2^2}{2g} - \frac{\left(\frac{a_2 v_2}{a_1}\right)^2}{2g} = \frac{v_2^2}{2g}\left[1 - \frac{a_2^2}{a_1^2}\right] = \frac{v_2^2}{2g}\left[\frac{a_1^2 - a_2^2}{a_1^2}\right]$$

or $$v_2^2 = 2gh\,\frac{a_1^2}{a_1^2 - a_2^2}$$

$\therefore$ $$v_2 = \sqrt{2gh\frac{a_1^2}{a_1^2 - a_2^2}} = \frac{a_1}{\sqrt{a_1^2 - a_2^2}}\sqrt{2gh}$$

$\therefore$ Discharge, $$Q = a_2 v_2$$

$$= a_2 \frac{a_1}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh} = \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh} \qquad ...(6.7)$$

Equation (6.7) gives the discharge under ideal conditions and is called, theoretical discharge. Actual discharge will be less than theoretical discharge.

$\therefore$ $$Q_{act} = C_d \times \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh} \qquad ...(6.8)$$

where C_d = Co-efficient of venturimeter and its value is less than 1.

Value of 'h' given by differential U-tube manometer

Case I. Let the differential manometer contains a liquid which is heavier than the liquid flowing through the pipe. Let

S_h = Sp. gravity of the heavier liquid
S_o = Sp. gravity of the liquid flowing through pipe
x = Difference of the heavier liquid column in U-tube

Then $$h = x\left[\frac{S_h}{S_o} - 1\right] \qquad ...(6.9)$$

Case II. If the differential manometer contains a liquid which is lighter than the liquid flowing through the pipe, the value of h is given by

$$h = x\left[1 - \frac{S_l}{S_o}\right] \qquad ...(6.10)$$

where S_l = Sp. gr. of lighter liquid in U-tube

S_o = Sp. gr. of fluid flowing through pipe

x = Difference of the lighter liquid columns in U-tube.

Case III. Inclined Venturimeter with Differential U-tube manometer. The above two cases are given for a horizontal venturimeter. This case is related to inclined venturimeter having differential U-tube manometer. Let the differential manometer contains heavier liquid then h is given as

$$h = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = x\left[\frac{S_h}{S_o} - 1\right] \qquad ...(6.11)$$

Case IV. Similarly, for inclined venturimeter in which differential manometer contains a liquid which is lighter than the liquid flowing through the pipe, the value of h is given as

$$h = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = x\left[1 - \frac{S_l}{S_o}\right] \qquad ...(6.12)$$

Problem 6.10 *A horizontal venturimeter with inlet and throat diameters 30 cm and 15 cm respectively is used to measure the flow of water. The reading of differential manometer connected to the inlet and the throat is 20 cm of mercury. Determine the rate of flow. Take $C_d = 0.98$.*

Solution. Given :

Dia. at inlet, $d_1 = 30$ cm

$\therefore$ Area at inlet, $a_1 = \frac{\pi}{4} d_1^2 = \frac{\pi}{4}(30)^2 = 706.85 \text{ cm}^2$

Dia. at throat, $d_2 = 15$ cm

$\therefore$ $a_2 = \frac{\pi}{4} \times 15^2 = 176.7 \text{ cm}^2$

$C_d = 0.98$

Reading of differential manometer = x = 20 cm of mercury.

$\therefore$ Difference of pressure head is given by (6.9)

or $$h = x\left[\frac{S_h}{S_o} - 1\right]$$

where S_h = Sp. gravity of mercury = 13.6, S_o = Sp. gravity of water = 1

$$= 20\left[\frac{13.6}{1} - 1\right] = 20 \times 12.6 \text{ cm} = 252.0 \text{ cm of water.}$$

The discharge through venturimeter is given by eqn. (6.8)

$$Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$$

$$= 0.98 \times \frac{706.85 \times 176.7}{\sqrt{(706.85)^2 - (176.7)^2}} \times \sqrt{2 \times 9.81 \times 252}$$

$$= \frac{86067593.36}{\sqrt{499636.9 - 31222.9}} = \frac{86067593.36}{684.4}$$

$$= 125756 \text{ cm}^3/\text{s} = \frac{125756}{1000} \text{ lit/s} = \textbf{125.756 lit/s. Ans.}$$

Problem 6.11 *An oil of sp. gr. 0.8 is flowing through a venturimeter having inlet diameter 20 cm and throat diameter 10 cm. The oil-mercury differential manometer shows a reading of 25 cm. Calculate the discharge of oil through the horizontal venturimeter. Take C_d = 0.98.*

Solution. Given :

Sp. gr. of oil, $S_o = 0.8$

Sp. gr. of mercury, $S_h = 13.6$

Reading of differential manometer, $x = 25$ cm

$\therefore$ Difference of pressure head, $h = x\left[\frac{S_h}{S_o} - 1\right]$

$$= 25\left[\frac{13.6}{0.8} - 1\right] \text{ cm of oil} = 25\ [17 - 1] = 400 \text{ cm of oil.}$$

Dia. at inlet, $d_1 = 20$ cm

$\therefore$ $a_1 = \frac{\pi}{4} d_1^2 = \frac{\pi}{4} \times 20^2 = 314.16 \text{ cm}^2$

$d_2 = 10$ cm

$\therefore$ $a_2 = \frac{\pi}{4} \times 10^2 = 78.54 \text{ cm}^2$

$C_d = 0.98$

$\therefore$ The discharge Q is given by equation (6.8)

or $$Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$$

$$= 0.98 \times \frac{314.16 \times 78.54}{\sqrt{(314.16)^2 - (78.54)^2}} \times \sqrt{2 \times 981 \times 400}$$

$$= \frac{21421375.68}{\sqrt{98696 - 6168}} = \frac{21421375.68}{304} \text{ cm}^3/\text{s}$$

$$= 70465 \text{ cm}^3/\text{s} = \textbf{70.465 litres/s. Ans.}$$

Problem 6.12 *A horizontal venturimeter with inlet diameter 20 cm and throat diameter 10 cm is used to measure the flow of oil of sp. gr. 0.8. The discharge of oil through venturimeter is 60 litres/s. Find the reading of the oil-mercury differential manometer. Take C_d = 0.98.*

Solution. Given : $d_1 = 20$ cm

$\therefore$ $a_1 = \frac{\pi}{4} 20^2 = 314.16 \text{ cm}^2$

$d_2 = 10$ cm

$\therefore \quad a_2 = \frac{\pi}{4} \times 10^2 = 78.54 \text{ cm}^2$

$C_d = 0.98$

$Q = 60 \text{ litres/s} = 60 \times 1000 \text{ cm}^3/\text{s}$

Using the equation (6.8), $Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$

or $$60 \times 1000 = 9.81 \times \frac{314.16 \times 78.54}{\sqrt{(314.16)^2 - (78.54)^2}} \times \sqrt{2 \times 981 \times h} = \frac{1071068.78\sqrt{h}}{304}$$

or $$\sqrt{h} = \frac{304 \times 60000}{1071068.78} = 17.029$$

$\therefore \quad h = (17.029)^2 = 289.98 \text{ cm of oil}$

But $$h = x\left[\frac{S_h}{S_o} - 1\right]$$

where S_h = Sp. gr. of mercury = 13.6

S_o = Sp. gr. of oil = 0.8

x = Reading of manometer

$\therefore \quad 289.98 = x\left[\frac{13.6}{0.8} - 1\right] = 16x$

$\therefore \quad x = \frac{289.98}{16} = 18.12 \text{ cm.}$

$\therefore$ Reading of oil-mercury differential manometer = **18.12 cm. Ans.**

Problem 6.13 *A horizontal venturimeter with inlet diameter 20 cm and throat diameter 10 cm is used to measure the flow of water. The pressure at inlet is 17.658 N/cm^2 and the vacuum pressure at the throat is 30 cm of mercury. Find the discharge of water through venturimeter. Take C_d = 0.98.*

Solution. Given :

Dia. at inlet, $d_1 = 20 \text{ cm}$

$\therefore \quad a_1 = \frac{\pi}{4} \times (20)^2 = 314.16 \text{ cm}^2$

Dia. at throat, $d_2 = 10 \text{ cm}$

$\therefore \quad a_2 = \frac{\pi}{4} \times 10^2 = 78.74 \text{ cm}^2$

$p_1 = 17.658 \text{ N/cm}^2 = 17.658 \times 10^4 \text{ N/m}^2$

ρ for water $= 1000 \frac{\text{kg}}{\text{m}^3}$ and $\therefore \frac{p_1}{\rho g} = \frac{17.658 \times 10^4}{9.81 \times 1000} = 18 \text{ m of water}$

$\frac{p_2}{\rho g} = -30 \text{ cm of mercury}$

$= -0.30 \text{ m of mercury} = -0.30 \times 13.6 = -4.08 \text{ m of water}$

$\therefore$ Differential head $= h = \dfrac{p_1}{\rho g} - \dfrac{p_2}{\rho g} = 18 - (-4.08)$

$= 18 + 4.08 = 22.08$ m of water $= 2208$ cm of water

The discharge Q is given by equation (6.8)

$$Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$$

$$= 0.98 \times \frac{314.16 \times 78.54}{\sqrt{(314.16)^2 - (78.74)^2}} \times \sqrt{2 \times 981 \times 2208}$$

$$= \frac{50328837.21}{304} \times 165555 \text{ cm}^3/\text{s} = \textbf{165.555 lit/s. Ans.}$$

Problem 6.14 *The inlet and throat diameters of a horizontal venturimeter are 30 cm and 10 cm respectively. The liquid flowing through the meter is water. The pressure intensity at inlet is 13.734 N/cm^2 while the vacuum pressure head at the throat is 37 cm of mercury. Find the rate of flow. Assume that 4% of the differential head is lost between the inlet and throat. Find also the value of C_d for the venturimeter.*

Solution. Given :

Dia. at inlet, $d_1 = 30$ cm

$\therefore$ $a_1 = \dfrac{\pi}{4}(30)^2 = 706.85$ cm^2

Dia. at throat, $d_2 = 10$ cm

$\therefore$ $a_2 = \dfrac{\pi}{4}(10)^2 = 78.54$ cm^2

Pressure, $p_1 = 13.734$ N/cm^2 $= 13.734 \times 10^4$ N/m^2

$\therefore$ Pressure head, $\dfrac{p_1}{\rho g} = \dfrac{13.734 \times 10^4}{1000 \times 9.81} = 14$ m of water

$\dfrac{p_2}{\rho g} = -37$ cm of mercury

$= \dfrac{-37 \times 13.6}{100}$ m of water $= -5.032$ m of water

Differential head, $h = p_1/\rho g - p_2/\rho g$

$= 14.0 - (-5.032) = 14.0 + 5.032$

$= 19.032$ m of water $= 1903.2$ cm

Head lost, $h_f = 4\%$ of $h = \dfrac{4}{100} \times 19.032 = 0.7613$ m

$\therefore$ $C_d = \sqrt{\dfrac{h - h_f}{h}} = \sqrt{\dfrac{19.032 - .7613}{19.032}} = 0.98$

$\therefore$ Discharge $= C_d \dfrac{a_1 a_2 \sqrt{2gh}}{\sqrt{a_1^2 - a_2^2}}$

$$= \frac{0.98 \times 706.85 \times 78.54 \times \sqrt{2 \times 981 \times 1903.2}}{\sqrt{(706.85)^2 - (78.54)^2}}$$

$$= \frac{105132247.8}{\sqrt{499636.9 - 6168}} = 149692.8 \text{ cm}^3/\text{s} = \mathbf{0.14969\ m^3/s.\ Ans.}$$

PROBLEMS ON INCLINED VENTURIMETER

Problem 6.15 *A 30 cm × 15 cm venturimeter is inserted in a vertical pipe carrying water, flowing in the upward direction. A differential mercury manometer connected to the inlet and throat gives a reading of 20 cm. Find the discharge. Take C_d = 0.98.*

Solution. Given :

Dia. at inlet, $d_1 = 30$ cm

$\therefore$ $a_1 = \dfrac{\pi}{4}(30)^2 = 706.85 \text{ cm}^2$

Dia. at throat, $d_2 = 15$ cm

$\therefore$ $a_2 = \dfrac{\pi}{4}(15)^2 = 176.7 \text{ cm}^2$

$$h = x\left[\frac{S_h}{S_o} - 1\right] = 20\left[\frac{13.6}{1.0} - 1.0\right] = 20 \times 12.6 = 252.0 \text{ cm of water}$$

$C_d = 0.98$

Discharge, $Q = C_d \dfrac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$

$$= 0.98 \times \frac{706.85 \times 176.7}{\sqrt{(706.85)^2 - (176.7)^2}} \times \sqrt{2 \times 981 \times 252}$$

$$= \frac{86067593.36}{\sqrt{499636.3 - 31222.9}} = \frac{86067593.36}{684.4}$$

$= 125756 \text{ cm}^3/\text{s} = \mathbf{125.756\ lit/s.\ Ans.}$

Problem 6.16 *A 20 cm × 10 cm venturimeter is inserted in a vertical pipe carrying oil of sp. gr. 0.8, the flow of oil is in upward direction. The difference of levels between the throat and inlet section is 50 cm. The oil mercury differential manometer gives a reading of 30 cm of mercury. Find the discharge of oil. Neglect losses.*

Solution. Dia. at inlet, $d_1 = 20$ cm

$\therefore$ $a_1 = \dfrac{\pi}{4}(20)^2 = 314.16 \text{ cm}^2$

Dia. at throat, $d_2 = 10$ cm

$\therefore \quad a_2 = \frac{\pi}{4}(10)^2 = 78.54 \text{ cm}^2$

Sp. gr. of oil, $S_o = 0.8$

Sp. gr. of mercury, $S_g = 13.6$

Differential manometer reading, $x = 30$ cm

$\therefore \quad h = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = x\left[\frac{S_g}{S_o} - 1\right]$

$= 30\left[\frac{13.6}{0.8} - 1\right] = 30\ [17 - 1] = 30 \times 16 = 480$ cm of oil

$C_d = 1.0$

The discharge, $Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$

$= \frac{1.0 \times 314.16 \times 78.54}{\sqrt{(314.16)^2 - (78.54)^2}} \times \sqrt{2 \times 981 \times 480}$ cm^3/s

$= \frac{23932630.7}{304} = 78725.75$ cm^3/s $=$ **78.725 litres/s. Ans.**

Problem 6.17 *In a vertical pipe conveying oil of specific gravity 0.8, two pressure gauges have been installed at A and B where the diameters are 16 cm and 8 cm respectively. A is 2 metres above B. The pressure gauge readings have shown that the pressure at B is greater than at A by 0.981 N/cm^2. Neglecting all losses, calculate the flow rate. If the gauges at A and B are replaced by tubes filled with the same liquid and connected to a U-tube containing mercury, calculate the difference of level of mercury in the two limbs of the U-tube.*

Solution. Given :

Sp. gr. of oil, $S_o = 0.8$

$\therefore$ Density, $\rho = 0.8 \times 1000 = 800 \frac{\text{kg}}{\text{m}^3}$

Dia. at A, $D_A = 16$ cm $= 0.16$ m

$\therefore$ Area at A, $A_1 = \frac{\pi}{4}(.16)^2 = 0.0201$ m^2

Dia. at B, $D_B = 8$ cm $= 0.08$ m

$\therefore$ Area at B, $A_2 = \frac{\pi}{4}(.08)^2 = 0.005026$ m^2

(i) Difference of pressures, $p_B - p_A = 0.981$ N/cm^2

$= 0.981 \times 10^4$ N/m$^2 = \frac{9810\text{N}}{\text{m}^2}$

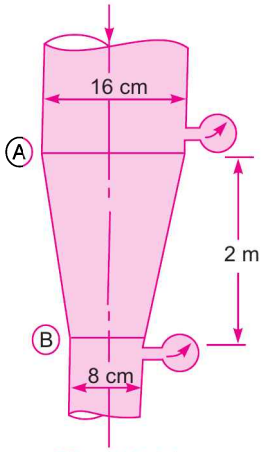


Fig. 6.9 (a)

Difference of pressure head

$\therefore \quad \frac{p_B - p_A}{\rho g} = \frac{9810}{800 \times 9.81} = 1.25$ ($\because$ $\rho = 800$ kg/m^3)

Applying Bernoulli's theorem at A and B and taking the reference line passing through section B, we get

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + Z_A = \frac{p_B}{\rho g} + \frac{V_B^2}{2g} + Z_B$$

or $$\frac{p_A}{\rho g} - \frac{p_B}{p_g} - Z_A - Z_B = \frac{V_B^2}{2g} - \frac{V_A^2}{2g}$$

or $$\left(\frac{p_A - p_B}{\rho g}\right) + 2.0 - 0.0 = \frac{V_B^2}{2g} - \frac{V_A^2}{2g}$$

or $$-1.25 + 2.0 = \frac{V_B^2}{2g} - \frac{V_A^2}{2g} \qquad \left(\because \frac{p_B - p_A}{\rho g} = 1.25\right)$$

$$0.75 = \frac{V_B^2}{2g} - \frac{V_A^2}{2g} \qquad ...(i)$$

Now applying continuity equation at A and B, we get

$$V_A \times A_1 = V_B \times A_2$$

or $$V_B = \frac{V_A \times A_1}{A_2} = \frac{V_A \times \frac{\pi}{4}(.16)^2}{\frac{\pi}{4}(.08)^2} = 4V_A$$

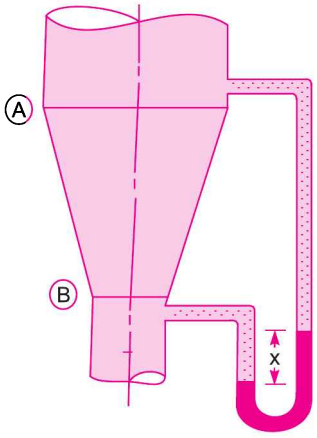


Fig. 6.9 (*b*)

Substituting the value of V_B in equation (*i*), we get

$$0.75 = \frac{16V_A^2}{2g} - \frac{V_A^2}{2g} = \frac{15V_A^2}{2g}$$

$$\therefore \quad V_A = \sqrt{\frac{0.75 \times 2 \times 9.81}{15}} = 0.99 \text{ m/s}$$

$\therefore$ Rate of flow, $Q = V_A \times A_1$

$= 0.99 \times 0.0201 =$ **0.01989 m^3/s. Ans.**

(*ii*) Difference of level of mercury in the U-tube.

Let h = Difference of mercury level.

Then $$h = x\left(\frac{S_g}{S_o} - 1\right)$$

where $$h = \left(\frac{p_A}{\rho g} + Z_A\right) - \left(\frac{p_B}{\rho g} + Z_B\right) = \frac{p_A - p_B}{\rho g} + Z_A - Z_B$$

$= -1.25 + 2.0 - 0$

$= 0.75$ $\qquad \left(\because \frac{p_B - p_A}{\rho g} = 1.25\right)$

$$\therefore \quad 0.75 = x\left[\frac{13.6}{0.8} - 1\right] = x \times 16$$

$$\therefore \quad x = \frac{0.75}{16} = 0.04687 \text{ m} = \textbf{4.687 cm. Ans.}$$

Problem 6.18 *Find the discharge of water flowing through a pipe 30 cm diameter placed in an inclined position where a venturimeter is inserted, having a throat diameter of 15 cm. The difference of pressure between the main and throat is measured by a liquid of sp. gr. 0.6 in an inverted U-tube which gives a reading of 30 cm. The loss of head between the main and throat is 0.2 times the kinetic head of the pipe.*

Solution. Dia. at inlet, $d_1 = 30$ cm

$$\therefore \quad a_1 = \frac{\pi}{4}(30)^2 = 706.85 \text{ cm}^2$$

Dia. at throat, $d_2 = 15$ cm

$$\therefore \quad a_2 = \frac{\pi}{4}(15)^2 = 176.7 \text{ cm}^2$$

Reading of differential manometer, $x = 30$ cm

Difference of pressure head, h is given by

$$\left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = h$$

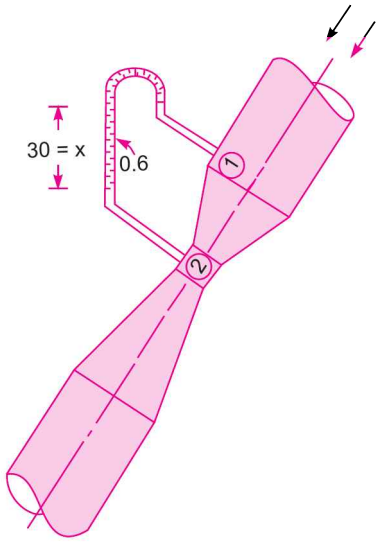


Fig. 6.10

Also $$h = x\left[1 - \frac{S_l}{S_o}\right]$$

where $S_l = 0.6$ and $S_o = 1.0$

$$= 30\left[1 - \frac{0.6}{1.0}\right] = 30 \times .4 = 12.0 \text{ cm of water}$$

Loss of head, $h_L = 0.2 \times$ kinetic head of pipe $= 0.2 \times \frac{v_1^2}{2g}$

Now applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + z_1 + \frac{v_1^2}{2g} = \frac{p_2}{\rho g} + z_2 + \frac{v_2^2}{2g} + h_L$$

or $$\left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) + \frac{v_1^2}{2g} - \frac{v_2^2}{2g} = h_L$$

But $$\left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = h = 12.0 \text{ cm of water}$$

and $$h_L = 0.2 \times v_1^2/2g$$

$$\therefore \quad 12.0 + \frac{v_1^2}{2g} - \frac{v_2^2}{2g} = 0.2 \times \frac{v_1^2}{2g}$$

$$\therefore \quad 12.0 + 0.8\,\frac{v_1^2}{2g} - \frac{v_2^2}{2g} = 0 \qquad \text{...(1)}$$

Applying continuity equation at sections (1) and (2), we get

$$a_1 v_1 = a_2 v_2$$

$$\therefore \quad v_1 = \frac{a_2}{a_1} v_2 = \frac{\frac{\pi}{4}(15)^2 v_2}{\frac{\pi}{4}(30)^2} = \frac{v_2}{4}$$

Substituting this value of v_1 in equation (1), we get

$$12.0 + \frac{0.8}{2g}\left(\frac{v_2}{4}\right)^2 - \frac{v_2^2}{2g} = 0 \text{ or } 12.0 + \frac{v_2^2}{2g}\left[\frac{0.8}{16} - 1\right] = 0$$

or $$\frac{v_2^2}{2g}[.05 - 1] = -12.0 \text{ or } \frac{0.95\, v_2^2}{2g} = 12.0$$

$$\therefore \quad v_2 = \sqrt{\frac{2 \times 981 \times 12.0}{0.95}} = 157.4 \text{ cm/s}$$

$\therefore$ Discharge $= a_2 v_2$

$= 176.7 \times 157.4 \text{ cm}^3/\text{s} = 27800 \text{ cm}^3/\text{s} =$ **27.8 litres/s. Ans.**

Problem 6.19 *A 30 cm × 15 cm venturimeter is provided in a vertical pipe line carrying oil of specific gravity 0.9, the flow being upwards. The difference in elevation of the throat section and entrance section of the venturimeter is 30 cm. The differential U-tube mercury manometer shows a gauge deflection of 25 cm. Calculate :*

(i) the discharge of oil, and

(ii) the pressure difference between the entrance section and the throat section. Take the co-efficient of discharge as 0.98 and specific gravity of mercury as 13.6.

Solution. Given :

Dia. at inlet, $d_1 = 30$ cm

$\therefore$ Area, $a_1 = \frac{\pi}{4}(30)^2 = 706.85 \text{ cm}^2$

Dia. at throat, $d_2 = 15$ cm

$\therefore$ Area, $a_2 = \frac{\pi}{4}(15)^2 = 176.7 \text{ cm}^2$

Let section (1) represents inlet and section (2) represents throat. Then $z_2 - z_1 = 30$ cm

Sp. gr. of oil, $S_o = 0.9$

Sp. gr. of mercury, $S_g = 13.6$

Reading of diff. manometer, $x = 25$ cm

The differential head, h is given by

$$h = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right)$$

$$= x\left[\frac{S_g}{S_o} - 1\right] = 25\left[\frac{13.6}{0.9} - 1\right] = 352.77 \text{ cm of oil}$$

(*i*) The discharge, Q of oil $= C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$

$$= \frac{0.98 \times 706.85 \times 176.7}{\sqrt{(706.85)^2 - (176.7)^2}} = \sqrt{2 \times 981 \times 352.77}$$

$$= \frac{101832219.9}{684.4} = 148790.5 \text{ cm}^3/\text{s}$$

$$= \mathbf{148.79 \text{ litres/s. Ans.}}$$

(*ii*) Pressure difference between entrance and throat section

$$h = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = 352.77$$

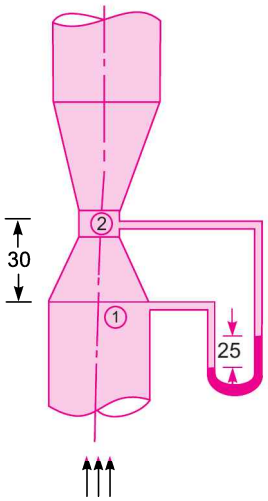


Fig. 6.11

or $$\left(\frac{p_1}{\rho g} - \frac{p_2}{\rho g}\right) + z_1 - z_2 = 352.77$$

But $$z_2 - z_1 = 30 \text{ cm}$$

$\therefore$ $$\left(\frac{p_1}{\rho g} - \frac{p_2}{\rho g}\right) - 30 = 352.77$$

$\therefore$ $$\frac{p_1}{\rho g} - \frac{p_2}{\rho g} = 352.77 + 30 = 382.77 \text{ cm of oil} = \mathbf{3.8277 \text{ m of oil. Ans.}}$$

or $$(p_1 - p_2) = 3.8277 \times \rho g$$

But density of oil $= \text{Sp. gr. of oil} \times 1000 \text{ kg/m}^3$

$$= 0.9 \times 1000 = 900 \text{ kg/cm}^3$$

$\therefore$ $$(p_1 - p_2) = 3.8277 \times 900 \times 9.81 \frac{\text{N}}{\text{m}^2}$$

$$= \frac{33795}{10^4} \text{ N/cm}^2 = \mathbf{3.3795 \text{ N/cm}^2. \text{ Ans.}}$$

Problem 6.20 *Crude oil of specific gravity 0.85 flows upwards at a volume rate of flow of 60 litre per second through a vertical venturimeter with an inlet diameter of 200 mm and a throat diameter of 100 mm. The co-efficient of discharge of the venturimeter is 0.98. The vertical distance between the pressure tappings is 300 mm.*

(i) If two pressure gauges are connected at the tappings such that they are positioned at the levels of their corresponding tapping points, determine the difference of readings in N/cm^2 of the two pressure gauges.

(ii) If a mercury differential manometer is connected, in place of pressure gauges, to the tappings such that the connecting tube upto mercury are filled with oil, determine the difference in the level of the mercury column.

Solution. Given :

Specific gravity of oil, $S_o = 0.85$

∴ Density, $\rho = 0.85 \times 1000 = 850 \text{ kg/m}^3$

Discharge, $Q = 60 \text{ litre/s}$

$$= \frac{60}{1000} = 0.06 \text{ m}^3/\text{s}$$

Inlet dia, $d_1 = 200 \text{ mm} = 0.2 \text{ m}$

∴ Area, $a_1 = \frac{\pi}{4}(.2)^2 = 0.0314 \text{ m}^2$

Throat dia., $d_2 = 100 \text{ mm} = 0.1 \text{ m}$

∴ Area, $a_2 = \frac{\pi}{4}(0.1)^2 = 0.00785 \text{ m}^2$

Value of C_d $= 0.98$

Let section (1) represents inlet and section (2) represents throat. Then

$$z_2 - z_1 = 300 \text{ mm} = 0.3 \text{ m}$$

(*i*) *Difference of readings in N/cm² of the two pressure gauges*

The discharge Q is given by,

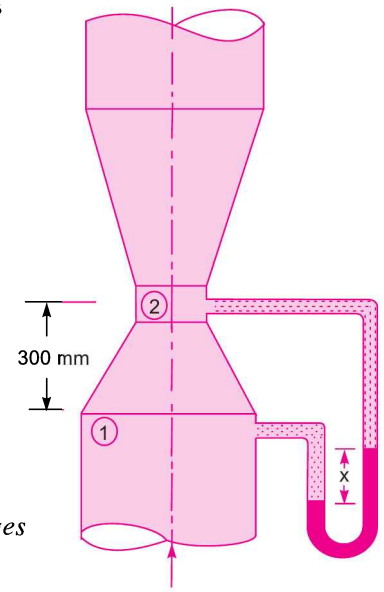


Fig. 6.11 (*a*)

$$Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$$

or
$$0.06 = \frac{0.98 \times 0.0314 \times 0.00785}{\sqrt{0.0314^2 - 0.00785^2}} \times \sqrt{2 \times 9.81 \times h}$$

$$= \frac{0.98 \times 0.00024649}{0.0304} \times 4.429 \sqrt{h}$$

∴
$$\sqrt{h} = \frac{0.06 \times 0.0304}{0.98 \times 0.00024649 \times 4.429} = 1.705$$

∴
$$h = 1.705^2 = 2.908 \text{ m}$$

But for a vertical venturimeter, $h = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right)$

∴
$$2.908 = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = \left(\frac{p_1}{\rho g} - \frac{p_2}{\rho g}\right) + z_1 - z_2$$

$$\frac{p_1 - p_2}{\rho g} = 2.908 + z_2 - z_1 = 2.908 + 0.3 \qquad (\because \quad z_2 - z_1 = 0.3 \text{ m})$$

$$= 3.208 \text{ m of oil}$$

∴
$$p_1 - p_2 = \rho g \times 3.208$$

$$= 850 \times 9.81 \times 3.208 \text{ N/m}^2 = \frac{850 \times 9.81 \times 3.208}{10^4} \text{ N/cm}^2$$

$$= \mathbf{2.675 \text{ N/cm}^2. \text{ Ans.}}$$

(*ii*) *Difference in the levels of mercury columns* (*i.e., x*)

The value of h is given by, $h = x\left[\frac{S_g}{S_o} - 1\right]$

$\therefore \quad 2.908 = x\left[\frac{13.6}{0.85} - 1\right] = x\,[16 - 1] = 15\,x$

$\therefore \quad x = \frac{2.908}{15} = 0.1938 \text{ m} = \mathbf{19.38 \text{ cm of oil. Ans.}}$

Problem 6.21 *In a 100 mm diameter horizontal pipe a venturimeter of 0.5 contraction ratio has been fixed. The head of water on the metre when there is no flow is 3 m (gauge). Find the rate of flow for which the throat pressure will be 2 metres of water absolute. The co-efficient of discharge is 0.97. Take atmospheric pressure head = 10.3 m of water.*

Solution. Given :

Dia. of pipe, $d_1 = 100 \text{ mm} = 10 \text{ cm}$

$\therefore$ Area, $a_1 = \frac{\pi}{4} d_1^2 = \frac{\pi}{4} (10)^2 = 78.54 \text{ cm}^2$

Dia. at throat, $d_2 = 0.5 \times d_1 = 0.5 \times 10 = 5 \text{ cm}$

$\therefore$ Area, $a_2 = \frac{\pi}{4} (5)^2 = 19.635 \text{ cm}^2$

Head of water for no flow $= \frac{p_1}{\rho g} = 3 \text{ m (gauge)} = 3 + 10.3 = 13.3 \text{ m (abs.)}$

Throat pressure head $= \frac{p_2}{\rho g} = 2 \text{ m of water absolute.}$

$\therefore$ Difference of pressure head, $h = \frac{p_1}{\rho g} - \frac{p_2}{\rho g} = 13.3 - 2.0 = 11.3 \text{ m} = 1130 \text{ cm}$

$\therefore$ Rate of flow, Q is given by $Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$

$$= 0.97 \times \frac{78.54 \times 19.635}{\sqrt{(78.54)^2 - (19.635)^2}} \times \sqrt{2 \times 981 \times 1130}$$

$$= \frac{2227318.17}{76} = 29306.8 \text{ cm}^3/\text{s} = \mathbf{29.306 \text{ litres/s. Ans.}}$$

6.7.2 Orifice Meter or Orifice Plate. It is a device used for measuring the rate of flow of a fluid through a pipe. It is a cheaper device as compared to venturimeter. It also works on the same principle as that of venturimeter. It consists of a flat circular plate which has a circular sharp edged hole called orifice, which is concentric with the pipe. The orifice diameter is kept generally 0.5 times the diameter of the pipe, though it may vary from 0.4 to 0.8 times the pipe diameter.

A differential manometer is connected at section (1), which is at a distance of about 1.5 to 2.0 times the pipe diameter upstream from the orifice plate, and at section (2), which is at a distance of about half the diameter of the orifice on the downstream side from the orifice plate.

Let p_1 = pressure at section (1),
v_1 = velocity at section (1),
a_1 = area of pipe at section (1), and

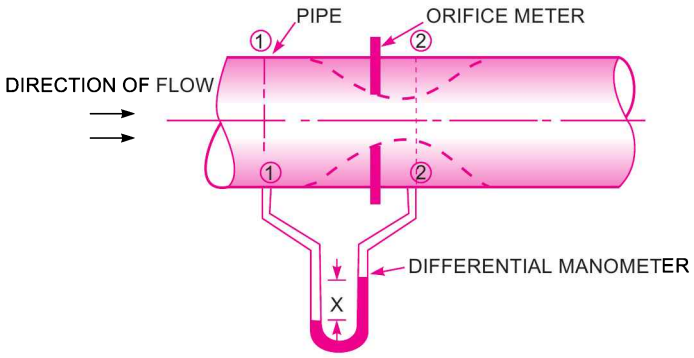


Fig. 6.12. *Orifice meter.*

p_2, v_2, a_2 are corresponding values at section (2). Applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2$$

or $$\left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = \frac{v_2^2}{2g} - \frac{v_1^2}{2g}$$

But $$\left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = h = \text{Differential head}$$

$\therefore$ $$h = \frac{v_2^2}{2g} - \frac{v_1^2}{2g} \quad \text{or} \quad 2gh = v_2^2 - v_1^2$$

or $$v_2 = \sqrt{2gh + v_1^2} \qquad ...(i)$$

Now section (2) is at the vena-contracta and a_2 represents the area at the vena-contracta. If a_0 is the area of orifice then, we have

$$C_c = \frac{a_2}{a_0}$$

where C_c = Co-efficient of contraction

$\therefore$ $$a_2 = a_0 \times C_c \qquad ...(ii)$$

By continuity equation, we have

$$a_1 v_1 = a_2 v_2 \quad \text{or} \quad v_1 = \frac{a_2}{a_1} v_2 = \frac{a_0 C_c}{a_1} v_2 \qquad ...(iii)$$

Substituting the value of v_1 in equation (i), we get

$$v_2 = \sqrt{2gh + \frac{a_0^2 C_c^2 v_2^2}{a_1^2}}$$

or $$v_2^2 = 2gh + \left(\frac{a_0}{a_1}\right)^2 C_c^2 v_2^2 \text{ or } v_2^2 \left[1 - \left(\frac{a_0}{a_1}\right)^2 C_c^2\right] = 2gh$$

$$\therefore \quad v_2 = \frac{\sqrt{2gh}}{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2 C_c^2}}$$

$\therefore$ The discharge $Q = v_2 \times a_2 = v_2 \times a_0 C_c$ $\{\because a_2 = a_0 C_c$ from $(ii)\}$

$$= \frac{a_0 C_c \sqrt{2gh}}{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2 C_c^2}} \qquad ...(iv)$$

The above expression is simplified by using

$$C_d = C_c \frac{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2}}{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2 C_c^2}}$$

$$\therefore \quad C_c = C_d \frac{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2 C_c^2}}{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2}}$$

Substituting this value of C_c in equation (iv), we get

$$Q = a_0 \times C_d \frac{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2 C_c^2}}{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2}} \times \frac{\sqrt{2gh}}{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2 C_c^2}}$$

$$= \frac{C_d a_0 \sqrt{2gh}}{\sqrt{1 - \left(\frac{a_0}{a_1}\right)^2}} = \frac{C_d a_0 a_1 \sqrt{2gh}}{\sqrt{a_1^2 - a_0^2}}. \qquad ...(6.13)$$

where C_d = Co-efficient of discharge for orifice meter.

The co-efficient of discharge for orifice meter is much smaller than that for a venturimeter.

Problem 6.22 *An orifice meter with orifice diameter 10 cm is inserted in a pipe of 20 cm diameter. The pressure gauges fitted upstream and downstream of the orifice meter gives readings of 19.62 N/cm^2 and 9.81 N/cm^2 respectively. Co-efficient of discharge for the orifice meter is given as 0.6. Find the discharge of water through pipe.*

Solution. Given :

Dia. of orifice, $d_0 = 10$ cm

$\therefore$ Area, $a_0 = \frac{\pi}{4}(10)^2 = 78.54 \text{ cm}^2$

Dia. of pipe, $d_1 = 20$ cm

$\therefore$ Area, $a_1 = \frac{\pi}{4}(20)^2 = 314.16 \text{ cm}^2$

$p_1 = 19.62 \text{ N/cm}^2 = 19.62 \times 10^4 \text{ N/m}^2$

$\therefore$ $\frac{p_1}{\rho g} = \frac{19.62 \times 10^4}{1000 \times 9.81} = 20$ m of water

Similarly $\frac{p_2}{\rho g} = \frac{9.81 \times 10^4}{1000 \times 9.81} = 10$ m of water

$\therefore$ $h = \frac{p_1}{\rho g} - \frac{p_2}{\rho g} = 20.0 - 10.0 = 10$ m of water $= 1000$ cm of water

$C_d = 0.6$

The discharge, Q is given by equation (6.13)

$$Q = C_d \frac{a_0 a_1}{\sqrt{a_1^2 - a_0^2}} \times \sqrt{2gh}$$

$$= 0.6 \times \frac{78.54 \times 314.16}{\sqrt{(314.16)^2 - (78.54)^2}} \times \sqrt{2 \times 981 \times 1000}$$

$$= \frac{20736838.09}{304} = 68213.28 \text{ cm}^3/\text{s} = \textbf{68.21 litres/s. Ans.}$$

Problem 6.23 *An orifice meter with orifice diameter 15 cm is inserted in a pipe of 30 cm diameter. The pressure difference measured by a mercury oil differential manometer on the two sides of the orifice meter gives a reading of 50 cm of mercury. Find the rate of flow of oil of sp. gr. 0.9 when the co-efficient of discharge of the orifice meter = 0.64.*

Solution. Given :

Dia. of orifice, $d_0 = 15$ cm

$\therefore$ Area, $a_0 = \frac{\pi}{4}(15)^2 = 176.7 \text{ cm}^2$

Dia. of pipe, $d_1 = 30$ cm

$\therefore$ Area, $a_1 = \frac{\pi}{4}(30)^2 = 706.85 \text{ cm}^2$

Sp. gr. of oil, $S_o = 0.9$

Reading of diff. manometer, $x = 50$ cm of mercury

$\therefore$ Differential head, $h = x\left[\frac{S_g}{S_o} - 1\right] = 50\left[\frac{13.6}{0.9} - 1\right]$ cm of oil

$= 50 \times 14.11 = 705.5$ cm of oil

$C_d = 0.64$

$\therefore$ The rate of the flow, Q is given by equation (6.13)

$$Q = C_d \cdot \frac{a_0 a_1}{\sqrt{a_1^2 - a_0^2}} \times \sqrt{2gh}$$

$$= 0.64 \times \frac{176.7 \times 706.85}{\sqrt{(706.85)^2 - (176.7)^2}} \times \sqrt{2 \times 981 \times 705.5}$$

$$= \frac{94046317.78}{684.4} = 137414.25 \text{ cm}^3/\text{s} = \textbf{137.414 litres/s. Ans.}$$

6.7.3 Pitot-tube. It is a device used for measuring the velocity of flow at any point in a pipe or a channel. It is based on the principle that if the velocity of flow at a point becomes zero, the pressure there is increased due to the conversion of the kinetic energy into pressure energy. In its simplest form, the pitot-tube consists of a glass tube, bent at right angles as shown in Fig. 6.13.

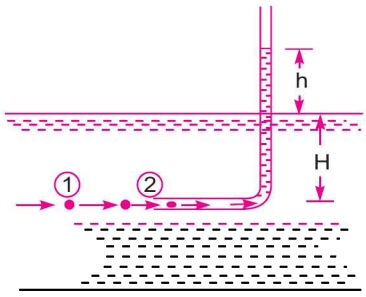


Fig. 6.13 *Pitot-tube.*

The lower end, which is bent through 90° is directed in the up-stream direction as shown in Fig. 6.13. The liquid rises up in the tube due to the conversion of kinetic energy into pressure energy. The velocity is determined by measuring the rise of liquid in the tube.

Consider two points (1) and (2) at the same level in such a way that point (2) is just as the inlet of the pitot-tube and point (1) is far away from the tube.

Let p_1 = intensity of pressure at point (1)

v_1 = velocity of flow at (1)

p_2 = pressure at point (2)

v_2 = velocity at point (2), which is zero

H = depth of tube in the liquid

h = rise of liquid in the tube above the free surface.

Applying Bernoulli's equation at points (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2$$

But $z_1 = z_2$ as points (1) and (2) are on the same line and $v_2 = 0$.

$$\frac{p_1}{\rho g} = \text{pressure head at (1)} = H$$

$$\frac{p_2}{\rho g} = \text{pressure head at (2)} = (h + H)$$

Substituting these values, we get

$$\therefore \quad H + \frac{v_1^2}{2g} = (h + H) \quad \therefore \quad h = \frac{v_1^2}{2g} \quad \text{or} \quad v_1 = \sqrt{2gh}$$

This is theoretical velocity. Actual velocity is given by

$$(v_1)_{act} = C_v \sqrt{2gh}$$

where C_v = Co-efficient of pitot-tube

$\therefore$ Velocity at any point $v = C_v \sqrt{2gh}$ ...(6.14)

Velocity of flow in a pipe by pitot-tube. For finding the velocity at any point in a pipe by pitot-tube, the following arrangements are adopted :

1. Pitot-tube along with a vertical piezometer tube as shown in Fig. 6.14.
2. Pitot-tube connected with piezometer tube as shown in Fig. 6.15.
3. Pitot-tube and vertical piezometer tube connected with a differential U-tube manometer as shown in Fig. 6.16.

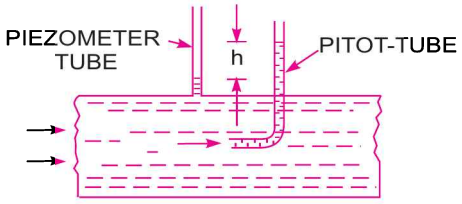


Fig. 6.14

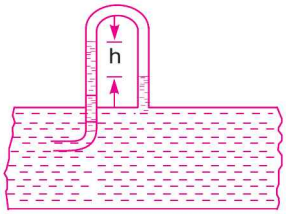


Fig. 6.15

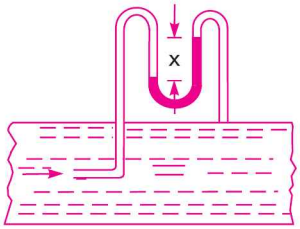


Fig. 6.16

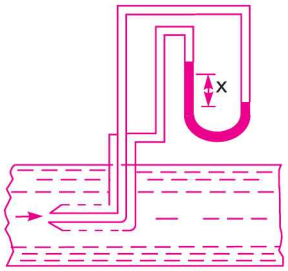


Fig. 6.17

4. Pitot-static tube, which consists of two circular concentric tubes one inside the other with some annular space in between as shown in Fig. 6.17. The outlet of these two tubes are connected to the differential manometer where the difference of pressure head 'h' is measured by knowing the difference of the levels of the manometer liquid say x. Then $h = x \left[\frac{S_g}{S_o} - 1\right]$.

Problem 6.24 *A pitot-static tube placed in the centre of a 300 mm pipe line has one orifice pointing upstream and other perpendicular to it. The mean velocity in the pipe is 0.80 of the central velocity. Find the discharge through the pipe if the pressure difference between the two orifices is 60 mm of water. Take the co-efficient of pitot tube as $C_v = 0.98$.*

Solution. Given :

Dia. of pipe, $d = 300$ mm $= 0.30$ m

Diff. of pressure head, $h = 60$ mm of water $= .06$ m of water

$C_v = 0.98$

Mean velocity, $\overline{V} = 0.80 \times$ Central velocity

Central velocity is given by equation (6.14)

$$= C_v \sqrt{2gh} = 0.98 \times \sqrt{2 \times 9.81 \times .06} = 1.063 \text{ m/s}$$

$\therefore$ $\overline{V} = 0.80 \times 1.063 = 0.8504$ m/s

Discharge, Q = Area of pipe $\times \overline{V}$

$$= \frac{\pi}{4} d^2 \times \overline{V} = \frac{\pi}{4} (.30)^2 \times 0.8504 = \textbf{0.06 m}^3\textbf{/s. Ans.}$$

Problem 6.25 *Find the velocity of the flow of an oil through a pipe, when the difference of mercury level in a differential U-tube manometer connected to the two tappings of the pitot-tube is 100 mm. Take co-efficient of pitot-tube 0.98 and sp. gr. of oil = 0.8.*

Solution. Given :

Diff. of mercury level, $x = 100$ mm $= 0.1$ m

Sp. gr. of oil, $S_o = 0.8$

Sp. gr. of mercury, $S_g = 13.6$

$C_v = 0.98$

Diff. of pressure head, $h = x \left[\frac{S_g}{S_o} - 1 \right] = .1 \left[\frac{13.6}{0.8} - 1 \right] = 1.6$ m of oil

$\therefore$ Velocity of flow $= C_v \sqrt{2gh} = 0.98 \sqrt{2 \times 9.81 \times 1.6} = \textbf{5.49 m/s. Ans.}$

Problem 6.26 *A pitot-static tube is used to measure the velocity of water in a pipe. The stagnation pressure head is 6 m and static pressure head is 5 m. Calculate the velocity of flow assuming the co-efficient of tube equal to 0.98.*

Solution. Given :

Stagnation pressure head, $h_s = 6$ m

Static pressure head, $h_t = 5$ m

$\therefore$ $h = 6 - 5 = 1$ m

Velocity of flow, $V = C_v \sqrt{2gh} = 0.98 \sqrt{2 \times 9.81 \times 1} = \textbf{4.34 m/s. Ans.}$

Problem 6.27 *A sub-marine moves horizontally in sea and has its axis 15 m below the surface of water. A pitot-tube properly placed just in front of the sub-marine and along its axis is connected to the two limbs of a U-tube containing mercury. The difference of mercury level is found to be 170 mm. Find the speed of the sub-marine knowing that the sp. gr. of mercury is 13.6 and that of sea-water is 1.026 with respect of fresh water.*

Solution. Given :

Diff. of mercury level, $x = 170$ mm $= 0.17$ m

Sp. gr. of mercury, $S_g = 13.6$

Sp. gr. of sea-water, $S_o = 1.026$

$\therefore$ $h = x \left[\frac{S_g}{S_o} - 1 \right] = 0.17 \left[\frac{13.6}{1.026} - 1 \right] = 2.0834$ m

$\therefore$ $V = \sqrt{2gh} = \sqrt{2 \times 9.81 \times 2.0834} = 6.393$ m/s

$$= \frac{6.393 \times 60 \times 60}{1000} \text{ km/hr} = \textbf{23.01 km/hr. Ans.}$$

Problem 6.28 *A pitot-tube is inserted in a pipe of 300 mm diameter. The static pressure in pipe is 100 mm of mercury (vacuum). The stagnation pressure at the centre of the pipe, recorded by the*

pitot-tube is 0.981 N/cm^2. Calculate the rate of flow of water through pipe, if the mean velocity of flow is 0.85 times the central velocity. Take C_v = 0.98.

Solution. Given :

Dia. of pipe, $d = 300 \text{ mm} = 0.30 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} (.3)^2 = 0.07068 \text{ m}^2$

Static pressure head = 100 mm of mercury (vacuum)

$= -\frac{100}{1000} \times 13.6 = -1.36 \text{ m of water}$

Stagnation pressure $= .981 \text{ N/cm}^2 = .981 \times 10^4 \text{ N/m}^2$

$\therefore$ Stagnation pressure head $= \frac{.981 \times 10^4}{\rho g} = \frac{.981 \times 10^4}{1000 \times 9.81} = 1 \text{ m}$

$\therefore$ h = Stagnation pressure head – Static pressure head

$= 1.0 - (-1.36) = 1.0 + 1.36 = 2.36 \text{ m of water}$

$\therefore$ Velocity at centre $= C_v \sqrt{2gh}$

$= 0.98 \times \sqrt{2 \times 9.81 \times 2.36} = 6.668 \text{ m/s}$

Mean velocity, $\overline{V} = 0.85 \times 6.668 = 5.6678 \text{ m/s}$

$\therefore$ Rate of flow of water $= \overline{V} \times$ area of pipe

$= 5.6678 \times 0.07068 \text{ m}^3/\text{s} =$ **0.4006 m^3/s. Ans.**

▶ 6.8 THE MOMENTUM EQUATION

It is based on the law of conservation of momentum or on the momentum principle, which states that the net force acting on a fluid mass is equal to the change in momentum of flow per unit time in that direction. The force acting on a fluid mass 'm' is given by the Newton's second law of motion,

$$F = m \times a$$

where a is the acceleration acting in the same direction as force F.

But $$a = \frac{dv}{dt}$$

$\therefore$ $$F = m \frac{dv}{dt}$$

$$= \frac{d(mv)}{dt} \quad \{m \text{ is constant and can be taken inside the differential}\}$$

$\therefore$ $$F = \frac{d(mv)}{dt} \qquad ...(6.15)$$

Equation (6.15) is known as the momentum principle.

Equation (6.15) can be written as $F.dt = d(mv)$ ...(6.16)

which is known as the *impulse-momentum equation* and states that the impulse of a force F acting on a fluid of mass m in a short interval of time dt is equal to the change of momentum $d(mv)$ in the direction of force.

Force exerted by a flowing fluid on a pipe bend

The impulse-momentum equation (6.16) is used to determine the resultant force exerted by a flowing fluid on a pipe bend.

Consider two sections (1) and (2), as shown in Fig. 6.18.

Let v_1 = velocity of flow at section (1),

p_1 = pressure intensity at section (1),

A_1 = area of cross-section of pipe at section (1) and

v_2, p_2, A_2 = corresponding values of velocity, pressure and area at section (2).

Let F_x and F_y be the components of the forces exerted by the flowing fluid on the bend in x-and y-directions respectively. Then the force exerted by the bend on the fluid in the directions of x and y will be equal to F_x and F_y but in the opposite directions. Hence component of the force exerted by bend on the fluid in the x-direction $= -F_x$ and in the direction of $y = -F_y$. The other external forces acting on the fluid are p_1A_1 and p_2A_2 on the sections (1) and (2) respectively. Then momentum equation in x-direction is given by

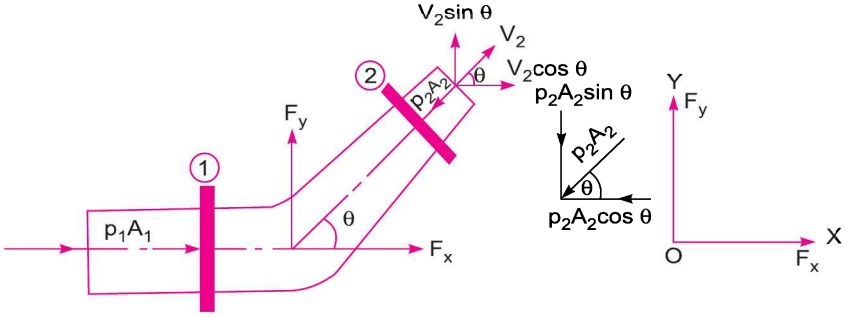


Fig. 6.18 *Forces on bend.*

Net force acting on fluid in the direction of x = Rate of change of momentum in x-direction

$\therefore$ $p_1A_1 - p_2A_2 \cos\theta - F_x$ = (Mass per sec) (change of velocity)

$= \rho Q$ (Final velocity in the direction of x

– Initial velocity in the direction of x)

$= \rho Q\,(V_2 \cos\theta - V_1)$ ...(6.17)

$\therefore$ $F_x = \rho Q\,(V_1 - V_2 \cos\theta) + p_1A_1 - p_2A_2 \cos\theta$ ...(6.18)

Similarly the momentum equation in y-direction gives

$0 - p_2A_2 \sin\theta - F_y = \rho Q\,(V_2 \sin\theta - 0)$ ...(6.19)

$\therefore$ $F_y = \rho Q\,(-V_2 \sin\theta) - p_2A_2 \sin\theta$ ...(6.20)

Now the resultant force (F_R) acting on the bend

$$= \sqrt{F_x^2 + F_y^2} \qquad \text{...(6.21)}$$

And the angle made by the resultant force with horizontal direction is given by

$$\tan\theta = \frac{F_y}{F_x} \qquad \text{...(6.22)}$$

Problem 6.29 *A 45° reducing bend is connected in a pipe line, the diameters at the inlet and outlet of the bend being 600 mm and 300 mm respectively. Find the force exerted by water on the bend if the intensity of pressure at inlet to bend is 8.829 N/cm^2 and rate of flow of water is 600 litres/s.*

Solution. Given :

Angle of bend, $\theta = 45°$

Dia. at inlet, $D_1 = 600 \text{ mm} = 0.6 \text{ m}$

$\therefore$ Area, $A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} (.6)^2$

$= 0.2827 \text{ m}^2$

Dia. at outlet, $D_2 = 300 \text{ mm} = 0.30 \text{ m}$

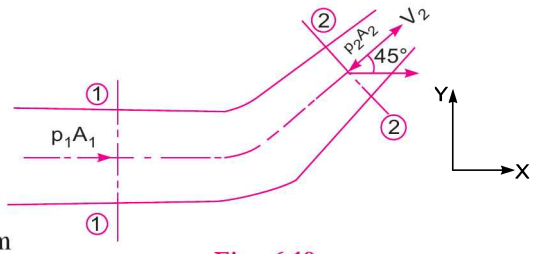


Fig. 6.19

$\therefore$ Area, $A_2 = \frac{\pi}{4} (.3)^2 = 0.07068 \text{ m}^2$

Pressure at inlet, $p_1 = 8.829 \text{ N/cm}^2 = 8.829 \times 10^4 \text{ N/m}^2$

$Q = 600 \text{ lit/s} = 0.6 \text{ m}^3/\text{s}$

$$V_1 = \frac{Q}{A_1} = \frac{0.6}{.2827} = 2.122 \text{ m/s}$$

$$V_2 = \frac{Q}{A_2} = \frac{0.6}{.07068} = 8.488 \text{ m/s.}$$

Applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2$$

But $z_1 = z_2$

$\therefore$ $$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} \quad \text{or} \quad \frac{8.829 \times 10^4}{1000 \times 9.81} + \frac{2.122^2}{2 \times 9.81} = \frac{p_2}{\rho g} + \frac{8.488^2}{2 \times 9.81}$$

$$9 + .2295 = p_2/\rho g + 3.672$$

$\therefore$ $$\frac{p_2}{\rho g} = 9.2295 - 3.672 = 5.5575 \text{ m of water}$$

$\therefore$ $p_2 = 5.5575 \times 1000 \times 9.81 \text{ N/m}^2 = 5.45 \times 10^4 \text{ N/m}^2$

Forces on the bend in x- and y-directions are given by equations (6.18) and (6.20) as

$$F_x = \rho Q [V_1 - V_2 \cos \theta] + p_1 A_1 - p_2 A_2 \cos \theta$$

$$= 1000 \times 0.6 [2.122 - 8.488 \cos 45°] + 8.829 \times 10^4 \times .2827 - 5.45 \times 10^4 \times .07068 \times \cos 45°$$

$$= - 2327.9 + 24959.6 - 2720.3 = 24959.6 - 5048.2$$

$$= 19911.4 \text{ N}$$

and $$F_y = \rho Q [- V_2 \sin \theta] - p_2 A_2 \sin \theta$$

$$= 1000 \times 0.6 [- 8.488 \sin 45°] - 5.45 \times 10^4 \times .07068 \times \sin 45°$$

$$= - 3601.1 - 2721.1 = - 6322.2 \text{ N}$$

–ve sign means F_y is acting in the downward direction

$\therefore$ Resultant force, $F_R = \sqrt{F_x^2 + F_y^2}$

$$= \sqrt{(19911.4)^2 + (-6322.2)^2}$$

$$= \mathbf{20890.9\ N.\ Ans.}$$

The angle made by resultant force with x-axis is given by equation (6.22) or

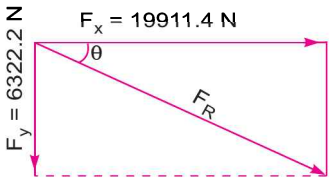


Fig. 6.20

$$\tan\theta = \frac{F_y}{F_x} = \frac{6322.2}{19911.4} = 0.3175$$

$$\therefore \qquad \theta = \tan^{-1} .3175 = \mathbf{17°\ 36'.\ Ans.}$$

Problem 6.30 *250 litres/s of water is flowing in a pipe having a diameter of 300 mm. If the pipe is bent by 135° (that is change from initial to final direction is 135°), find the magnitude and direction of the resultant force on the bend. The pressure of water flowing is 39.24 N/cm^2.*

Solution. Given :

Pressure, $p_1 = p_2 = 39.24\ \text{N/cm}^2 = 39.24 \times 10^4\ \text{N/m}^2$

Discharge, $Q = 250\ \text{litres/s} = 0.25\ \text{m}^3/\text{s}$

Dia. of bend at inlet and outlet, $D_1 = D_2 = 300\ \text{mm} = 0.3\ \text{m}$

$\therefore$ Area, $A_1 = A_2 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} \times .3^2 = 0.07068\ \text{m}^2$

Velocity of water at sections (1) and (2), $V = V_1 = V_2 = \frac{Q}{\text{Area}} = \frac{0.25}{.07068} = 3.537\ \text{m/s}.$

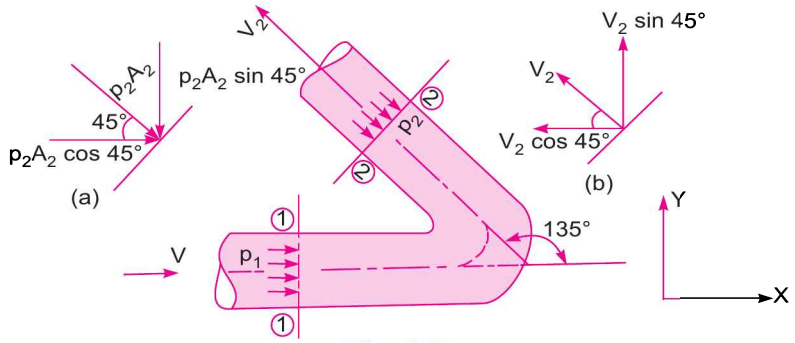


Fig. 6.21

Force along x-axis

$$= F_x = \rho Q[V_{1x} - V_{2x}] + p_{1x}A_1 + p_{2x}A_2$$

where, V_{1x} = initial velocity in the direction of $x = 3.537$ m/s

V_{2x} = final velocity in the direction of $x = -V_2 \cos 45° = -3.537 \times .7071$

p_{1x} = pressure at section (1) in x-direction

$= 39.24\ \text{N/cm}^2 = 39.24 \times 10^4\ \text{N/m}^2$

p_{2x} = pressure at section (2) in x-direction

$= p_2 \cos 45° = 39.24 \times 10^4 \times .7071$

$\therefore$ $F_x = 1000 \times .25[3.537 - (-3.537 \times .7071)] + 39.24 \times 10^4 \times .07068 + 39.24 \times 10^4 \times .07068 \times .7071$

$= 1000 \times .25[3.537 + 3.537 \times .7071] + 39.24 \times 10^4 \times .07068\ [1 + .7071]$

$= 1509.4 + 47346 = 48855.4$ N

Force along y-axis

$= F_y = \rho Q[V_{1y} - V_{2y}] + (p_1A_1)_y + (p_2A_2)_y$

where V_{1y} = initial velocity in y-direction = 0

V_{2y} = final velocity in y-direction = $- V_2 \sin 45° = 3.537 \times .7071$

$(p_1A_1)_y$ = pressure force in y-direction = 0

$(p_2A_2)_y$ = pressure force at (2) in y-direction

$= - p_2A_2 \sin 45° = - 39.24 \times 10^4 \times .07068 \times .7071$

$\therefore \quad F_y = 1000 \times .25[0 - 3.537 \times .7071] + 0 + (- 39.24 \times 10^4 \times .07068 \times .7071)$

$= - 625.2 - 19611.1 = - 20236.3$ N

–ve sign means F_y is acting in the downward direction

$\therefore$ Resultant force, $F_R = \sqrt{F_x^2 + F_y^2}$

$= \sqrt{48855.4^2 + 20236.3^2}$

$=$ **52880.6 N. Ans.**

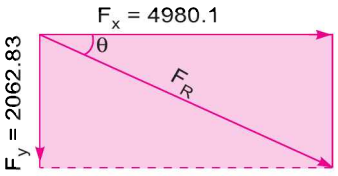


Fig. 6.22

The direction of the resultant force F_R, with the x-axis is given as

$$\tan \theta = \frac{F_y}{F_x} = \frac{20236.3}{48855.4} = 0.4142$$

$\therefore \quad \theta =$ **22° 30′. Ans.**

Problem 6.31 *A 300 mm diameter pipe carries water under a head of 20 metres with a velocity of 3.5 m/s. If the axis of the pipe turns through 45°, find the magnitude and direction of the resultant force at the bend.*

Solution. Given :

Dia. of bend, $D = D_1 = D_2 = 300$ mm $= 0.30$ m

$\therefore$ Area, $A = A_1 = A_2 = \frac{\pi}{4} D^2 = \frac{\pi}{4} \times .3^2 = 0.07068$ m^2

Velocity, $V = V_1 = V_2 = 3.5$ m/s

$\theta = 45°$

Discharge, $Q = A \times V = 0.07068 \times 3.5 = 0.2475$ m^3/s

Pressure head $= 20$ m of water or $\frac{p}{\rho g} = 20$ m of water

$\therefore \quad p = 20 \times \rho g = 20 \times 1000 \times 9.81$ N/m^2 $= 196200$ N/m^2

$\therefore$ Pressure intensity, $p = p_1 = p_2 = 196200$ N/m^2

Now $V_{1x} = 3.5$ m/s, $V_{2x} = V_2 \cos 45° = 3.5 \times .7071$

$V_{1y} = 0$, $V_{2y} = V_2 \sin 45° = 3.5 \times .7071$

$(p_1A_1)_x = p_1A_1 = 196200 \times .07068$, $(p_1A_1)_y = 0$

$(p_2A_2)_x = - p_2A_2 \cos 45°$, $(p_2A_2)_y = - p_2A_2 \sin 45°$

Force along x-axis, $F_x = \rho Q[V_{1x} - V_{2x}] + (p_1A_1)_x + (p_2A_2)_x$

$= 1000 \times .2475[3.5 - 3.5 \times .7071] + 196200 \times .07068 - p_2A_2 \times \cos 45°$

$= 253.68 + 196200 \times .07068 - 196200 \times .07068 \times 0.7071$

$= 253.68 + 13871.34 - 9808.04 = 4316.98$ N

Force along y-axis, $F_y = \rho Q\,[V_1 y - V_2 y] + (p_1 A_1)_y + (p_2 A_2)_y$

$= 1000 \times .2475[0 - 3.5 \times .7071] + 0 + [-\,p_2 A_2 \sin 45°]$

$= - 612.44 - 196200 \times .07068 \times .7071$

$= - 612.44 - 9808 = - 10420.44$ N

$\therefore$ Resultant force $F_R = \sqrt{F_x^2 + F_y^2} = \sqrt{(4316.98)^2 + (10420.44)^2}$ = **11279 N. Ans.**

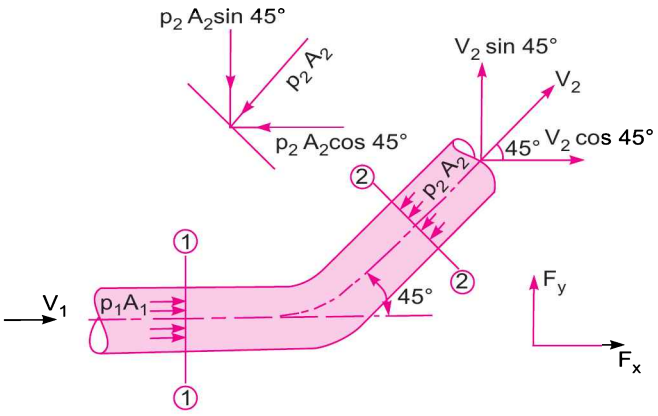


Fig. 6.23

The angle made by F_R with x-axis

$$\tan\theta = \frac{F_y}{F_x} = \frac{10420.44}{4316.98} = 2.411$$

$\therefore$ $\theta = \tan^{-1} 2.411 =$ **67° 28′. Ans.**

Problem 6.32 *In a 45° bend a rectangular air duct of 1 m^2 cross-sectional area is gradually reduced to 0.5 m^2 area. Find the magnitude and direction of the force required to hold the duct in position if the velocity of flow at the 1 m^2 section is 10 m/s, and pressure is 2.943 N/cm^2. Take density of air as 1.16 kg/m^3.*

Solution. Given :

Area at section (1), $A_1 = 1$ m^2

Area at section (2), $A_2 = 0.5$ m^2

Velocity at section (1), $V_1 = 10$ m/s

Pressure at section (1), $p_1 = 2.943$ N/cm^2 $= 2.943 \times 10^4$ N/m^2 $= 29430$ N/m^2

Density of air, $\rho = 1.16$ kg/m^3

Applying continuity equation at sections (1) and (2)

$$A_1 V_1 = A_2 V_2$$

$\therefore$ $$V_2 = \frac{A_1 V_1}{A_2} = \frac{1}{0.5} \times 10 = 20 \text{ m/s}$$

Discharge $Q = A_1V_1 = 1 \times 10 = 10\ m^3/s$

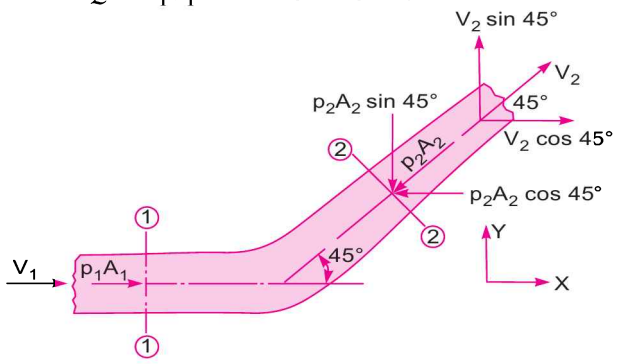


Fig. 6.24

Applying Bernoulli's equation at sections (1) and (2)

$$\therefore \quad \frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} \qquad \{\because \quad Z_1 = Z_2\}$$

or $$\frac{2.943 \times 10^4}{1.16 \times 9.81} + \frac{10^2}{2 \times 9.81} = \frac{p_2}{\rho g} + \frac{20^2}{2 \times 9.81}$$

$$\therefore \quad \frac{p_2}{\rho g} = \frac{2.943 \times 10^4}{1.16 \times 9.81} + \frac{10^2}{2 \times 9.81} - \frac{20^2}{2 \times 9.81}$$

$$= 2586.2 + 5.0968 - 20.387 = 2570.90 \text{ m}$$

$$\therefore \quad p_2 = 2570.90 \times 1.16 \times 9.81 = 29255.8 \text{ N}$$

Force along x-axis, $F_x = \rho Q\ [V_{1x} - V_{2x}] + (p_1A_1)_x + (p_2A_2)_x$

where $A_{1x} = 10$ m/s, $V_{2x} = V_2 \cos 45° = 20 \times .7071$,

$(p_1A_1)_x = p_1A_1 = 29430 \times 1 = 29430$ N

and $(p_2A_2)_x = - p_2A_2 \cos 45° = - 29255.8 \times 0.5 \times .7071$

$$\therefore \quad F_x = 1.16 \times 10[10 - 20 \times .7071] + 29430 \times 1 - 29255.8 \times .5 \times .7071$$

$$= - 48.04 + 29430 - 10343.37 = 0 - 19038.59 \text{ N}$$

Similarly *force along y-axis,* $F_y = \rho Q[V_{1y} - V_{2y}] + (p_1A_1)_y + (p_2A_2)_y$

where $V_{1y} = 0$, $V_{2y} = V_2 \sin 45° = 20 \times .7071 = 14.142$

$(p_1A_1)_y = 0$ and $(p_2A_2)_y = - p_2A_2 \sin 45° = - 29255.8 \times .5 \times .7071 = - 10343.37$

$$F_y = 1.16 \times 10\ [0 - 14.142] + 0 - 10343.37$$

$$= - 164.05 - 10343.37 = - 10507.42 \text{ N}$$

$$\therefore \quad \text{Resultant force,} \quad F_R = \sqrt{F_x^2 + F_y^2} = \sqrt{(19038.6)^2 + (10507.42)^2} = \mathbf{21746.6\ N.\ Ans.}$$

The direction of F_R with x-axis is given as

$$\tan \theta = \frac{F_y}{F_x} = \frac{10507.42}{19038.6} = 0.5519$$

$$\therefore \quad \theta = \tan^{-1} .5519 = \mathbf{28°\ 53'.\ Ans.}$$

F_R is the force exerted on bend. Hence the force required to hold the duct in position is equal to 21746.6 N but it is acting in the opposite direction of F_R. **Ans.**

Problem 6.33 *A pipe of 300 mm diameter conveying 0.30 m^3/s of water has a right angled bend in a horizontal plane. Find the resultant force exerted on the bend if the pressure at inlet and outlet of the bend are 24.525 N/cm^2 and 23.544 N/cm^2.*

Solution. Given :

Dia. of bend, $D = 300 \text{ mm} = 0.3 \text{ m}$

$\therefore$ Area, $A = A_1 = A_2 = \frac{\pi}{4}(.3)^2 = 0.07068 \text{ m}^2$

$\therefore$ Discharge, $Q = 0.30 \text{ m}^3/\text{s}$

$\therefore$ Velocity, $V = V_1 = V_2 = \frac{Q}{A} = \frac{0.30}{.07068} = 4.244 \text{ m/s}$

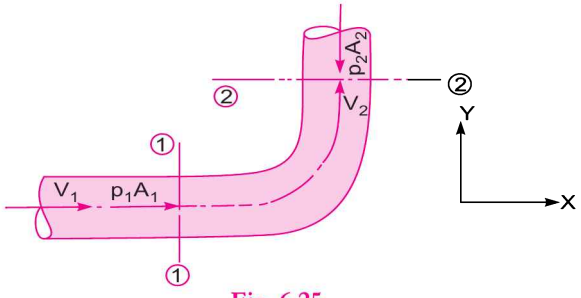


Fig. 6.25

Angle of bend, $\theta = 90°$

$p_1 = 24.525 \text{ N/cm}^2 = 24.525 \times 10^4 \text{ N/m}^2 = 245250 \text{ N/m}^2$

$p_2 = 23.544 \text{ N/cm}^2 = 23.544 \times 10^4 \text{ N/m}^2 = 235440 \text{ N/m}^2$

Force on bend along x-axis $F_x = \rho Q\,[V_{1x} - V_{2x}] + (p_1A_1)_x + (p_2A_2)_x$

where $\rho = 1000$, $V_{1x} = V_1 = 4.244$ m/s, $V_{2x} = 0$

$(p_1A_1)_x = p_1A_1 = 245250 \times .07068$

$(p_2A_2)_x = 0$

$\therefore$ $F_x = 1000 \times 0.30\,[4.244 - 0] + 245250 \times .07068 + 0$

$= 1273.2 + 17334.3 = 18607.5 \text{ N}$

Force on bend along y-axis, $F_y = \rho Q\,[V_{1y} - V_{2y}] + (p_1A_1)_y + (p_2A_2)_y$

where $V_{1y} = 0$, $V_{2y} = V_2 = 4.244$ m/s

$(p_1A_1)_y = 0$, $(p_2A_2)_y = -p_2A_2 = -235440 \times .07068 = -16640.9$

$\therefore$ $F_y = 1000 \times 0.30[0 - 4.244] + 0 - 16640.9$

$= -1273.2 - 16640.9 = -17914.1 \text{ N}$

$\therefore$ Resultant force, $F_R = \sqrt{F_x^2 + F_y^2} = \sqrt{(18607.5)^2 + (17914.1)^2} = -25829.3 \text{ N}$

and $\tan\theta = \frac{F_y}{F_x} = \frac{17914.1}{18607.5} = 0.9627$

$\therefore$ $\theta = \mathbf{43° 54'}$. **Ans.**

Problem 6.34 *A nozzle of diameter 20 mm is fitted to a pipe of diameter 40 mm. Find the force exerted by the nozzle on the water which is flowing through the pipe at the rate of 1.2 m^3/minute.*

Solution. Given :

Dia. of pipe, $D_1 = 40 \text{ mm} = 40 \times 10^{-3} \text{ m} = .04 \text{ m}$

$\therefore$ Area, $A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} (.04)^2 = 0.001256 \text{ m}^2$

Dia. of nozzle, $D_2 = 20 \text{ mm} = 0.02 \text{ m}$

$\therefore$ Area, $A_2 = \frac{\pi}{4} (.02)^2 = .000314 \text{ m}^2$

Discharge, $Q = 1.2 \text{ m}^3/\text{minute} = \frac{1.2}{60} \text{ m}^3/\text{s} = 0.02 \text{ m}^3/\text{s}$

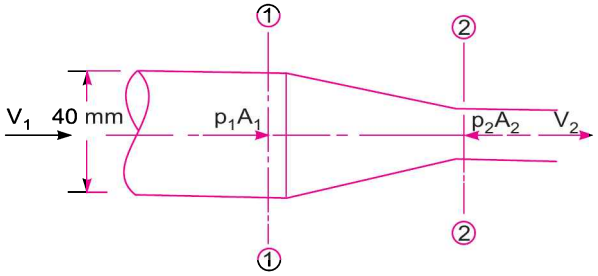


Fig. 6.26

Applying continuity equation at sections (1) and (2),

$$A_1V_1 = A_2V_2 = Q$$

$\therefore$ $$V_1 = \frac{Q}{A_1} = \frac{0.2}{.001256} = 15.92 \text{ m/s}$$

and $$V_2 = \frac{Q}{A_2} = \frac{0.2}{.000314} = 63.69 \text{ m/s}$$

Applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2$$

Now $z_1 = z_2$, $\frac{p_2}{\rho g}$ = atmospheric pressure = 0

$\therefore$ $$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{V_2^2}{2g}$$

$\therefore$ $$\frac{p_1}{\rho g} = \frac{V_2^2}{2g} - \frac{V_1^2}{2g} = \frac{(63.69^2)}{2 \times 9.81} - \frac{(15.92^2)}{2 \times 9.81} = 206.749 - 12.917$$

$$= 193.83 \text{ m of water}$$

$\therefore$ $$p_1 = 193.83 \times 1000 \times 9.81 \frac{\text{N}}{\text{m}^2} = 1901472 \frac{\text{N}}{\text{m}^2}$$

Let the force exerted by the nozzle on water = F_x

Net force in the direction of x = rate of change of momentum in the direction of x

$\therefore \quad p_1A_1 - p_2A_2 + F_x = \rho Q(V_2 - V_1)$

where p_2 = atmospheric pressure = 0 and $\rho = 1000$

$\therefore$ 1901472 × .001256 – 0 + F_x = 1000 × 0.02(63.69 – 15.92) or 2388.24 + F_x = 916.15

$\therefore \quad F_x = -2388.24 + 916.15 = \mathbf{-1472.09.}$ **Ans.**

–ve sign indicates that the force exerted by the nozzle on water is acting from right to left.

Problem 6.35 *The diameter of a pipe gradually reduces from 1 m to 0.7 m as shown in Fig. 6.27. The pressure intensity at the centre-line of 1 m section 7.848 kN/m² and rate of flow of water through the pipe is 600 litres/s. Find the intensity of pressure at the centre-line of 0.7 m section. Also determine the force exerted by flowing water on transition of the pipe.*

Solution. Given :

Dia. of pipe at section 1, $D_1 = 1$ m

$\therefore$ Area, $A_1 = \frac{\pi}{4}(1)^2 = 0.7854 \text{ m}^2$

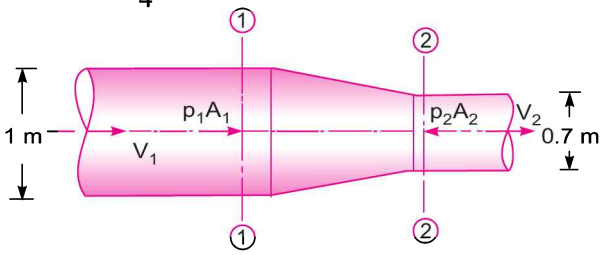


Fig. 6.27

Dia. of pipe at section 2, $D_2 = 0.7$ m

$\therefore$ Area, $A_2 = \frac{\pi}{4}(0.7)^2 = 0.3848 \text{ m}^2$

Pressure at section 1, $p_1 = 7.848 \text{ kN/m}^2 = 7848 \text{ N/m}^2$

Discharge, $Q = 600 \text{ litres/s} = \frac{600}{1000} = 0.6 \text{ m}^3/\text{s}$

Applying continuity equation,

$$A_1V_1 = A_2V_2 = Q$$

$\therefore \quad V_1 = \frac{Q}{A_2} = \frac{0.6}{0.7854} = 0.764 \text{ m/s}$

$$V_2 = \frac{Q}{A_2} = \frac{0.6}{.3854} = 1.55 \text{ m/s}$$

Applying Bernoulli's equation at sections (1) and (2),

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} \qquad \{\because \text{ pipe is horizontal, } \therefore \ z_1 = z_2\}$$

or $$\frac{7848}{1000 \times 9.81} + \frac{(.764)^2}{2 \times 9.81} = \frac{p_2}{\rho g} + \frac{(1.55)^2}{2 \times 9.81}$$

$$\therefore \quad \frac{p_2}{\rho g} = 0.8 + \frac{(.764)^2}{2 \times 9.81} - \frac{(1.55)^2}{2 \times 9.81}$$

$$= 0.8 + 0.0297 - 0.122 = 0.7077 \text{ m of water}$$

$$\therefore \quad p_2 = 0.7077 \times 9.81 \times 1000$$

$$= \mathbf{6942.54\ N/m^2 \text{ or } 6.942\ kN/m^2.\ Ans.}$$

Let F_x = the force exerted by pipe transition on the flowing water in the direction of flow

Then net force in the direction of flow = rate of change of momentum in the direction of flow

or $$p_1A_1 - p_2A_2 + F_x = \rho(V_2 - V_1)$$

$$\therefore \quad 7848 \times .7854 - 6942.54 \times .3848 + F_x = 1000 \times 0.6[1.55 - .764]$$

or $$6163.8 - 2671.5 + F_x = 471.56$$

$$\therefore \quad F_x = 471.56 - 6163.8 + 2671.5 = -3020.74 \text{ N}$$

$\therefore$ The force exerted by water on pipe transition

$$= -F_x = -(-3020.74) = \mathbf{3020.74\ N.\ Ans.}$$

▶ 6.9 MOMENT OF MOMENTUM EQUATION

Moment of momentum equation is derived from moment of momentum principle which states that the resulting torque acting on a rotating fluid is equal to the rate of change of moment of momentum.

Let V_1 = velocity of fluid at section 1,

r_1 = radius of curvature at section 1,

Q = rate of flow of fluid,

ρ = density of fluid,

and V_2 and r_2 = velocity and radius of curvature at section 2

Momentum of fluid at section 1 = mass × velocity = $\rho Q \times V_1/s$

$\therefore$ Moment of momentum per second at section 1,

$$= \rho Q \times V_1 \times r_1$$

Similarly moment of momentum per second of fluid at section 2

$$= \rho Q \times V_2 \times r_2$$

$\therefore$ Rate of change of moment of momentum

$$= \rho Q V_2 r_2 - \rho Q V_1 r_1 = \rho Q[V_2 r_2 - V_1 r_1]$$

According to moment of momentum principle

Resultant torque = rate of change of moment of momentum

or $$T = \rho Q[V_2 r_2 - V_1 r_1] \qquad \ldots(6.23)$$

Equation (6.23) is known as moment of momentum equation. This equation is applied :

1. For analysis of flow problems in turbines and centrifugal pumps.
2. For finding torque exerted by water on sprinkler.

Problem 6.36 *A lawn sprinkler with two nozzles of diameter 4 mm each is connected across a tap of water as shown in Fig. 6.28. The nozzles are at a distance of 30 cm and 20 cm from the centre of the tap. The rate of flow of water through tap is 120 cm^3/s. The nozzles discharge water in the downward direction. Determine the angular speed at which the sprinkler will rotate free.*

Solution. Given :

Dia. of nozzles A and B,

$$D = D_A = D_B = 4 \text{ mm} = .004 \text{ m}$$

$\therefore$ Area, $$A = \frac{\pi}{4} (.004)^2 = .00001256 \text{ m}^2$$

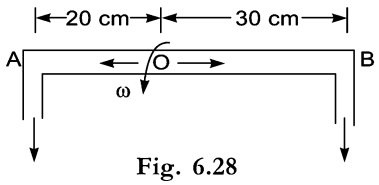


Fig. 6.28

Discharge $$Q = 120 \text{ cm}^3/\text{s}$$

Assuming the discharge to be equally divided between the two nozzles, we have

$$Q_A = Q_B = \frac{Q}{2} = \frac{120}{2} = 60 \text{ cm}^3/\text{s} = 60 \times 10^{-6} \text{ m}^3/\text{s}$$

$\therefore$ Velocity of water at the outlet of each nozzle,

$$V_A = V_B = \frac{Q_A}{A} = \frac{60 \times 10^{-6}}{.00001256} = 4.777 \text{ m/s.}$$

The jet of water coming out from nozzles A and B is having velocity 4.777 m/s. These jets of water will exert force in the opposite direction, *i.e.*, force exerted by the jets will be in the upward direction. The torque exerted will also be in the opposite direction. Hence torque at B will be in the anti-clockwise direction and at A in the clockwise direction. But torque at B is more than the torque at A and hence sprinkle, if free, will rotate in the anti-clockwise direction as shown in Fig. 6.28.

Let ω = angular velocity of the sprinkler.

Then absolute velocity of water at A,

$$V_1 = V_A + \omega \times r_A$$

where r_A = distance of nozzle A from the centre of tap

$= 20 \text{ cm} = 0.2 \text{ m}$ $\{\omega \times r_A$ = tangential velocity due to rotation$\}$

$$V_1 = (4.777 + \omega \times 0.2) \text{ m/s}$$

Here $\omega \times r_A$ is added to V_A as V_A and tangential velocity due to rotation ($\omega \times r_A$) are in the same direction as shown in Fig. 6.28.

Similarly, absolute velocity of water at B,

$$V_2 = V_B - \text{tangential velocity due to rotation}$$
$$= 4.777 - \omega \times r_B \qquad \{\text{where } r_B = 30 \text{ cm} = 0.3 \text{ m}\}$$
$$= (4.777 - \omega \times 0.3)$$

Now applying equation (6.23), we get

$$T = \rho Q[V_2 r_2 - V_1 r_1] \qquad \text{Here } r_2 = r_B,\ r_1 = r_A,\ Q = Q_A = Q_B$$
$$= \rho Q_A[V_2 r_B - V_1 r_A]$$
$$= 1000 \times 60 \times 10^{-6} [(4.777 \times 0.3\ \omega) \times .3 - (4.777 + 0.2\ \omega) \times .2]$$

The moment of momentum of the fluid entering sprinkler is given zero and also there is no external torque applied on the sprinkler. Hence resultant external torque is zero, *i.e.*, $T = 0$

$\therefore$ $$1000 \times 60 \times 10^{-6} [(4.777 - 0.3\ \omega) \times .3 - (4.777 + 0.2\ \omega) \times .2] = 0$$

or $$(4.777 - 0.3\ \omega) \times 0.3 - (4.777 + 0.2\ \omega) \times .2 = 0$$

or $$4.777 \times .3 - .09\ \omega - 4.777 \times .2 - .04\ \omega = 0$$

or $$0.1 \times 4.777 = (.09 + .04)\omega = .13\ \omega$$

$\therefore$ $$\omega = \frac{.4777}{0.13} = \mathbf{3.6746 \text{ rad/s. Ans.}}$$

Problem 6.37 *A lawn sprinkler shown in Fig. 6.29 has 0.8 cm diameter nozzle at the end of a rotating arm and discharges water at the rate of 10 m/s velocity. Determine the torque required to hold the rotating arm stationary. Also determine the constant speed of rotation of the arm, if free to rotate.*

Solution. Dia. of each nozzle = 0.8 cm = .008 m

$\therefore$ Area of each nozzle $= \frac{\pi}{4}(.008)^2 = .00005026 \text{ m}^2$

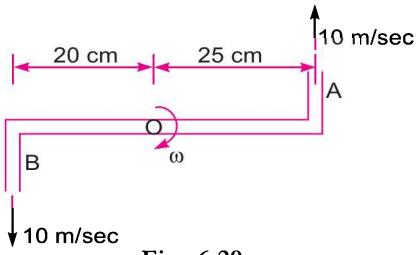


Fig. 6.29

Velocity of flow at each nozzle = 10 m/s.

$\therefore$ Discharge through each nozzle,

Q = Area × Velocity

$= .00005026 \times 10 = .0005026 \text{ m}^3/\text{s}$

Torque exerted by water coming through nozzle A on the sprinkler = moment of momentum of water through A

$= r_A \times \rho \times Q \times V_A = 0.25 \times 1000 \times .0005026 \times 10$ clockwise

Torque exerted by water coming through nozzle B on the sprinkler

$= r_B \times \rho \times Q \times V_B = 0.20 \times 1000 \times .0005026 \times 10$ clockwise

$\therefore$ Total torque exerted by water on sprinkler

$= .25 \times 1000 \times .0005026 \times 10 + .20 \times 1000 \times .0005026 \times 10$

$= 1.2565 + 1.0052 = 2.26$ Nm

$\therefore$ Torque required to hold the rotating arm stationary = Torque exerted by water on sprinkler

$=$ **2.26 Nm. Ans.**

Speed of rotation of arm, if free to rotate

Let ω = speed of rotation of the sprinkler

The absolute velocity of flow of water at the nozzles A and B are

$V_1 = 10.0 - 0.25 \times \omega$ and $V_2 = 10.0 - 0.20 \times \omega$

Torque exerted by water coming out at A, on sprinkler

$= r_A \times \rho \times Q \times V_1 = 0.25 \times 1000 \times .0005026 \times (10 - 0.25\ \omega)$

$= 0.12565\ (10 - 0.25\ \omega)$

Torque exerted by water coming out at B, on sprinkler

$= r_B \times \rho \times Q \times V_2 = 0.20 \times 1000 \times .0005026 \times (10.0 - 0.2\ \omega)$

$= 0.10052\ (10.0 - 0.2\ \omega)$

$\therefore$ Total torque exerted by water = $0.12565\ (10.0 - 0.25\ \omega) + 0.10052\ (10.0 - 0.2\ \omega)$

Since moment of momentum of the flow entering is zero and no external torque is applied on sprinkler, so the resultant torque on the sprinkler must be zero.

$\therefore\ 0.12565\ (10.0 - 0.25\ \omega) + 0.10052(10.0 - 0.2\ \omega) = 0$

$1.2565 - 0.0314\ \omega + 1.0052 - 0.0201\ \omega = 0$

$1.2565 + 1.0052 = \omega\ (0.0314 + 0.0201)$

$2.2617 = 0.0515\ \omega$

$\therefore\ \omega = \frac{2.2617}{0.0515} =$ **43.9 rad/s. Ans.**

and $$N = \frac{60 \times \omega}{2\pi} = \frac{60 \times 43.9}{2\pi} = \textbf{419.2 r.p.m. Ans.}$$

▶ 6.10 FREE LIQUID JETS

Free liquid jet is defined as the jet of water coming out from the nozzle in atmosphere. The path travelled by the free jet is parabolic.

Consider a jet coming from the nozzle as shown in Fig. 6.30. Let the jet at A, makes an angle θ with the horizontal direction. If U is the velocity of jet of water, then the horizontal component and vertical component of this velocity at A are $U \cos \theta$ and $U \sin \theta$.

Consider another point $P(x, y)$ on the centre line of the jet. The co-ordinates of P from A are x and y. Let the velocity of jet at P in the x- and y-directions are u and v. Let a liquid particle takes time 't' to reach from A to P. Then the horizontal and vertical distances travelled by the liquid particle in time 't' are :

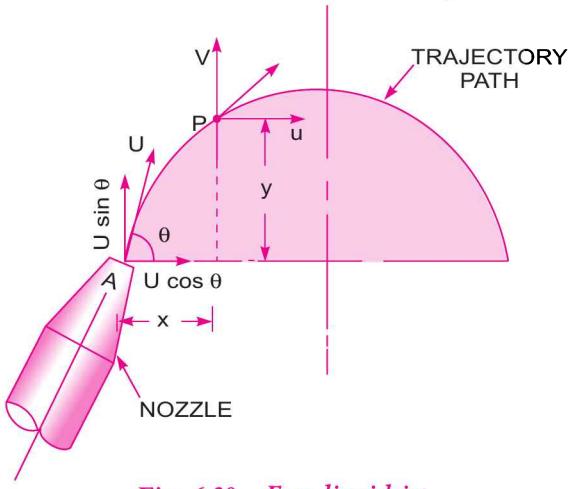


Fig. 6.30 *Free liquid jet.*

$$x = \text{velocity component in } x\text{-direction} \times t$$
$$= U \cos \theta \times t \quad ...(i)$$

and $$y = (\text{vertical component in } y\text{-direction} \times \text{time} - \frac{1}{2} gt^2)$$
$$= U \sin \theta \times t - \frac{1}{2} gt^2 \quad ...(ii)$$

{$\because$ Horizontal component of velocity is constant while the vertical distance is affected by gravity}

From equation (i), the value of t is given as $t = \dfrac{x}{U \cos \theta}$

Substituting this value in equation (ii)

$$y = U \sin \theta \times \frac{x}{U \cos \theta} - \frac{1}{2} \times g \times \left(\frac{x}{U \cos \theta} \right)^2 = x \frac{\sin \theta}{\cos \theta} - \frac{gx^2}{2U^2 \cos^2 \theta}$$

$$= x \tan \theta - \frac{gx^2}{2U^2} \sec^2 \theta \qquad \left\{ \because \frac{1}{\cos^2 \theta} = \sec^2 \theta \right\} \quad ...(6.24)$$

Equation (6.24) gives the variation of y with the square of x. Hence this is the equation of a parabola. Thus the path travelled by the free jet in atmosphere is parabolic.

(*i*) **Maximum height attained by the jet.** Using the relation $V_2^2 - V_1^2 = -2gS$, we get in this case $V_1 = 0$ at the highest point

V_1 = Initial vertical component
$= U \sin\theta$

–ve sign on right hand side is taken as g is acting in the downward direction but particles is moving up.

$$\therefore \quad 0 - (U \sin\theta)^2 = -2g \times S$$

where S is the maximum vertical height attained by the particle.

or $$-U^2 \sin^2\theta = -2gS$$

$$\therefore \quad S = \frac{U^2 \sin^2\theta}{2g} \qquad ...(6.25)$$

(*ii*) **Time of flight.** It is the time taken by the fluid particle in reaching from A to B as shown in Fig. 6.30. Let T is the time of flight.

Using equation (*ii*), we have $y = U \sin\theta \times t - \frac{1}{2} gt^2$

when the particle reaches at B, $y = 0$ and $t = T$

$\therefore$ Above equation becomes as $0 = U \sin\theta \times T - \frac{1}{2} g \times T^2$

or $$0 = U \sin\theta - \frac{1}{2} gT \qquad \{\text{Cancelling } T\}$$

or $$T = \frac{2U \sin\theta}{g} \qquad ...(6.26)$$

(*iii*) **Time to reach highest point.** The time to reach highest point is half the time of flight. Let T^* is the time to reach highest point, then

$$T^* = \frac{T}{2} = \frac{2U \sin\theta}{g \times 2} = \frac{U \sin\theta}{g} \qquad ...(6.27)$$

(*iv*) **Horizontal range of the jet.** The total horizontal distance travelled by the fluid particle is called horizontal range of the jet, *i.e.*, the horizontal distance AB in Fig. 6.30 is called horizontal range of the jet. Let this range is denoted by x^*.

Then x^* = velocity component in x-direction
× time taken by the particle to reach from A to B
$= U \cos\theta \times$ Time of flight

$$= U \cos\theta \times \frac{2U \sin\theta}{g} \qquad \left\{\because T = \frac{2U \sin\theta}{g}\right\}$$

$$= \frac{U^2}{g} 2 \cos\theta \sin\theta = \frac{U^2}{g} \sin 2\theta \qquad ...(6.28)$$

(*v*) **Value of θ for maximum range.** The range x^* will be maximum for a given velocity of projection (U), when $\sin 2\theta$ is maximum

or when $\sin 2\theta = 1$ or $\sin 2\theta = \sin 90° = 1$

$\therefore$ $2\theta = 90°$ or $\theta = 45°$

Then maximum range, $$x^*_{max} = \frac{U^2}{g} \sin^2\theta = \frac{U^2}{g} \qquad \{\because \quad \sin 90° = 1\} \quad ...(6.29)$$

Problem 6.38 *A vertical wall is of 8 m in height. A jet of water is coming out from a nozzle with a velocity of 20 m/s. The nozzle is situated at a distance of 20 m from the vertical wall. Find the angle of projection of the nozzle to the horizontal so that the jet of water just clears the top of the wall.*

Solution. Given :

Height of wall = 8 m

Velocity of jet, $U = 20$ m/s

Distance of jet from wall, $x = 20$ m

Let the required angle = θ

Using equation (6.24), we have

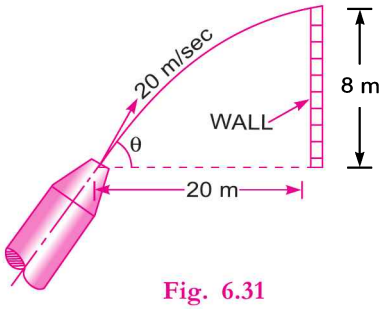


Fig. 6.31

$$y = x \tan\theta - \frac{gx^2}{2U^2} \sec^2\theta$$

where $y = 8$ m, $x = 20$ m, $U = 20$ m/s

$$8 = 20 \tan\theta - \frac{9.81 \times 20^2}{2 \times 20^2} \sec^2\theta$$

$$= 20 \tan\theta - 4.905 \sec^2\theta$$

$$= 20 \tan\theta - 4.905 [1 + \tan^2\theta] \qquad \{\because \sec^2\theta = 1 + \tan^2\theta\}$$

$$= 20 \tan\theta - 4.905 - 4.905 \tan^2\theta$$

or $4.905 \tan^2\theta - 20 \tan\theta + 8 + 4.905 = 0$

or $4.905 \tan^2\theta - 20 \tan\theta + 12.905 = 0$

$$\therefore \quad \tan\theta = \frac{20 \pm \sqrt{20^2 - 4 \times 12.905 \times 4.905}}{2 \times 4.905} = \frac{20 \pm \sqrt{400 - 253.19}}{9.81}$$

$$= \frac{20 \pm \sqrt{146.81}}{9.81} = \frac{20 \pm 12.116}{9.81} = \frac{32.116}{9.81} \text{ or } \frac{7.889}{9.81}$$

$\therefore$ $= 3.273$ or 0.8036

$\therefore$ $\theta = 73°\ 0.8'$ **or** $38°\ 37'$. **Ans.**

Problem 6.39 *A fire-brigade man is holding a fire stream nozzle of 50 mm diameter as shown in Fig. 6.32. The jet issues out with a velocity of 13 m/s and strikes the window. Find the angle or angles of inclination with which the jet issues from the nozzle. What will be the amount of water falling on the window ?*

Solution. Given :

Dia. of nozzle, $d = 50$ mm $= .05$ m

$\therefore$ Area, $A = \frac{\pi}{4}(.05)^2 = 0.001963$ m^2

Velocity of jet, $U = 13$ m/s.

The jet is coming out from nozzle at A. It strikes the window and let the angle made by the jet at A with horizontal is equal to θ.

The co-ordinates of window, with respect to origin at A.

$x = 5$ m, $y = 7.5 - 1.5 = 6.0$ m

The equation of the jet is given by (6.24) as

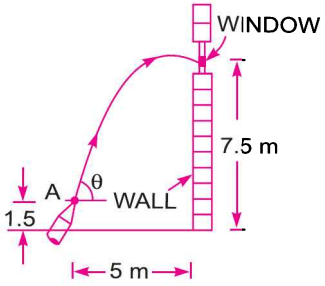


Fig. 6.32

$$y = x \tan\theta - \frac{gx^2}{2U^2}\sec^2\theta$$

or $$6.0 = 5 \times \tan\theta - \frac{9.81 \times 5}{2 \times 13^2}[1 + \tan^2\theta] \qquad \{\because \sec^2\theta = 1 + \tan^2\theta\}$$

or $$6.0 = 5\tan\theta - .7256\,(1 + \tan^2\theta)$$

$$= 5\tan\theta - .7256 - .7256\tan^2\theta$$

or $$0.7256\tan^2\theta - 5\tan\theta + 6 + .7256 = 0$$

or $$0.7256\tan^2\theta - 5\tan\theta + 6.7256 = 0$$

This is a quadratic equation in tan θ. Hence solution is

$$\tan\theta = \frac{5 \pm \sqrt{5^2 - 4 \times .7256 \times 6.7256}}{2 \times .7256}$$

$$= \frac{5 \pm \sqrt{25 - 19.52}}{1.4512} = \frac{5 + 2.341}{1.4512} = 5.058 \text{ or } 1.8322$$

$\therefore$ $\theta = \tan^{-1} 5.058$ or $\tan^{-1} 1.8322$ = **78.8° or 61.37°. Ans.**

Amount of water falling on window = Discharge from nozzle

= Area of nozzle × Velocity of jet at nozzle

= $0.001963 \times U = 0.001963 \times 13.0$ = **0.0255 m³/s. Ans.**

Problem 6.40 *A nozzle is situated at a distance of 1 m above the ground level and is inclined at an angle of 45° to the horizontal. The diameter of the nozzle is 50 mm and the jet of water from the nozzle strikes the ground at a horizontal distance of 4 m. Find the rate of flow of water.*

Solution. Given :

Distance of nozzle above ground = 1 m

Angle of inclination, $\theta = 45°$

Dia. of nozzle, $d = 50$ mm $= .05$ m

$\therefore$ Area, $A = \frac{\pi}{4}(.05)^2 = .001963$ m²

The horizontal distance $x = 4$ m

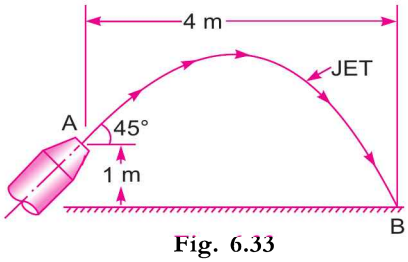


Fig. 6.33

The co-ordinates of the point *B*, which is on the centre-line of the jet of water and is situated on the ground, with respect to *A* (origin) are

$x = 4$ m and $y = -1.0$ m {From *A*, point *B* is vertically down by 1 m}

The equation of the jet is given by (6.24) as $y = x\tan\theta - \frac{gx^2}{2U^2}\sec^2\theta$

Substituting the known values as

$$-1.0 = 4\tan 45° - \frac{9.81 \times 4^2}{2U^2} \times \sec^2 45°$$

$$= 4 - \frac{78.48}{U^2} \times \left(\sqrt{2}\right)^2 \qquad \left\{\sec 45° = \frac{1}{\cos 45°} = \frac{1}{\frac{1}{\sqrt{2}}} = \sqrt{2}\right\}$$

$$-1.0 = 4 - \frac{78.48 \times 2}{U^2} \quad \text{or} \quad \frac{78.48 \times 2}{U^2} = +4.0 + 1.0 = 5.0$$

$$\therefore \quad U^2 = \frac{78.48 \times 2.0}{5.0} = 31.39$$

$$\therefore \quad U = \sqrt{31.39} = 5.60 \text{ m/s}$$

Now the rate of flow of fluid = Area × Velocity of jet

$= A \times U = .001963 \times 5.6 \text{ m}^3/\text{sec}$

$= 0.01099 \approx$ **.011 m³/s. Ans.**

Problem 6.41 *A window, in a vertical wall, is at a distance of 30 m above the ground level. A jet of water, issuing from a nozzle of diameter 50 mm is to strike the window. The rate of flow of water through the nozzle is 3.5 m³/minute and nozzle is situated at a distance of 1 m above ground level. Find the greatest horizontal distance from the wall of the nozzle so that jet of water strikes the window.*

Solution. Given :

Distance of window from ground level = 30 m

Dia. of nozzle, $d = 50 \text{ mm} = 0.05 \text{ m}$

$\therefore$ Area $A = \frac{\pi}{4}(.05)^2 = 0.001963 \text{ m}^2$

The discharge, $Q = 3.5 \text{ m}^3/\text{minute}$

$= \frac{3.5}{60} = 0.0583 \text{ m}^3/\text{s}$

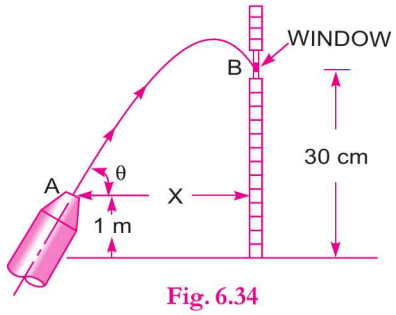


Fig. 6.34

Distance of nozzle from ground = 1 m.

Let the greatest horizontal distance of the nozzle from the wall = x and let angle of inclination = θ. If the jet reaches the window, then the point B on the window is on the centre-line of the jet. The co-ordinates of B with respect to A are

$$x = x, \; y = 30 - 1.0 = 29 \text{ m}$$

The velocity of jet, $U = \frac{\text{Discharge}}{\text{Area}} = \frac{Q}{A} = \frac{.0583}{.001963} = 29.69 \text{ m/sec}$

Using the equation (6.34), which is the equation of jet,

$$y = x \tan\theta - \frac{gx^2}{2U^2} \sec^2\theta$$

or
$$29.0 = x \tan\theta - \frac{9.81x^2}{2 \times (29.69)^2} \sec^2\theta$$

$$= x \tan\theta - 0.0055 \sec^2\theta \times x^2$$

$$= x \tan\theta - \frac{.0055\, x^2}{\cos^2\theta}$$

$$x \tan\theta - .0055\, x^2/\cos^2\theta - 29 = 0 \qquad ...(i)$$

The maximum value of x with respect to θ is obtained, by differentiating the above equation w.r.t. θ and substituting the value of $\frac{dx}{d\theta} = 0$. Hence differentiating the equation (i) w.r.t. θ, we have

$$\left[x \sec^2\theta + \tan\theta \times \frac{dx}{d\theta}\right] - 0.0055 \left[x^2 \times \left(\frac{(-2)}{\cos^3\theta}\right)(-\sin\theta) + \frac{1}{\cos^2\theta} \times 2x \frac{dx}{d\theta}\right]$$

$$\left\{\because \frac{d}{d\theta}(x\tan\theta) = x\sec^2\theta + \tan\theta\frac{dx}{d\theta} \text{ and } \frac{d}{d\theta}\left(\frac{x^2}{\cos^2\theta}\right) = x^2\frac{d}{d\theta}\left(\frac{1}{\cos^2\theta}\right) + \frac{1}{\cos^2\theta}\frac{d}{d\theta}(x^2)\right\}$$

$$\therefore \quad x\sec^2\theta + \tan\theta\frac{dx}{d\theta} - .0055\left[\frac{2x^2\sin\theta}{\cos^3\theta} + \frac{2x}{\cos^2\theta}\frac{dx}{d\theta}\right] = 0$$

For maximum value of x, w.r.t. θ, we have $\frac{dx}{d\theta} = 0$

Substituting this value in the above equation, we have

$$x\sec^2\theta - .0055\left[\frac{2x^2\sin\theta}{\cos^3\theta}\right] = 0$$

or $$\frac{x}{\cos^2\theta} - \frac{.0055\times 2x^2\sin\theta}{\cos^3\theta} = 0 \text{ or } x - .011\times x^2\frac{\sin\theta}{\cos\theta} = 0$$

or $$x - .011\,x^2\tan\theta = 0 \text{ or } 1 - .011\,x\tan\theta = 0$$

or $$x\tan\theta = \frac{1}{.011} = 90.9 \qquad ...(ii)$$

or $$x = \frac{90.9}{\tan\theta} \qquad ...(iii)$$

Substituting this value of x in equation (i), we get

$$\frac{90.9}{\tan\theta}\times\tan\theta - .0055\times\frac{(90.9)^2}{\tan^2\theta}\times\frac{1}{\cos^2\theta} - 29 = 0$$

$$90.9 - \frac{45.445}{\sin^2\theta} - 29 = 0 \text{ or } 61.9 - \frac{45.445}{\sin^2\theta} = 0$$

or $$61.9 = \frac{45.445}{\sin^2\theta} \text{ or } \sin^2\theta = \frac{45.445}{61.90} = 0.7341$$

$$\therefore \quad \sin\theta = \sqrt{0.7341} = 0.8568$$

$$\therefore \quad \theta = \tan^{-1}.8568 = 58^\circ\ 57.8'$$

Substituting this value of θ in equation (iii), we get

$$x = \frac{90.9}{\tan\theta} = \frac{90.9}{\tan 58^\circ 57.8'} = \frac{90.9}{\tan 58.95} = \frac{90.9}{1.66} = 54.759 \text{ m}$$

= 54.76 m. Ans.

HIGHLIGHTS

1. The study of fluid motion with the forces causing flow is called dynamics of fluid flow, which is analysed by the Newton's second law of motion.
2. Bernoulli's equation is obtained by integrating the Euler's equation of motion. Bernoulli's equation states "For a steady, ideal flow of an incompressible fluid, the total energy which consists of pressure energy, kinetic energy and datum energy, at any point of the fluid is constant". Mathematically,

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2$$

where $\frac{p_1}{\rho g}$ = pressure energy per unit weight = pressure head

$\frac{v_1^2}{2g}$ = kinetic energy per unit weight = kinetic head

z_1 = datum energy per unit weight = datum head.

3. Bernoulli's equation for real fluids

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2 + h_L$$

where h_L = loss of energy between sections 1 and 2.

4. The discharge, Q, through a venturimeter or an orifice meter is given by

$$Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$$

where a_1 = area at the inlet of venturimeter,
a_2 = area at the throat of venturimeter,
C_d = co-efficient of venturimeter,
h = difference of pressure head in terms of fluid head flowing through venturimeter.

5. The value of h is given by differential U-tube manometer

$$h = x\left[\frac{S_h}{S_o} - 1\right]$$ ... (when differential manometer contains heavier liquid)

$$h = x\left[1 - \frac{S_l}{S_o}\right]$$ ...(when differential manometer contains lighter liquid)

$$h = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = x\left[\frac{S_h}{S_o} - 1\right]$$ ...(for inclined venturimeter in which differential manometer contains heavier liquid)

$$h = \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = x\left[1 - \frac{S_l}{S_o}\right]$$ ...(for inclined venturimeter in which differential manometer contains lighter liquid)

where x = difference in the readings of differential manometer,
S_h = sp. gr. of heavier liquid
S_o = sp. gr. of fluid flowing through venturimeter
S_l = sp. gr. of lighter liquid.

6. Pitot-tube is used to find the velocity of a flowing fluid at any point in a pipe or a channel. The velocity is given by the relation

$$V = C_v \sqrt{2gh}$$

where C_v = co-efficient of Pitot-tube
h = rise of liquid in the tube above free surface of liquid

$$= x\left[\frac{S_g}{S_o} - 1\right] \text{ (for pipes or channels).}$$

7. The momentum equation states that the net force acting on a fluid mass is equal to the change in momentum per second in that direction. This is given as $F = \frac{d}{dt}(mv)$

The impulse-momentum equation is given by $F \cdot dt = d(mv)$.

8. The force exerted by a fluid on a pipe bend in the directions of x and y are given by

$$F_x = \frac{\text{mass}}{\text{sec}}(\text{Initial velocity in the direction of } x - \text{Final velocity in } x\text{-direction})$$
$$+ \text{Initial pressure force in } x\text{-direction} + \text{Final pressure force in } x\text{-direction}$$
$$= \rho Q[V_{1x} - V_{2x}] + (p_1A_1)_x + (p_2A_2)_x$$

and $$F_y = \rho Q[V_{1y} - V_{2y}] + (p_1A_1)_y + (p_2A_2)_y$$

Resultant force, $$F_R = \sqrt{F_x^2 + F_y^2}$$

and the direction of the resultant with horizontal is $\tan\theta = \frac{F_y}{F_x}$.

9. The force exerted by the nozzle on the water is given by $F_x = \rho Q[V_{2x} - V_{1x}]$
and force exerted by the water on the nozzle is $= -F_x = \rho Q[V_{1x} - V_{2x}]$.

10. Moment of momentum equation states that the resultant torque acting on a rotating fluid is equal to the rate of change of moment of momentum. Mathematically, it is given by $T = \rho Q[V_2r_2 - V_1r_1]$.

11. Free liquid jet is the jet of water issuing from a nozzle in atmosphere. The path travelled by the free jet is parabolic. The equation of the jet is given by

$$y = x\tan\theta - \frac{gx^2}{2U^2}\sec^2\theta$$

where x, y = co-ordinates of any point on jet w.r.t. to the nozzle
U = velocity of jet of water issuing from nozzle
θ = inclination of jet issuing from nozzle with horizontal.

12. (*i*) Maximum height attained by jet $= \frac{U^2\sin^2\theta}{2g}$

(*ii*) Time of flight, $T = \frac{2U\sin\theta}{g}$

(*iii*) Time to reach highest point, $T^* = \frac{T}{2} = \frac{U\sin\theta}{2g}$

(*iv*) Horizontal range of the jet, $x^* = \frac{U^2}{g}\sin 2\theta$

(*v*) Value of θ for maximum range, $\theta = 45°$

(*vi*) Maximum range, $x^*_{max} = U^2/g$.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Name the different forces present in a fluid flow. For the Euler's equation of motion, which forces are taken into consideration.
2. What is Euler's equation of motion ? How will you obtain Bernoulli's equation from it ?
3. Derive Bernoulli's equation for the flow of an incompressible frictionless fluid from consideration of momentum.
4. State Bernoulli's theorem for steady flow of an incompressible fluid. Derive an expression for Bernoulli's theorem from first principle and state the assumptions made for such a derivation.
5. What is a venturimeter ? Derive an expression for the discharge through a venturimeter.
6. Explain the principle of venturimeter with a neat sketch. Derive the expression for the rate of flow of fluid through it.
7. Discuss the relative merits and demerits of venturimeter with respect to orifice-meter.
(*Delhi University, Dec. 2002*)
8. Define an orifice-meter. Prove that the discharge through an orifice-meter is given by the relation

$$Q = C_d \frac{a_0 a_1}{\sqrt{a_1^2 - a_0^2}} \times \sqrt{2gh}$$

where a_1 = area of pipe in which orifice-meter is fitted
a_0 = area of orifice (*Technical University of M.P., S 2002*)
9. What is a pitot-tube ? How will you determine the velocity at any point with the help of pitot-tube ?
(*Delhi University, Dec. 2002*)
10. What is the difference between pitot-tube and pitot-static tube ?
11. State the momentum equation. How will you apply momentum equation for determining the force exerted by a flowing liquid on a pipe bend ?
12. What is the difference between momentum equation and impulse momentum equation.
13. Define moment of momentum equation. Where this equation is used.
14. What is a free jet of liquid ? Derive an expression for the path travelled by free jet issuing from a nozzle.
15. Prove that the equation of the free jet of liquid is given by the expression,

$$y = x \tan \theta - \frac{gx^2}{2U^2} \sec^2 \theta$$

where x, y = co-ordinates of a point on the jet
U = velocity of issuing jet
θ = inclination of the jet with horizontal.
16. Which of the following statement is correct in case of pipe flow :
(*a*) flow takes place from higher pressure to lower pressure ;
(*b*) flow takes place from higher velocity to lower velocity ;
(*c*) flow takes place from higher elevation to lower elevation ;
(*d*) flow takes place from higher energy to lower energy.
17. Derive Euler's equation of motion along a stream line for an ideal fluid stating clearly the assumptions. Explain how this is integrated to get Bernoulli's equation along a stream-line.
18. State Bernoulli's theorem. Mention the assumptions made. How is it modified while applying in practice? List out its engineering applications.
19. Define continuity equation and Bernoulli's equation.

20. What are the different forms of energy in a flowing fluid ? Represent schematically the Bernoulli's equation for flow through a tapering pipe and show the position of total energy line and the datum line.
21. Write Euler's equation of motion along a stream line and integrate it to obtain Bernoulli's equation. State all assumptions made.
22. Describe with the help of sketch the construction, operation and use of Pitot-static tube.
23. Starting with Euler's equation of motion along a stream line, obtain Bernoulli's equation by its integration. List all the assumptions made.
24. State the different devices that one can use to measure the discharge through a pipe and also through an open channel. Describe one of such devices with a neat sketch and explain how one can obtain the actual discharge with its help?
25. Derive Bernoulli's equation from fundamentals.

(B) NUMERICAL PROBLEMS

1. Water is flowing through a pipe of 100 mm diameter under a pressure of 19.62 N/cm^2 (gauge) and with mean velocity of 3.0 m/s. Find the total head of the water at a cross-section, which is 8 m above the datum line. [**Ans.** 28.458 m]
2. A pipe, through which water is flowing is having diameters 40 cm and 20 cm at the cross-sections 1 and 2 respectively. The velocity of water at section 1 is given 5.0 m/s. Find the velocity head at the sections 1 and 2 and also rate of discharge. [**Ans.** 1.274 m ; 20.387 m ; 0.628 m^3/s]
3. The water is flowing through a pipe having diameters 20 cm and 15 cm at sections 1 and 2 respectively. The rate of flow through pipe is 40 litres/s. The section 1 is 6 m above datum line and section 2 is 3 m above the datum. If the pressure at section 1 is 29.43 N/cm^2, find the intensity of pressure at section 2. [**Ans.** 32.19 N/cm^2]
4. Water is flowing through a pipe having diameters 30 cm and 15 cm at the bottom and upper end respectively. The intensity of pressure at the bottom end is 29.43 N/cm^2 and the pressure at the upper end is 14.715 N/cm^2. Determine the difference in datum head if the rate of flow through pipe is 50 lit/s. [**Ans.** 14.618 m]
5. The water is flowing through a taper pipe of length 50 m having diameters 40 cm at the upper end and 20 cm at the lower end, at the rate of 60 litres/s. The pipe has a slope of 1 in 40. Find the pressure at the lower end if the pressure at the higher level is 24.525 N/cm^2. [**Ans.** 25.58 N/cm^2]
6. A pipe of diameter 30 cm carries water at a velocity of 20 m/sec. The pressures at the points A and B are given as 34.335 N/cm^2 and 29.43 N/cm^2 respectively, while the datum head at A and B are 25 m and 28 m. Find the loss of head between A and B. [**Ans.** 2 m]
7. A conical tube of length 3.0 m is fixed vertically with its smaller end upwards. The velocity of flow at the smaller end is 4 m/s while at the lower end it is 2 m/s. The pressure head at the smaller end is 2.0 m of liquid. The loss of head in the tube is 0.95 $(v_1 - v_2)^2/2g$, where v_1 is the velocity at the smaller end and v_2 at the lower end respectively. Determine the pressure head at the lower end. Flow takes place in downward direction. [**Ans.** 5.56 m of fluid]
8. A pipe line carrying oil of specific gravity 0.8, changes in diameter from 300 mm at a position A to 500 mm diameter to a position B which is 5 m at a higher level. If the pressures at A and B are 19.62 N/cm^2 and 14.91 N/cm^2 respectively, and the discharge is 150 litres/s, determine the loss of head and direction of flow. [**Ans.** 1.45 m, Flow takes place from A to B]
9. A horizontal venturimeter with inlet and throat diameters 30 cm and 15 cm respectively is used to measure the flow of water. The reading of differential manometer connected to inlet and throat is 10 cm of mercury. Determine the rate of flow. Take $C_d = 0.98$. [**Ans.** 88.92 litres/s]

10. An oil of sp. gr. 0.9 is flowing through a venturimeter having inlet diameter 20 cm and throat diameter 10 cm. The oil-mercury differential manometer shows a reading of 20 cm. Calculate the discharge of oil through the horizontal venturimeter. Take $C_d = 0.98$. [**Ans.** 59.15 litres/s]

11. A horizontal venturimeter with inlet diameter 30 cm and throat diameter 15 cm is used to measure the flow of oil of sp. gr. 0.8. The discharge of oil through venturimeter is 50 litres/s, find the reading of the oil-mercury differential manometer. Take $C_d = 0.98$. [**Ans.** 2.489 cm]

12. A horizontal venturimeter with inlet diameter 20 cm and throat diameter 10 cm is used to measure the flow of water. The pressure at inlet is 14.715 N/cm^2 and vacuum pressure at the throat is 40 cm of mercury. Find the discharge of water through venturimeter. [**Ans.** 162.539 lit./s]

13. A 30 cm × 15 cm venturimeter is inserted in a vertical pipe carrying water, flowing in the upward direction. A differential mercury manometer connected to the inlet and throat gives a reading of 30 cm. Find the discharge. Take $C_d = 0.98$. [**Ans.** 154.02 lit/s]

14. If in the problem 13, instead of water, oil of sp. gr. 0.8 is flowing through the venturimeter, determine the rate of flow of oil in litres/s. [**Ans.** 173.56 lit/s]

15. The water is flowing through a pipe of diameter 30 cm. The pipe is inclined and a venturimeter is inserted in the pipe. The diameter of venturimeter at throat is 15 cm. The difference of pressure between the inlet and throat of the venturimeter is measured by a liquid of sp. gr. 0.8 in an inverted U-tube which gives a reading of 40 cm. The loss of head between the inlet and throat is 0.3 times the kinetic head of the pipe. Find the discharge. [**Ans.** 22.64 lit./s]

16. A 20 × 10 cm venturimeter is provided in a vertical pipe line carrying oil of sp. gr. 0.8, the flow being upwards. The difference in elevation of the throat section and entrance section of the venturimeter is 50 cm. The differential U-tube mercury manometer shows a gauge deflection of 40 cm. Calculate : (*i*) the discharge of oil, and (*ii*) the pressure difference between the entrance section and the throat section. Take $C_d = 0.98$ and sp. gr. of mercury as 13.6. [**Ans.** (*i*) 89.132 lit/s, (*ii*) 5.415 N/cm^2]

17. In a 200 mm diameter horizontal pipe a venturimeter of 0.5 contraction ratio has been fixed. The head of water on the venturimeter when there is no flow is 4 m (gauge). Find the rate of flow for which the throat pressure will be 4 metres of water absolute. Take $C_d = 0.97$ and atmospheric pressure head = 10.3 m of water. [**Ans.** 111.92 lit/s]

18. An orifice meter with orifice diameter 15 cm is inserted in a pipe of 30 cm diameter. The pressure gauges fitted upstream and downstream of the orifice meter give readings of 14.715 N/cm^2 and 9.81 N/cm^2 respectively. Find the rate of flow of water through the pipe in litres/s. Take $C_d = 0.6$. [**Ans.** 108.434 lit/s]

19. If in problem 18, instead of water, oil of sp. gr. 0.8 is flowing through the orifice meter in which the pressure difference is measured by a mercury oil differential manometer on the two sides of the orifice meter, find the rate of flow of oil when the reading of manometer is 40 cm. [**Ans.** 122.68 lit/s]

20. The pressure difference measured by the two tappings of a pitot-static tube, one tapping pointing upstream and other perpendicular to the flow, placed in the centre of a pipe line of diameter 40 cm is 10 cm of water. The mean velocity in the pipe is 0.75 times the central velocity. Find the discharge through the pipe. Take co-efficient of pitot-tube as 0.98. [**Ans.** 0.1293 m^3/s]

21. Find the velocity of flow of an oil through a pipe, when the difference of mercury level in a differential U-tube manometer connected to the two tappings of the pitot-tube is 15 cm. Take sp. gr. of oil = 0.8 and co-efficient of pitot-tube as 0.98. [**Ans.** 6.72 m/s]

22. A sub-marine moves horizontally in sea and has its axis 20 m below the surface of water. A pitot-static tube placed in front of sub-marine and along its axis, is connected to the two limbs of a U-tube containing mercury. The difference of mercury level is found to be 20 cm. Find the speed of sub-marine. Take sp. gr. of mercury 13.6 and of sea-water 1.026. [**Ans.** 24.958 km/hr.]

23. A 45° reducing bend is connected in a pipe line, the diameters at the inlet and outlet of the bend being 40 cm and 20 cm respectively. Find the force exerted by water on the bend if the intensity of pressure at inlet of bend is 21.58 N/cm^2. The rate of flow of water is 500 litres/s. [**Ans.** 22696.5 N ; 20° 3.5′]

24. The discharge of water through a pipe of diameter 40 cm is 400 litres/s. If the pipe is bend by 135°, find the magnitude and direction of the resultant force on the bend. The pressure of flowing water is 29.43 N/cm^2. [**Ans.** 7063.2 N, $\theta = 22° 29.9'$ with x-axis clockwise]

25. A 30 cm diameter pipe carries water under a head of 15 metres with a velocity of 4 m/s. If the axis of the pipe turns through 45°, find the magnitude and direction of the resultant force at the bend. [**Ans.** 8717.5 N, $\theta = 67° 30'$]

26. A pipe of 20 cm diameter conveying 0.20 m^3/sec of water has a right angled bend in a horizontal plane. Find the resultant force exerted on the bend if the pressure at inlet and outlet of the bend are 22.563 N/cm^2 and 21.582 N/cm^2 respectively. [**Ans.** 11604.7 N, $\theta = 43° 54.2'$]

27. A nozzle of diameter 30 mm is fitted to a pipe of 60 mm diameter. Find the force exerted by the nozzle on the water which is flowing through the pipe at the rate of 4.0 m^3/minute. [**Ans.** 7057.7 N]

28. A lawn sprinkler with two nozzles of diameters 3 mm each is connected across a tap of water. The nozzles are at a distance of 40 cm and 30 cm from the centre of the tap. The rate of water through tap is 100 cm^3/s. The nozzle discharges water in the downward directions. Determine the angular speed at which the sprinkler will rotate free. [**Ans.** 2.83 rad/s]

29. A lawn sprinkler has two nozzles of diameters 8 mm each at the end of a rotating arm and the velocity of flow of water from each nozzle is 12 m/s. One nozzle discharges water in the downward direction, while the other nozzle discharges water vertically up. The nozzles are at a distance of 40 cm from the centre of the rotating arm. Determine the torque required to hold the rotating arm stationary. Also determine the constant speed of rotation of arm, if it is free to rotate. [**Ans.** 5.78 Nm, 30 rad/s]

30. A vertical wall is of 10 m in height. A jet of water is issuing from a nozzle with a velocity of 25 m/s. The nozzle is situated at a horizontal distance of 20 m from the vertical wall. Find the angle of projection of the nozzle to the horizontal so that the jet of water just clears the tap of the wall. [**Ans.** 79° 55′ or 36° 41′]

31. A fire-brigade man is holding a fire stream nozzle of 50 mm diameter at a distance of 1 m above the ground and 6 m from a vertical wall. The jet is coming out with a velocity of 15 m/s. This jet is to strike a window, situated at a distance of 10 m above ground in the vertical wall. Find the angle or angles of inclination with the horizontal made by the jet, coming out from the nozzle. What will be the amount of water falling on the window ? [**Ans.** 79° 16.7′ or 67° 3.7′ ; 0.0294 m^3/s]

32. A window, in a vertical wall, is at a distance of 12 m above the ground level. A jet of water, issuing from a nozzle of diameter 50 mm, is to strike the window. The rate of flow of water through the nozzle is 40 litres/sec. The nozzle is situated at a distance of 1 m above ground level. Find the greatest horizontal distance from the wall of the nozzle so that jet of water strikes the window. [**Ans.** 29.38 m]

33. Explain in brief the working of a pitot-tube. Calculate the velocity of flow of water in a pipe of diameter 300 mm at a point, where the stagnation pressure head is 5 m and static pressure head is 4 m. Given the co-efficient of pitot-tube = 0.97. [**Ans.** 4.3 m/sec]

34. Find the rate of flow of water through a venturimeter fitted in a pipeline of diameter 30 cm. The ratio of diameter of throat and inlet of the venturimeter is *. The pressure at the inlet of the venturimeter is 13.734 N/cm^2 (gauge) and vacuum in the throat is 37.5 cm of mercury. The co-efficient of venturimeter is given as 0.98. [**Ans.** 0.15 m^3/s]

35. A 30 cm × 15 cm venturimeter is inserted in a vertical pipe carrying an oil of sp. gr. 0.8, flowing in the upward direction. A differential mercury manometer connected to the inlet and throat gives a reading of 30 cm. The difference in the elevation of the throat section and inlet section is 50 cm. Find the rate of flow of oil.

36. A venturimeter is used for measurement of discharge of water in horizontal pipe line. If the ratio of upstream pipe diameter to that of throat is 2 : 1, upstream diameter is 300 mm, the difference in pressure between the throat and upstream is equal to 3 m head of water and loss of head through meter is one-eighth of the throat velocity head, calculate the discharge in the pipe. [**Ans.** 0.107 m^3/s]

37. A liquid of specific gravity 0.8 is flowing upwards at the rate of 0.08 m^3/s, through a vertical venturimeter with an inlet diameter of 200 mm and throat diameter of 100 mm. The C_d = 0.98 and the vertical distance between pressure tappings is 300 mm. Find :

(*i*) the difference in readings of the two pressure gauges, which are connected to the two pressure tappings, and

(*ii*) the difference in the level of the mercury columns of the differential manometer which is connected to the tappings, in place of pressure gauges. [**Ans.** (*i*) 42.928 kN/m^2, (*ii*) 32.3 cm]

[**Hint.** $Q = 0.08$ m^3/s, $d_1 = 200$ mm $= 0.2$ m, $d_2 = 100$ mm $= 0.1$ m,

$$C_d = 0.98,\ z_2 - z_1 = 300 \text{ mm} = 0.3 \text{ m},\ a_1 = \frac{\pi}{4}(.2^2) = 0.0314 \text{ m}^2$$

$$a_2 = \frac{\pi}{4}(.1^2) = 0.007854 \text{ m}^2. \text{ Using } Q = C_d \frac{a_1 \times a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$$

Find 'h'. This value of $h = 5.17$ m.

Now use $h = \left(\frac{p_1}{\rho g} - \frac{p_2}{\rho g}\right) + (z_1 - z_2)$, where $\rho = 800$ kg/m^3. Find $(p_1 - p_2)$.

Now use the formula $h = x\left[\frac{S_g}{S_f} - 1\right]$,

where $h = 5.17$ m, $S_g = 13.6$ and $S_f = 0.8$. Find the value of x which will be 32.3 cm.]

38. A venturimeter is installed in a 300 mm diameter horizontal pipe line. The throat pipe rates is 1/3. Water flows through the installation. The pressure in the pipe line is 13.783 N/cm^2 (gauge) and vacuum in the throat is 37.5 cm of mercury. Neglecting head loss in the venturimeter, determine the rate of flow in the pipe line. [**Ans.** 0.153 m^3/sec]

[**Hint.** $d_1 = 300$ mm $= 0.3$ m, $d_2 = \frac{1}{3} \times 300 = 100$ mm $= 0.1$ m, $p_1 = 13.783$ N/cm^2 $= 13.783 \times 10^4$ N/m^2.

Hence $p_1/\rho \times g = 13.783 \times 10^4/1000 \times 9.81$

$= 14.05$ m, $p_2/\rho g = -37.5$ cm of Hg $= -0.375 \times 13.6$ m of water

$= -5.1$ m of water. Hence $h = 14.05 - (-5.1) = 19.15$ m of water.

Value of $C_d = 1.0$. Now use the formula $Q = C_d \frac{a_1 a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$]

39. The maximum flow through a 300 mm diameter horizontal main pipe line is 18200 litre/minute. A venturimeter is introduced at a point of the pipe line where the pressure head is 4.6 m of water. Find the smallest dia. of throat so that the pressure at the throat is never negative. Assume co-efficient of meter as unity. [**Ans.** $d_2 = 192.4$ mm]

[**Hint.** $d_1 = 300$ mm $= 0.3$ m, $Q = 18200$ litres/minute $= 18200/60 = 303.33$ litres/s $= 0.3033$ m^3/s, $p_1/\rho g = 4.6$ m, $p_2/\rho g = 0$. Hence $h = 4.6$ m, $C_d = 1$. d_2 = dia. at throat. Use formula $Q = C_d \frac{a_1 \times a_2}{\sqrt{a_1^2 - a_2^2}} \times \sqrt{2gh}$ and find the value of a_2 . Then $a_2 = \frac{\pi}{4} d_2^2$ and find d_2.]

40. The following are the data given of a change in diameter effected in laying a water supply pipe. The change in diameter is gradual from 20 cm at A to 50 cm at B. Pressures at A and B are 7.848 N/cm^2 and 5.886 N/cm^2 respectively with the end B being 3 m higher than A. If the flow in the pipe line is 200 litre/s, find : (*i*) direction of flow, (*ii*) the head lost in friction between A and B.

[**Ans.** (*i*) From A to B, (*ii*) 1.015 m]

[**Hint.** $D_A = 20$ cm $= 0.2$ m, $D_B = 50$ cm $= 0.5$ m, $p_A = 7.848$ N/cm^2 $= 7.848 \times 10^4$ N/m^2

$p_B = 5.886$ N/cm^2 $= 5.886 \times 10^4$ N/m^2, $Z_A = 0$, $Z_B = 3$ m, $Q = 0.2$ m^3/s

$V_A = 0.2/\frac{\pi}{4}(.2^2) = 6.369$ m/s, $V_B = 0.2/\frac{\pi}{4}(.5^2) = 1.018$ m/s

$E_A = (p_A/\rho \times g) + \frac{V_A^2}{2g} + Z_A = (7.848 \times 10^4/1000 \times 9.81) + (6.369^2/2 \times 9.81) + 0 = 10.067$ m

$E_B = (p_B/\rho \times g) + \frac{V_B^2}{2g} + Z_B = (5.886 \times 10^4/1000 \times 9.81) + (1.018^2/2 \times 9.81) + 3 = 9.052$ m]

41. A venturimeter of inlet diameter 300 mm and throat diameter 150 mm is fixed in a vertical pipe line. A liquid of sp. gr. 0.8 is flowing upward through the pipe line. A differential manometer containing mercury gives a reading of 100 mm when connected at inlet and throat. The vertical difference between inlet and throat is 500 mm. If $C_d = 0.98$, then find : (*i*) rate of flow of liquid in litre per second and (*ii*) difference of pressure between inlet and throat in N/m^2. [**Ans.** (*i*) 100 litre/s, (*ii*) 15980 N/m^2]

42. A venturimeter with a throat diameter of 7.5 cm is installed in a 15 cm diameter pipe. The pressure at the entrance to the meter is 70 kPa (gauge) and it is desired that the pressure at any point should not fall below 2.5 m of absolute water. Determine the maximum flow rate of water through the meter. Take $C_d = 0.97$ and atmospheric pressure as 100 kPa. (*J.N.T.U., Hyderabad S 2002*)

[**Hint.** The pressure at the throat will be minimum. Hence $\frac{p_2}{\rho g} = 2.5$ m (abs.)

Given : $d_1 = 15$ cm $\therefore A_1 = \frac{\pi}{4}(15^2) = 176.7$ cm^2

$d_2 = 7.5$ cm $\therefore A_2 = \frac{\pi}{4}(7.5^2) = 44.175$ cm^2

$p_1 = 70$ kPa $= 70 \times 10^3$ N/m^2 (gauge), $p_{atm} = 100$ kPa $= 100 \times 10^3$ N/m^2

$\therefore$ p_1 (abs.) $= 70 \times 10^3 + 100 \times 10^3 = 170 \times 10^3$ N/m^2 (abs.)

$\therefore$ $\frac{p_1}{\rho g} = \frac{170 \times 10^3}{1000 \times 9.81} = 17.33$ m of water (abs.)

$\therefore$ $h = \frac{p_1}{\rho g} - \frac{p_2}{\rho g} = 17.33 - 2.5 = 14.83$ m of water $= 1483$ cm of water

Now $Q = \frac{C_d A_1 A_2}{\sqrt{A_1^2 - A_2^2}} \times \sqrt{2gh} = \frac{0.97 \times 176.7 \times 44.175 \times \sqrt{2 \times 981 \times 1483}}{\sqrt{176.7^2 - 44 \cdot 175^2}} = 75488$ cm^3/s

= 75.488 litre/s.]

43. Find the discharge of water flowing through a pipe 20 cm diameter placed in an inclined position, where a venturimeter is inserted, having a throat diameter of 10 cm. The difference of pressure between the main and throat is measured by a liquid of specific gravity 0.4 in an inverted *U*-tube, which gives a reading of 30 cm. The loss of head between the main and throat is 0.2 times the kinetic head of pipe. (*Delhi University, Dec. 2002*)

[**Hint.** Given : $d_1 = 20$ cm $\therefore a_1 = \frac{\pi}{4}(20^2) = 100\,\pi$ cm^2 ; $d_2 = 10$ cm $\therefore a_2 = \frac{\pi}{4}(10^2) = 25\,\pi$ cm^2.

$x = 30$ cm, $h = x\left(1 - \frac{S_l}{S_o}\right) = 30\left(1 - \frac{0.4}{1.0}\right) = 18$ cm $= 0.18$ m

But h is also $= \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) \therefore \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = 18$ cm $= 0.18$ m

$$h_L = 0.2 \times \frac{V_1^2}{2g}$$

From Bernoulli's equation, $\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + h_L$

or $\left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) + \frac{V_1^2}{2g} - \frac{V_2^2}{2g} = h_L$

or $0.18 + \frac{V_1^2}{2g} - \frac{V_2^2}{2g} = \frac{0.2\,V_1^2}{2g}$ $\quad \left(\because \left(\frac{p_1}{\rho g} + z_1\right) - \left(\frac{p_2}{\rho g} + z_2\right) = 0.18 \text{ m and } h_L = \frac{0.2\,V_1^2}{2g}\right)$

or $0.18 + \frac{V_1^2}{2g} - \frac{V_2^2}{2g} - \frac{0.2\,V_1^2}{2g} = 0$ or $0.18 + \frac{0.8\,V_1^2}{2g} - \frac{V_2^2}{2g} = 0$

From continuity equation, $a_1V_1 = a_2V_2$ or $V_2 = \frac{a_1V_1}{a_2} = \frac{\frac{\pi}{4}\left(20^2\right)V_1}{\frac{\pi}{4}\left(10^2\right)} = 4V_1$

Now $0.18 + \frac{0.8\,V_1^2}{2g} - \frac{V_2^2}{2g} = 0$ or $0.18 + \frac{0.8\,V_1^2}{2g} - \frac{(4V_1)^2}{2g} = 0$

or $0.18 + \frac{0.8\,V_1^2}{2g} - \frac{16V_1^2}{2g} = 0$ or $0.18 = \frac{16V_1^2}{2g} - \frac{0.8\,V_1^2}{2g} = \frac{15.2V_1^2}{2g}$

$\therefore$ $V_1 = \sqrt{\frac{0.18 \times 2 \times 9.81}{15.2}} = 0.48 \text{ m/s} = 48 \text{ cm/s}$

$\therefore$ $Q = A_1V_1 = \frac{\pi}{4}(20^2) \times 48 = 15140 \text{ cm}^3\text{/s} =$ **15.14 litre/s.]**

7 CHAPTER

ORIFICES AND MOUTHPIECES

▶ 7.1 INTRODUCTION

Orifice is a small opening of any cross-section (such as circular, triangular, rectangular etc.) on the side or at the bottom of a tank, through which a fluid is flowing. A mouthpiece is a short length of a pipe which is two to three times its diameter in length, fitted in a tank or vessel containing the fluid. Orifices as well as mouthpieces are used for measuring the rate of flow of fluid.

▶ 7.2 CLASSIFICATIONS OF ORIFICES

The orifices are classified on the basis of their size, shape, nature of discharge and shape of the upstream edge. The following are the important classifications :

1. The orifices are classified as **small orifice** or **large orifice** depending upon the size of orifice and head of liquid from the centre of the orifice. If the head of liquid from the centre of orifice is more than five times the depth of orifice, the orifice is called small orifice. And if the head of liquids is less than five times the depth of orifice, it is known as large orifice.

2. The orifices are classified as (*i*) Circular orifice, (*ii*) Triangular orifice, (*iii*) Rectangular orifice and (*iv*) Square orifice depending upon their cross-sectional areas.

3. The orifices are classified as (*i*) Sharp-edged orifice and (*ii*) Bell-mouthed orifice depending upon the shape of upstream edge of the orifices.

4. The orifices are classified as (*i*) Free discharging orifices and (*ii*) Drowned or sub-merged orifices depending upon the nature of discharge.

The sub-merged orifices are further classified as (*a*) Fully sub-merged orifices and (*b*) Partially sub-merged orifices.

▶ 7.3 FLOW THROUGH AN ORIFICE

Consider a tank fitted with a circular orifice in one of its sides as shown in Fig. 7.1. Let H be the head of the liquid above the centre of the orifice. The liquid flowing through the orifice forms a jet of liquid whose area of cross-section is less than that of orifice. The area of jet of fluid goes on decreasing and at a section C-C, the area is minimum. This section is approximately at a distance of half of diameter of the orifice. At this section, the streamlines are straight and parallel to each other and perpendicular to the

plane of the orifice. This section is called **Vena-contracta**. Beyond this section, the jet diverges and is attracted in the downward direction by the gravity.

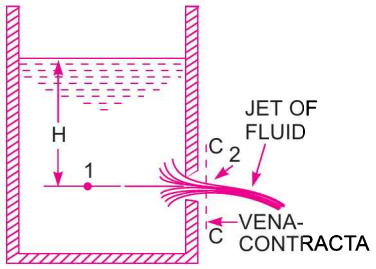


Fig. 7.1 *Tank with an orifice.*

Consider two points 1 and 2 as shown in Fig. 7.1. Point 1 is inside the tank and point 2 at the vena-contracta. Let the flow is steady and at a constant head H. Applying Bernoulli's equation at points 1 and 2.

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2$$

But $$z_1 = z_2$$

$$\therefore \quad \frac{p_1}{\rho g} + \frac{v_1^2}{2g} = \frac{p_2}{\rho g} + \frac{v_2^2}{2g}$$

Now $$\frac{p_1}{\rho g} = H$$

$$\frac{p_2}{\rho g} = 0 \text{ (atmospheric pressure)}$$

v_1 is very small in comparison to v_2 as area of tank is very large as compared to the area of the jet of liquid.

$$\therefore \quad H + 0 = 0 + \frac{v_2^2}{2g}$$

$$\therefore \quad v_2 = \sqrt{2gH} \qquad ...(7.1)$$

This is theoretical velocity. Actual velocity will be less than this value.

▶ 7.4 HYDRAULIC CO-EFFICIENTS

The hydraulic co-efficients are

1. Co-efficient of velocity, C_v
2. Co-efficient of contraction, C_c
3. Co-efficient of discharge, C_d.

7.4.1 Co-efficient of Velocity (C_v). It is defined as the ratio between the actual velocity of a jet of liquid at vena-contracta and the theoretical velocity of jet. It is denoted by C_v and mathematically, C_v is given as

$$C_v = \frac{\text{Actual velocity of jet at vena-contracta}}{\text{Theoretical velocity}}$$

$$= \frac{V}{\sqrt{2gH}}, \text{ where } V = \text{actual velocity, } \sqrt{2gH} = \text{Theoretical velocity} \qquad ...(7.2)$$

The value of C_v varies from 0.95 to 0.99 for different orifices, depending on the shape, size of the orifice and on the head under which flow takes place. Generally the value of $C_v = 0.98$ is taken for sharp-edged orifices.

7.4.2 Co-efficient of Contraction (C_c). It is defined as the ratio of the area of the jet at vena-contracta to the area of the orifice. It is denoted by C_c.

Let a = area of orifice and

a_c = area of jet at vena-contracta.

Then
$$C_c = \frac{\text{area of jet at vena-contracta}}{\text{area of orifice}}$$

$$= \frac{a_c}{a} \qquad ...(7.3)$$

The value of C_c varies from 0.61 to 0.69 depending on shape and size of the orifice and head of liquid under which flow takes place. In general, the value of C_c may be taken as 0.64.

7.4.3 Co-efficient of Discharge (C_d). It is defined as the ratio of the actual discharge from an orifice to the theoretical discharge from the orifice. It is denoted by C_d. If Q is actual discharge and Q_{th} is the theoretical discharge then mathematically, C_d is given as

$$C_d = \frac{Q}{Q_{th}} = \frac{\text{Actual velocity} \times \text{Actual area}}{\text{Theoretical velocity} \times \text{Theoretical area}}$$

$$= \frac{\text{Actual velocity}}{\text{Theoretical velocity}} \times \frac{\text{Actual area}}{\text{Theoretical area}}$$

$$\therefore \quad C_d = C_v \times C_c \qquad ...(7.4)$$

The value of C_d varies from 0.61 to 0.65. For general purpose the value of C_d is taken as 0.62.

Problem 7.1 *The head of water over an orifice of diameter 40 mm is 10 m. Find the actual discharge and actual velocity of the jet at vena-contracta. Take C_d = 0.6 and C_v = 0.98.*

Solution. Given :

Head, H = 10 cm

Dia. of orifice, d = 40 mm = 0.04 m

$\therefore$ Area, $a = \frac{\pi}{4}(.04)^2 = .001256 \text{ m}^2$

$C_d = 0.6$

$C_v = 0.98$

(*i*) $\frac{\text{Actual discharge}}{\text{Theoretical discharge}} = 0.6$

But Theoretical discharge = $V_{th} \times$ Area of orifice

V_{th} = Theoretical velocity, where $V_{th} = \sqrt{2gH} = \sqrt{2 \times 9.81 \times 10} = 14$ m/s

$\therefore$ Theoretical discharge = $14 \times .001256 = 0.01758 \frac{\text{m}^2}{\text{s}}$

$\therefore$ Actual discharge = 0.6 × Theoretical discharge

= $0.6 \times .01758 =$ **0.01054 m^3/s. Ans.**

(*ii*) $\dfrac{\text{Actual velocity}}{\text{Theoretical velocity}} = C_v = 0.98$

$\therefore$ Actual velocity = 0.98 × Theoretical velocity
= 0.98 × 14 = **13.72 m/s. Ans.**

Problem 7.2 *The head of water over the centre of an orifice of diameter 20 mm is 1 m. The actual discharge through the orifice is 0.85 litre/s. Find the co-efficient of discharge.*

Solution. Given :

Dia. of orifice, $d = 20 \text{ mm} = 0.02 \text{ m}$

$\therefore$ Area, $a = \dfrac{\pi}{4}(0.02)^2 = 0.000314 \text{ m}^2$

Head, $H = 1 \text{ m}$

Actual discharge, $Q = 0.85 \text{ litre/s} = 0.00085 \text{ m}^3/\text{s}$

Theoretical velocity, $V_{th} = \sqrt{2gH} = \sqrt{2 \times 9.81 \times 1} = 4.429 \text{ m/s}$

$\therefore$ Theoretical discharge, $Q_{th} = V_{th} \times$ Area of orifice
$= 4.429 \times 0.000314 = 0.00139 \text{ m}^3/\text{s}$

$\therefore$ Co-efficient of discharge = $\dfrac{\text{Actual discharge}}{\text{Theoretical discharge}} = \dfrac{0.00085}{0.00139}$ = **0.61. Ans.**

▶ 7.5 EXPERIMENTAL DETERMINATION OF HYDRAULIC CO-EFFICIENTS

7.5.1 Determination of Co-efficient of Discharge (C_d). The water is allowed to flow through an orifice fitted to a tank under a constant head, H as shown in Fig. 7.2. The water is collected in a measuring tank for a known time, t. The height of water in the measuring tank is noted down. Then actual discharge through orifice,

$$Q = \frac{\text{Area of measuring tank} \times \text{Height of water in measuring tank}}{\text{Time } (t)}$$

and theoretical discharge = area of orifice × $\sqrt{2gH}$

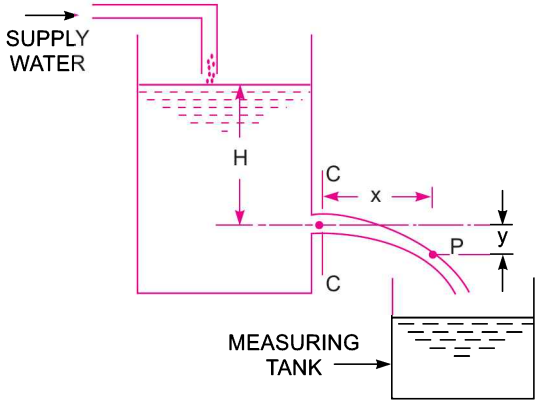


Fig. 7.2 *Value of C_d.*

$\therefore$ $$C_d = \frac{Q}{a \times \sqrt{2gH}} \qquad \ldots(7.5)$$

7.5.2 Determination of Co-efficient of Velocity (C_v). Let *C-C* represents the vena-contracta of a jet of water coming out from an orifice under constant head *H* as shown in Fig. 7.2. Consider a liquid particle which is at vena-contracta at any time and takes the position at *P* along the jet in time '*t*'.

Let x = horizontal distance travelled by the particle in time '*t*'

y = vertical distance between *P* and *C-C*

V = actual velocity of jet at vena-contracta.

Then horizontal distance, $x = V \times t$ ...(*i*)

and vertical distance, $y = \frac{1}{2} g t^2$ ...(*ii*)

From equation (*i*), $t = \frac{x}{V}$

Substituting this value of '*t*' in (*ii*), we get

$$y = \frac{1}{2} g \times \frac{x^2}{V^2}$$

$$V^2 = \frac{g x^2}{2y}$$

$$\therefore \quad V = \sqrt{\frac{g x^2}{2y}}$$

But theoretical velocity,

$$V_{th} = \sqrt{2gH}$$

$$\therefore \text{ Co-efficient of velocity, } C_v = \frac{V}{V_{th}} = \sqrt{\frac{g x^2}{2y}} \times \frac{1}{\sqrt{2gH}} = \sqrt{\frac{x^2}{4yH}}$$

$$= \frac{x}{\sqrt{4yH}}. \qquad ...(7.6)$$

7.5.3 Determination of Co-efficient of Contraction (C_c). The co-efficient of contraction is determined from the equation (7.4) as

$$C_d = C_v \times C_c$$

$$\therefore \quad C_c = \frac{C_d}{C_v} \qquad ...(7.7)$$

Problem 7.3 *A jet of water, issuing from a sharp-edged vertical orifice under a constant head of 10.0 cm, at a certain point, has the horizontal and vertical co-ordinates measured from the vena-contracta as 20.0 cm and 10.5 cm respectively. Find the value of C_v. Also find the value of C_c if $C_d = 0.60$.*

Solution. Given :

Head, $H = 10.0$ cm

Horizontal distance, $x = 20.0$ cm

Vertical distance, $y = 10.5$ cm

$C_d = 0.6$

The value of C_v is given by equation (7.6) as

$$C_v = \frac{x}{\sqrt{4yH}} = \frac{20.0}{\sqrt{4 \times 10.5 \times 10.0}} = \frac{20}{20.493} = 0.9759 = \mathbf{0.976.\ Ans.}$$

The value of C_c is given by equation (7.7) as

$$C_c = \frac{C_d}{C_v} = \frac{0.6}{0.976} = 0.6147 = \mathbf{0.615.\ Ans.}$$

Problem 7.4 *The head of water over an orifice of diameter 100 mm is 10 m. The water coming out from orifice is collected in a circular tank of diameter 1.5 m. The rise of water level in this tank is 1.0 m in 25 seconds. Also the co-ordinates of a point on the jet, measured from vena-contracta are 4.3 m horizontal and 0.5 m vertical. Find the co-efficients, C_d, C_v and C_c.*

Solution. Given :

Head, $H = 10$ m

Dia. of orifice, $d = 100$ mm $= 0.1$ m

$\therefore$ Area of orifice, $a = \frac{\pi}{4}(.1)^2 = 0.007853$ m^2

Dia. of measuring tank, $D = 1.5$ m

$\therefore$ Area, $A = \frac{\pi}{4}(1.5)^2 = 1.767$ m^2

Rise of water, $h = 1$ m

Time, $t = 25$ seconds

Horizontal distance, $x = 4.3$ m

Vertical distance, $y = 0.5$ m

Now theoretical velocity, $V_{th} = \sqrt{2gH} = \sqrt{2 \times 9.81 \times 10} = 14.0$ m/s

$\therefore$ Theoretical discharge, $Q_{th} = V_{th} \times$ Area of orifice $= 14.0 \times 0.007854 = 0.1099$ m^3/s

Actual discharge, $Q = \frac{A \times h}{t} = \frac{1.767 \times 1.0}{25} = 0.07068$

$\therefore$ $$C_d = \frac{Q}{Q_{th}} = \frac{0.07068}{0.1099} = \mathbf{0.643.\ Ans.}$$

The value of C_v is given by equation (7.6) as

$$C_v = \frac{x}{\sqrt{4yH}} = \frac{4.3}{\sqrt{4 \times 0.5 \times 10}} = \frac{4.3}{4.472} = \mathbf{0.96.\ Ans.}$$

C_c is given by equation (7.7) as $C_c = \frac{C_d}{C_v} = \frac{0.643}{0.96} = \mathbf{0.669.\ Ans.}$

Problem 7.5 *Water discharge at the rate of 98.2 litres/s through a 120 mm diameter vertical sharp-edged orifice placed under a constant head of 10 metres. A point, on the jet, measured from the vena-contracta of the jet has co-ordinates 4.5 metres horizontal and 0.54 metres vertical. Find the co-efficient C_v, C_c and C_d of the orifice.*

Solution. Given :

Discharge, $Q = 98.2$ lit/s $= 0.0982$ m^3/s

Dia. of orifice, $d = 120$ mm $= 0.12$ m

$\therefore$ Area of orifice, $a = \frac{\pi}{4}(0.12)^2 = 0.01131$ m^2

Head, $H = 10$ m

Horizontal distance of a point on the jet from vena-contracta, $x = 4.5$ m and vertical distance, $y = 0.54$ m

Now theoretical velocity, $V_{th} = \sqrt{2g \times H} = \sqrt{2 \times 9.81 \times 10} = 14.0$ m/s

Theoretical discharge, $Q_{th} = V_{th} \times$ Area of orifice

$= 14.0 \times 0.01131 = 0.1583$ m^3/s

The value of C_d is given by, $C_d = \frac{\text{Actual discharge}}{\text{Theoretical discharge}} = \frac{Q}{Q_{th}} = \frac{0.0982}{0.1583} =$ **0.62. Ans.**

The value of C_c is given by equation (7.6),

$$C_v = \frac{x}{\sqrt{4yH}} = \frac{4.5}{\sqrt{4 \times 0.54 \times 10}} = \textbf{0.968. Ans.}$$

The value of C_c is given by equation (7.7) as

$$C_c = \frac{C_d}{C_v} = \frac{0.62}{0.968} = \textbf{0.64. Ans.}$$

Problem 7.6 *A 25 mm diameter nozzle discharges 0.76 m^3 of water per minute when the head is 60 m. The diameter of the jet is 22.5 mm. Determine : (i) the values of co-efficients C_c, C_v and C_d and (ii) the loss of head due to fluid resistance.*

Solution. Given :

Dia. of nozzle, $D = 25$ mm $= 0.025$ m

Actual discharge, $Q_{act} = 0.76$ m^3/minute $= \frac{0.76}{60} = 0.01267$ m^3/s

Head, $H = 60$ m

Dia. of jet, $d = 22.5$ mm $= 0.0225$ m.

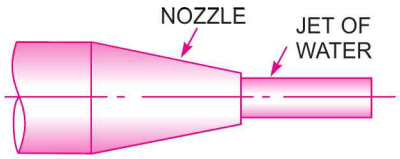


Fig. 7.3

(*i*) *Values of co-efficients :*

Co-efficient of contraction (C_c) is given by,

$$C_c = \frac{\text{Area of jet}}{\text{Area of nozzle}}$$

$$= \frac{\frac{\pi}{4}d^2}{\frac{\pi}{4}D^2} = \frac{d^2}{D^2} = \frac{0.0225^2}{0.025^2} = \textbf{0.81. Ans.}$$

Co-efficient of discharge (C_d) is given by,

$$C_d = \frac{\text{Actual discharge}}{\text{Theoretical discharge}}$$

$$= \frac{0.01267}{\text{Theoretical velocity} \times \text{Area of nozzle}}$$

$$= \frac{0.01267}{\sqrt{2gH} \times \frac{\pi}{4} D^2} = \frac{0.01267}{\sqrt{2 \times 9.81 \times 60} \times \frac{\pi}{4} (0.025)^2}$$

$$= \mathbf{0.752.\ Ans.}$$

Co-efficient of velocity (C_v) is given by,

$$C_v = \frac{C_d}{C_c} = \frac{0.752}{0.81} = \mathbf{0.928.\ Ans.}$$

(*ii*) *Loss of head due to fluid resistance* :

Applying Bernoulli's equation at the outlet of nozzle and to the jet of water, we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + \text{Loss of head}$$

But $\frac{p_1}{\rho g} = \frac{p_2}{\rho g}$ = Atmospheric pressure head

$z_1 = z_2$, $V_1 = \sqrt{2gH}$, V_2 = Actual velocity of jet = $C_v \sqrt{2gH}$

$$\therefore \quad \frac{\left(\sqrt{2gH}\right)^2}{2g} = \frac{\left(C_v \sqrt{2gH}\right)^2}{2g} + \text{Loss of head}$$

or $H = C_v^2 \times H + \text{Loss of head}$

$\therefore$ Loss of head $= H - C_v^2 \times H = H(1 - C_v^2)$

$= 60(1 - 0.928^2) = 60 \times 0.1388 = \mathbf{8.328\ m.\ Ans.}$

Problem 7.7 *A pipe, 100 mm in diameter, has a nozzle attached to it at the discharge end, the diameter of the nozzle is 50 mm. The rate of discharge of water through the nozzle is 20 litres/s and the pressure at the base of the nozzle is 5.886 N/cm^2. Calculate the co-efficient of discharge. Assume that the base of the nozzle and outlet of the nozzle are at the same elevation.*

Solution. Given :

Dia. of pipe, $D = 100$ mm $= 0.1$ m

$\therefore \quad A_1 = \frac{\pi}{4}(.1)^2 = .007854$ m^2

Dia. of nozzle, $d = 50$ mm $= 0.05$ m

$\therefore \quad A_2 = \frac{\pi}{4}(.05)^2 = .001963$ m^2

Actual discharge, $Q = 20$ lit/s $= 0.02$ m^3/s

Pressure at the base, $p_1 = 5.886$ N/cm^2 $= 5.886 \times 10^4 \frac{\text{N}}{\text{m}^2}$

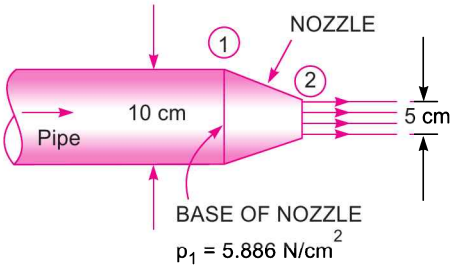


Fig. 7.4

From continuity equation, $A_1V_1 = A_2V_2$

or $.007854\ V_1 = .001963\ V_2$

$$\therefore \qquad V_1 = \frac{.001963 V_2}{.007854} = \frac{V_2}{4}$$

where V_1 and V_2 are theoretical velocity at sections (1) and (2).

Applying Bernoulli's equation at sections (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2$$

But $z_1 = z_2$

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g}$$

or $$\frac{5.886 \times 10^4}{1000 \times 9.81} + \frac{\left(\frac{V_2}{4}\right)^2}{2g} = 0 + \frac{V_2^2}{2g} \qquad \left\{\because \frac{p_2}{\rho g} = \text{Atmospheric pressure} = 0\right\}$$

$$6.0 + \frac{V_2^2}{2g \times 16} = \frac{V_2^2}{2g}$$

or $$\frac{V_2^2}{2g}\left[1 - \frac{1}{16}\right] = 6.0 \quad \text{or} \quad \frac{V_2^2}{2g}\left[\frac{15}{16}\right] = 6.0$$

$$\therefore \qquad V_2 = \sqrt{6.0 \times 2 \times 9.81 \times \frac{16}{15}} = 11.205 \text{ m/sec}$$

$\therefore$ Theoretical discharge $= V_2 \times A_2 = 11.205 \times .001963 = 0.022 \text{ m}^3/\text{s}$

$$\therefore \qquad C_d = \frac{\text{Actual discharge}}{\text{Theoretical discharge}} = \frac{0.02}{0.022} = \mathbf{0.909.\ Ans.}$$

Problem 7.8 *A tank has two identical orifices on one of its vertical sides. The upper orifice is 3 m below the water surface and lower one is 5 m below the water surface. If the value of C_v for each orifice is 0.96, find the point of intersection of the two jets.*

Solution. Given :

Height of water from orifice (1), $H_1 = 3$ m

From orifice (2), $H_2 = 5$ m

C_v for both = 0.96

Let P is the point of intersection of the two jets coming from orifices (1) and (2), such that

x = horizontal distance of P

y_1 = vertical distance of P from orifice (1)

y_2 = vertical distance of P from orifice (2)

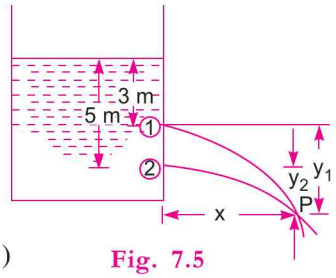


Fig. 7.5

Then $y_1 = y_2 + (5 - 3) = y_2 + 2$ m

The value of C_v is given by equation (7.6) as

For orifice (1), $$C_{v_1} = \frac{x}{\sqrt{4y_1 H_1}} = \frac{x}{\sqrt{4y_1 \times 3.0}} \quad ...(i)$$

For orifice (2), $$C_{v_2} = \frac{x}{\sqrt{4y_2 H_2}} = \frac{x}{\sqrt{4 \times y_2 \times 5.0}} \quad ...(ii)$$

As both the orifices are identical

$\therefore$ $$C_{v_1} = C_{v_2}$$

or $$\frac{x}{\sqrt{4y_1 \times 3.0}} = \frac{x}{\sqrt{4y_2 \times 5.0}} \text{ or } 3y_1 = 5y_2$$

But $$y_1 = y_2 + 2.0$$

$\therefore$ $$3(y_2 + 2.0) = 5y_2$$

$\therefore$ $$2y_2 = 6.0$$

$\therefore$ $$y_2 = 3.0$$

From (ii), $$C_{v_2} = \frac{x}{\sqrt{4y_2 \times H_2}}$$

or $$0.96 = \frac{x}{\sqrt{4 \times 3.0 \times 5.0}}$$

$\therefore$ $$x = 0.96 \times \sqrt{4 \times 3.0 \times 5.0} = \mathbf{7.436\ m.\ Ans.}$$

Problem 7.9 *A closed vessel contains water upto a height of 1.5 m and over the water surface there is air having pressure 7.848 N/cm^2 (0.8 kgf/cm^2) above atmospheric pressure. At the bottom of the vessel there is an orifice of diameter 100 mm. Find the rate of flow of water from orifice. Take $C_d = 0.6$.*

Solution. Given :

Dia. of orifice, $d = 100$ mm $= 0.1$ m

$C_d = 0.6$

Height of water, $H = 1.5$ m

Air pressure, $p = 7.848$ N/cm^2 $= 7.848 \times 10^4$ N/m^2

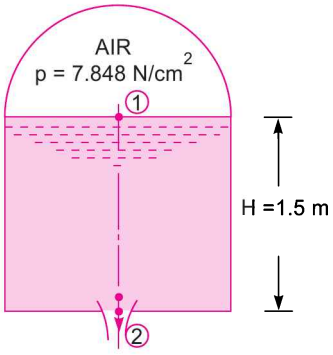


Fig. 7.6

Applying Bernoulli's equation at sections (1) (water surface) and (2), we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2$$

Taking datum line passing through (2) which is very close to the bottom surface of the tank. Then $z_2 = 0$, $z_1 = 1.5$ m

Also $$\frac{p_2}{\rho g} = 0 \text{ (atmospheric pressure)}$$

and $$\frac{p_1}{\rho g} = \frac{7.848 \times 10^4}{1000 \times 9.81} = 8 \text{ m of water}$$

$\therefore$ $$8 + 0 + 1.5 = 0 + \frac{V_2^2}{2g} + 0 \quad \{V_1 \text{ is negligible}\}$$

$\therefore$ $$9.5 = \frac{V_2^2}{2g}$$

$\therefore \quad V_2 = \sqrt{2 \times 9.81 \times 9.5} = 13.652$ m/s

$\therefore$ Rate of flow of water $= C_d \times a_2 \times V_2$

$= 0.6 \times \frac{\pi}{4} (.1)^2 \times 13.652$ m^3/s = **0.0643 m^3/s. Ans.**

Problem 7.10 *A closed tank partially filled with water upto a height of 0.9 m having an orifice of diameter 15 mm at the bottom of the tank. The air is pumped into the upper part of the tank. Determine the pressure required for a discharge of 1.5 litres/s through the orifice. Take $C_d = 0.62$.*

Solution. Given :

Height of water above orifice, $H = 0.9$ m

Dia. of orifice, $d = 15$ mm $= 0.015$ m

$\therefore$ Area, $a = \frac{\pi}{4}[d^2] = \frac{\pi}{4} (.015)^2 = 0.0001767$ m^2

Discharge, $Q = 1.5$ litres/s $= .0015$ m^3/s

$C_d = 0.62$

Let p is intensity of pressure required above water surface in N/cm^2.

Then pressure head of air $= \frac{p}{\rho g} = \frac{p \times 10^4}{1000 \times 9.81} = \frac{10p}{9.81}$ m of water.

If V_2 is the velocity at outlet of orifice, then

$$V_2 = \sqrt{2g\left(H + \frac{p}{\rho g}\right)} = \sqrt{2 \times 9.81\left(0.9 + \frac{10p}{9.81}\right)}$$

$\therefore$ Discharge $Q = C_d \times a \times \sqrt{2g\,(H + p/\rho g)}$

$$.0015 = 0.6 \times .0001767 \times \sqrt{2 \times 9.81\,(0.9 + p/\rho g)}$$

$\therefore \quad \sqrt{2 \times 9.81\left(0.9 + \frac{10p}{9.81}\right)} = \frac{.0015}{0.6 \times .0001767} = 14.148$

or $\quad 2 \times 9.81 \left(0.9 + \frac{10p}{9.81}\right) = 14.148 \times 14.148$

$\therefore \quad \frac{10p}{9.81} = \frac{14.148 \times 14.148}{2 \times 9.81} - 0.9 = 10.202 - 0.9 = 9.302$

$\therefore \quad p = \frac{9.302 \times 9.81}{10} =$ **9.125 N/cm^2. Ans.**

▶ 7.6 FLOW THROUGH LARGE ORIFICES

If the head of liquid is less than 5 times the depth of the orifice, the orifice is called large orifice. In case of small orifice, the velocity in the entire cross-section of the jet is considered to be constant and discharge can be calculated by $Q = C_d \times a \times \sqrt{2gh}$. But in case of a large orifice, the velocity is not constant over the entire cross-section of the jet and hence Q cannot be calculated by $Q = C_d \times a \times \sqrt{2gh}$.

7.6.1 Discharge Through Large Rectangular Orifice. Consider a large rectangular orifice in one side of the tank discharging freely into atmosphere under a constant head, H as shown in Fig. 7.7.

Let H_1 = height of liquid above top edge of orifice

H_2 = height of liquid above bottom edge of orifice

b = breadth of orifice

d = depth of orifice = $H_2 - H_1$

C_d = co-efficient of discharge.

Consider an elementary horizontal strip of depth 'dh' at a depth of 'h' below the free surface of the liquid in the tank as shown in Fig. 7.7 (b).

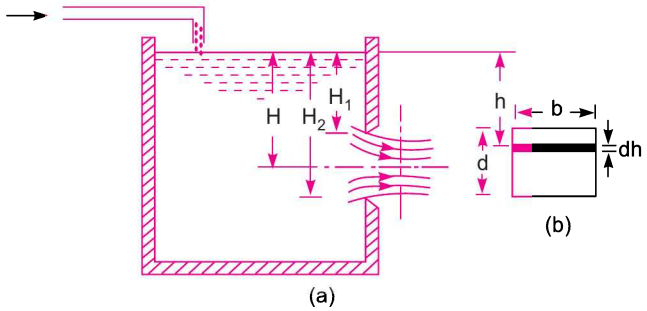


Fig. 7.7 *Large rectangular orifice.*

$\therefore$ Area of strip = $b \times dh$

and theoretical velocity of water through strip = $\sqrt{2gh}$.

$\therefore$ Discharge through elementary strip is given

$$dQ = C_d \times \text{Area of strip} \times \text{Velocity}$$

$$= C_d \times b \times dh \times \sqrt{2gh} = C_d\, b \times \sqrt{2gh}\ dh$$

By integrating the above equation between the limits H_1 and H_2, the total discharge through the whole orifice is obtained

$\therefore$ $$Q = \int_{H_1}^{H_2} C_d \times b \times \sqrt{2gh}\ dh$$

$$= C_d \times b \times \sqrt{2g} \int_{H_1}^{H_2} \sqrt{h}\ dh = C_d \times b \times \sqrt{2g} \left[\frac{h^{3/2}}{3/2}\right]_{H_1}^{H_2}$$

$$= \frac{2}{3} C_d \times b \sqrt{2g} \left[H_2^{3/2} - H_1^{3/2}\right]. \qquad ...(7.8)$$

Problem 7.11 *Find the discharge through a rectangular orifice 2.0 m wide and 1.5 m deep fitted to a water tank. The water level in the tank is 3.0 m above the top edge of the orifice. Take C_d = 0.62.*

Solution. Given :

Width of orifice, b = 2.0 m

Depth of orifice, d = 1.5 m

Height of water above top edge of the orifice, H_1 = 3 m

Height of water above bottom edge of the orifice,

$$H_2 = H_1 + d = 3 + 1.5 = 4.5 \text{ m}$$

$$C_d = 0.62$$

Discharge Q is given by equation (7.8) as

$$Q = \frac{2}{3} C_d \times b \times \sqrt{2g}\ [H_2^{3/2} - H_1^{3/2}]$$

$$= \frac{2}{3} \times 0.62 \times 2.0 \times \sqrt{2 + 9.81}\,[4.5^{1.5} - 3^{1.5}]\ \text{m}^3/\text{s}$$

$$= 3.66[9.545 - 5.196]\ \text{m}^3/\text{s} = \mathbf{15.917\ m^3/s.\ Ans.}$$

Problem 7.12 *A rectangular orifice, 1.5 m wide and 1.0 m deep is discharging water from a tank. If the water level in the tank is 3.0 m above the top edge of the orifice, find the discharge through the orifice. Take the co-efficient of discharging for the orifice = 0.6.*

Solution. Given :

Width of orifice, $b = 1.5$ m

Depth of orifice, $d = 1.0$ m

$H_1 = 3.0$ m

$H_2 = H_1 + d = 3.0 + 1.0 = 4.0$ m

$C_d = 0.6$

Discharge, Q is given by the equation (7.8) as

$$Q = \frac{2}{3} \times C_d \times b \times \sqrt{2g}\ [H_2^{3/2} - H_1^{3/2}]$$

$$= \frac{2}{3} \times 0.6 \times 1.5 \times \sqrt{2 + 9.81}\,[4.0^{1.5} - 3.0^{1.5}]\ \text{m}^3/\text{s}$$

$$= 2.657\ [8.0 - 5.196]\ \text{m}^3/\text{s} = \mathbf{7.45\ m^3/s.\ Ans.}$$

Problem 7.13 *A rectangular orifice 0.9 m wide and 1.2 m deep is discharging water from a vessel. The top edge of the orifice is 0.6 m below the water surface in the vessel. Calculate the discharge through the orifice if $C_d = 0.6$ and percentage error if the orifice is treated as a small orifice.*

Solution. Given :

Width of orifice, $b = 0.9$ m

Depth of orifice, $d = 1.2$ m

$H_2 = 0.6$ m

$H_2 = H_1 + d = 0.6 + 1.2 = 1.8$ m

$C_d = 0.6$

Discharge Q is given as $Q = \frac{2}{3} \times C_d \times b \times \sqrt{2g} \times [H_2^{3/2} - H_1^{3/2}]$

$$= \frac{2}{3} \times 0.6 \times 2.9 \times \sqrt{2 \times 9.81}\ [1.8^{3/2} - 0.6^{3/2}]\ \text{m}^3/\text{s}$$

$$= 1.5946\ [2.4149 - .4647] = \mathbf{3.1097\ m^3/s.\ Ans.}$$

Discharging for a small orifice

$$Q_1 = C_d \times a \times \sqrt{2gh}$$

where $h = H_1 + \frac{d}{2} = 0.6 + \frac{1.2}{2} = 1.2$ m and $a = b \times d = 0.9 \times 1.2$

$$Q_1 = 0.6 \times .9 \times 1.2 \times \sqrt{2 \times 9.81 \times 1.2} = 3.1442 \text{ m}^3/\text{s}$$

$$\% \text{ error} = \frac{Q_1 - Q}{Q} = \frac{3.1442 - 3.1097}{3.1097} = \mathbf{0.01109} \text{ or } \mathbf{1.109\%}. \textbf{ Ans.}$$

▶ 7.7 DISCHARGE THROUGH FULLY SUB-MERGED ORIFICE

Fully sub-merged orifice is one which has its whole of the outlet side sub-merged under liquid so that it discharges a jet of liquid into the liquid of the same kind. It is also called totally drowned orifice. Fig. 7.8 shows the fully sub-merged orifice. Consider two points (1) and (2), point 1 being in the reservoir on the upstream side of the orifice and point 2 being at the vena-contracta as shown in Fig. 7.8.

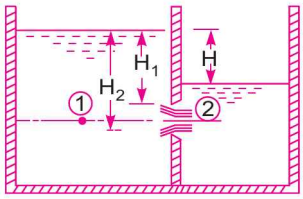


Fig. 7.8 *Fully sub-merged orifice.*

Let H_1 = Height of water above the top of the orifice on the upstream side,

H_2 = Height of water above the bottom of the orifice,

H = Difference in water level,

b = Width of orifice,

C_d = Co-efficient of discharge.

Height of water above the centre of orifice on upstream side

$$= H_1 + \frac{H_2 - H_1}{2} = \frac{H_1 + H_2}{2} \qquad ...(1)$$

Height of water above the centre of orifice on downstream side

$$= \frac{H_1 + H_2}{2} - H \qquad ...(2)$$

Applying Bernoulli's equation at (1) and (2), we get

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} \qquad [\because z_1 = z_2]$$

Now $\frac{p_1}{\rho g} = \frac{H_1 + H_2}{2}$, $\frac{p_2}{\rho g} = \frac{H_1 + H_2}{2} - H$ and V_1 is negligible

$$\therefore \quad \frac{H_1 + H_2}{2} + 0 = \frac{H_1 + H_2}{2} - H + \frac{V_2^2}{2g}$$

$$\therefore \quad \frac{V_2^2}{2g} = H$$

$$\therefore \quad V_2 = \sqrt{2gH}$$

Area of orifice $= b \times (H_2 - H_1)$

$\therefore$ Discharge through orifice $= C_d \times$ Area $\times$ Velocity

$$= C_d \times b\,(H_2 - H_1) \times \sqrt{2gH}$$

$$\therefore \quad Q = C_d \times b\,(H_2 - H_1) \times \sqrt{2gH}\,. \qquad ...(7.9)$$

Problem 7.14 *Find the discharge through a fully sub-merged orifice of width 2 m if the difference of water levels on both sides of the orifice be 50 cm. The height of water from top and bottom of the orifice are 2.5 m and 2.75 m respectively. Take C_d = 0.6.*

Solution. Given :

Width of orifice, $b = 2$ m

Difference of water level, $H = 50$ cm $= 0.5$ m

Height of water from top of orifice, $H_1 = 2.5$ m

Height of water from bottom of orifice, $H_2 = 2.5$ m

$C_d = 0.6$

Discharge through fully sub-merged orifice is given by equation (7.9)

or
$$Q = C_d \times b \times (H_2 - H_1) \times \sqrt{2gH}$$
$$= 0.6 \times 2.0 \times (2.75 - 2.5) \times \sqrt{2 \times 9.81 \times 0.5} \text{ m}^3/\text{s}$$
$$= \mathbf{0.9396\ m^3/s.\ Ans.}$$

Problem 7.15 *Find the discharge through a totally drowned orifice 2.0 m wide and 1 m deep, if the difference of water levels on both the sides of the orifice be 3 m. Take C_d = 0.62.*

Solution. Given :

Width of orifice, $b = 2.0$ m

Depth of orifice, $d = 1$ m.

Difference of water level on both the sides

$H = 3$ m

$C_d = 0.62$

Discharge through orifice is $Q = C_d \times \text{Area} \times \sqrt{2gH}$

$$= 0.62 \times b \times d \times \sqrt{2gH}$$
$$= 0.62 \times 2.0 \times 1.0 \times \sqrt{2 \times 9.81 \times 3} \text{ m}^3/\text{s} = \mathbf{9.513\ m^3/s.\ Ans.}$$

▶ 7.8 DISCHARGE THROUGH PARTIALLY SUB-MERGED ORIFICE

Partially sub-merged orifice is one which has its outlet side partially sub-merged under liquid as shown in Fig. 7.9. It is also known as partially drowned orifice. Thus the partially sub-merged orifice has two portions. The upper portion behaves as an orifice discharging free while the lower portion behaves as a sub-merged orifice. Only a large orifice can behave as a partially sub-merged orifice. The total discharge Q through partially sub-merged orifice is equal to the discharges through free and the sub-merged portions.

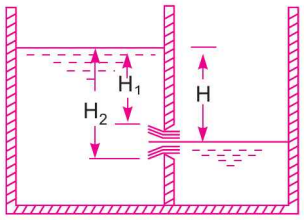


Fig. 7.9 *Partially sub-merged orifice.*

Discharge through the sub-merged portion is given by equation (7.9)

$$Q_1 = C_d \times b \times (H_2 - H) \times \sqrt{2gH}$$

Discharge through the free portion is given by equation (7.8) as

$$Q_2 = \frac{2}{3} C_d \times b \times \sqrt{2g}\ [H_2^{3/2} - H_1^{3/2}]$$

$\therefore$ Total discharge $Q = Q_1 + Q_2$

$$= C_d \times b \times (H_2 - H) \times \sqrt{2gH} + \frac{2}{3} C_d \times b \times \sqrt{2g}\ [H_2^{3/2} - H^{3/2}]. \quad ...(7.10)$$

Problem 7.16 *A rectangular orifice of 2 m width and 1.2 m deep is fitted in one side of a large tank. The water level on one side of the orifice is 3 m above the top edge of the orifice, while on the other side of the orifice, the water level is 0.5 m below its top edge. Calculate the discharge through the orifice if $C_d = 0.64$.*

Solution. Given : Width of orifice, $b = 2$ m

Depth of orifice, $d = 1.2$ m

Height of water from top edge of orifice, $H_1 = 3$ m

Difference of water level on both sides, $H = 3 + 0.5 = 3.5$ m

Height of water from the bottom edge of orifice, $H_2 = H_1 + d = 3 + 1.2 = 4.2$ m

The orifice is partially sub-merged. The discharge through sub-merged portion,

$$Q_1 = C_d \times b \times (H_2 - H) \times \sqrt{2gH}$$

$$= 0.64 \times 2.0 \times (4.2 - 3.5) \times \sqrt{2 \times 9.81 \times 3.5} = 7.4249 \text{ m}^3/\text{s}$$

The discharge through free portion is

$$Q_2 = \frac{2}{3} C_d \times b \times \sqrt{2g}\ [H^{3/2} - H_1^{3/2}]$$

$$= \frac{2}{3} \times 0.64 \times 2.0 \times \sqrt{2 \times 9.81}\ [3.5^{3/2} - 3.0^{3/2}]$$

$$= 3.779\ [6.5479 - 5.1961] = 5.108 \text{ m}^3/\text{s}$$

$\therefore$ Total discharge through the orifice is

$$Q = Q_1 + Q_2 = 7.4249 + 5.108 = \mathbf{12.5329\ m^3/s.\ Ans.}$$

▶ 7.9 TIME OF EMPTYING A TANK THROUGH AN ORIFICE AT ITS BOTTOM

Consider a tank containing some liquid upto a height of H_1. Let an orifice is fitted at the bottom of the tank. It is required to find the time for the liquid surface to fall from the height H_1 to a height H_2.

Let A = Area of the tank

a = Area of the orifice

H_1 = Initial height of the liquid

H_2 = Final height of the liquid

T = Time in seconds for the liquid to fall from H_1 to H_2.

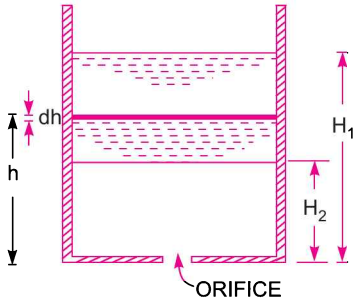


Fig. 7.9. (*a*)

Let at any time, the height of liquid from orifice is h and let the liquid surface fall by a small height dh in time dT. Then

Volume of liquid leaving the tank in time, $dT = A \times dh$

Also the theoretical velocity through orifice, $V = \sqrt{2gh}$

$\therefore$ Discharge through orifice/sec,

$$dQ = C_d \times \text{Area of orifice} \times \text{Theoretical velocity} = C_d \,.\, a \,.\, \sqrt{2gh}$$

$\therefore$ Discharge through orifice in time interval

$$dT = C_d \,.\, a \,.\, \sqrt{2gh} \;.\, dT$$

As the volume of liquid leaving the tank is equal to the volume of liquid flowing through orifice in time dT, we have

$$A(-\,dh) = C_d \,.\, a \,.\, \sqrt{2gh} \;.\, dT$$

– ve sign is inserted because with the increase of time, head on orifice decreases.

$\therefore$ $$-Adh = C_d \,.\, a \,.\, \sqrt{2gh} \;.\, dT \text{ or } dT = \frac{-A\,dh}{C_d \,.\, a \,.\, \sqrt{2gh}} = \frac{-A(h)^{-1/2}}{C_d \,.\, a \,.\, \sqrt{2g}} dh$$

By integrating the above equation between the limits H_1 and H_2, the total time, T is obtained as

$$\int_0^T dT = \int_{H_1}^{H_2} \frac{-Ah^{-1/2}\,dh}{C_d \,.\, a \,.\, \sqrt{2g}} = \frac{-A}{C_d \,.\, a \,.\, \sqrt{2g}} \int_{H_1}^{H_2} h^{-1/2}\,dh$$

or $$T = \frac{-A}{C_d \,.a.\, \sqrt{2g}} \left[\frac{h^{-\frac{1}{2}+1}}{-\frac{1}{2}+1} \right]_{H_1}^{H_2} = \frac{-A}{C_d \,.a.\, \sqrt{2g}} \left[\frac{\sqrt{h}}{\frac{1}{2}} \right]_{H_1}^{H_2}$$

$$= \frac{-2A}{C_d \,.\, a \,.\, \sqrt{2g}} \left[\sqrt{H_2} - \sqrt{H_1} \right] = \frac{2A\left[\sqrt{H_1} - \sqrt{H_2} \right]}{C_d \,.\, a \,.\, \sqrt{2g}} \qquad \ldots(7.11)$$

For emptying the tank completely, $H_2 = 0$ and hence

$$T = \frac{2A\sqrt{H_1}}{C_d \,.\, a \,.\, \sqrt{2g}}. \qquad \ldots(7.12)$$

Problem 7.17 *A circular tank of diameter 4 m contains water upto a height of 5 m. The tank is provided with an orifice of diameter 0.5 m at the bottom. Find the time taken by water (i) to fall from 5 m to 2 m (ii) for completely emptying the tank. Take $C_d = 0.6$.*

Solution. Given :

Dia. of tank, $D = 4$ m

$\therefore$ Area, $A = \frac{\pi}{4}(4)^2 = 12.566 \text{ m}^2$

Dia. of orifice, $d = 0.5$ m

$\therefore$ Area, $a = \frac{\pi}{4}(.5)^2 = 0.1963 \text{ m}^2$

Initial height of water, $H_1 = 5$ m

Final height of water, (*i*) $H_2 = 2$ m (*ii*) $H_2 = 0$

First Case. When $H_2 = 2$ m

Using equation (7.11), we have $T = \frac{2A}{C_d \,.\, a \,.\, \sqrt{2g}} \left[\sqrt{H_1} - \sqrt{H_2} \right]$

$$= \frac{2 \times 12.566}{0.6 \times .1963 \times \sqrt{2 \times 9.81}} \left[\sqrt{5} - \sqrt{2.0}\right] \text{ seconds}$$

$$= \frac{20.653}{0.5217} = \textbf{39.58 seconds. Ans.}$$

Second Case. When $H_2 = 0$

$$T = \frac{2A}{C_d \,.\, a \,.\, \sqrt{2g}} \sqrt{H_1} = \frac{2 \times 12.566 \times \sqrt{5}}{0.6 \times .1963 \times \sqrt{2 \times 9.81}}$$

$$= \textbf{107.7 seconds. Ans.}$$

Problem 7.18 *A circular tank of diameter 1.25 m contains water upto a height of 5 m. An orifice of 50 mm diameter is provided at its bottom. If $C_d = 0.62$, find the height of water above the orifice after 1.5 minutes.*

Solution. Given :

Dia. of tank, $D = 1.25$ m

$\therefore$ Area, $A = \frac{\pi}{4}(1.25)^2 = 1.227 \text{ m}^2$

Dia. of orifice, $d = 50 \text{ mm} = .05 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4}(.05)^2 = .001963 \text{ m}^2$

$C_d = 0.62$

Initial height of water, $H_1 = 5$ m

Time in seconds, $T = 1.5 \times 60 = 90$ seconds

Let the height of water after 90 seconds = H_2

Using equation (7.11), we have $T = \dfrac{2A\left[\sqrt{H_1} - \sqrt{H_2}\right]}{C_d \,.\, a \,.\, \sqrt{2g}}$

or $$90 = \frac{2 \times 1.227\left[\sqrt{5} - \sqrt{H_2}\right]}{0.62 \times 0.001963 \times \sqrt{2 \times 9.81}} = 455.215\left[2.236 - \sqrt{H_2}\right]$$

$\therefore$ $$\sqrt{H_2} = 2.236 - \frac{90}{455.215} = 236 - 0.1977 = 2.0383$$

$\therefore$ $$H_2 = 2.0383 \times 2.0383 = \textbf{4.154 m. Ans.}$$

▶ 7.10 TIME OF EMPTYING A HEMISPHERICAL TANK

Consider a hemispherical tank of radius R fitted with an orifice of area 'a' at its bottom as shown in Fig. 7.10. The tank contains some liquid whose initial height is H_1 and in time T, the height of liquid falls to H_2. It is required to find the time T.

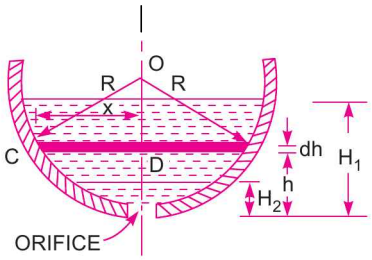


Fig. 7.10 *Hemispherical tank.*

Let at any instant of time, the head of liquid over the orifice is h and at this instant let x be the radius of the liquid surface. Then

Area of liquid surface, $A = \pi x^2$

and theoretical velocity of liquid = $\sqrt{2gh}$.

Let the liquid level falls down by an amount of dh in time dT.

$\therefore$ Volume of liquid leaving tank in time $dT = A \times dh$

$$= \pi x^2 \times dh \qquad ...(i)$$

Also volume of liquid flowing through orifice

$$= C_d \times \text{area of orifice} \times \text{velocity} = C_d.a.\sqrt{2gh} \text{ second}$$

$\therefore$ Volume of liquid flowing through orifice in time dT

$$= C_d.a.\sqrt{2gh} \times dT \qquad ...(ii)$$

From equations (i) and (ii), we get

$$\pi x^2 (-dh) = C_d.a.\sqrt{2gh} \; . \; dT$$

–ve sign is introduced, because with the increase of T, h will decrease

$$\therefore \qquad -\pi x^2 \, dh = C_d.a.\sqrt{2gh} \; . \; dT \qquad ...(iii)$$

But from Fig. 7.10, for ΔOCD, we have $OC = R$

$$DO = R - h$$

$$\therefore \qquad CD = x = \sqrt{OC^2 - OD^2} = \sqrt{R^2 - (R-h)^2}$$

$$\therefore \qquad x^2 = R^2 - (R-h)^2 = R^2 - (R^2 + h^2 - 2Rh) = 2Rh - h^2$$

Substituting x^2 in equation (iii), we get

$$-\pi(2Rh - h^2)dh = C_d.a.\sqrt{2gh} \; . \; dT$$

or
$$dT = \frac{-\pi\left(2Rh - h^2\right)dh}{C_d \, . \, a \, . \sqrt{2gh}} = \frac{-\pi}{C_d \, . \, a \, . \sqrt{2g}} \left(2Rh - h^2\right) h^{-1/2} \; dh$$

$$= \frac{-\pi}{C_d \, . \, a \, . \sqrt{2g}} (2Rh^{1/2} - h^{3/2})dh$$

The total time T required to bring the liquid level from H_1 to H_2 is obtained by integrating the above equation between the limits H_1 and H_2.

$\therefore$
$$T = \int_{H_1}^{H_2} \frac{-\pi}{C_d \, . \, a \, . \sqrt{2g}} (2Rh^{1/2} - h^{3/2})dh$$

$$= \frac{-\pi}{C_d \, . \, a \, . \sqrt{2g}} \int_{H_1}^{H_2} (2Rh^{1/2} - h^{3/2})dh$$

$$= \frac{-\pi}{C_d \times a \times \sqrt{2g}} \left[2R \frac{h^{1/2+1}}{\frac{1}{2}+1} - \frac{h^{3/2}+1}{\frac{3}{2}+1} \right]_{H_1}^{H_2}$$

$$= \frac{-\pi}{C_d \times a \times \sqrt{2g}} \left[2 \times \frac{2}{3} R h^{3/2} - \frac{2}{5} h^{5/2} \right]_{H_1}^{H_2}$$

$$= \frac{-\pi}{C_d \times a \times \sqrt{2g}} \left[\frac{4}{3} R \left(H_2^{3/2} - H_1^{3/2} \right) - \frac{2}{5} \left(H_2^{5/2} - H_1^{5/2} \right) \right]$$

$$= \frac{\pi}{C_d \times a \times \sqrt{2g}} \left[\frac{4}{3} R \left(H_1^{3/2} - H_2^{3/2} \right) - \frac{2}{5} \left(H_1^{5/2} - H_2^{5/2} \right) \right] \quad ...(7.13)$$

For completely emptying the tank, $H_2 = 0$ and hence

$$T = \frac{\pi}{C_d \,.\, a \,.\, \sqrt{2g}} \left[\frac{4}{3} R H_1^{3/2} - \frac{2}{5} H_1^{5/2} \right]. \quad ...(7.14)$$

Problem 7.19 *A hemispherical tank of diameter 4 m contains water upto a height of 1.5 m. An orifice of diameter 50 mm is provided at the bottom. Find the time required by water (i) to fall from 1.5 m to 1.0 m (ii) for completely emptying the tank. Tank C_d = 0.6.*

Solution. Given :

Dia. of hemispherical tank, $D = 4$ m

$\therefore$ Radius, $R = 2.0$ m

Dia. of orifice, $d = 50$ mm $= 0.05$ m

$\therefore$ Area, $a = \frac{\pi}{4}(.05)^2 = 0.001963$ m^2

Initial height of water, $H_1 = 1.5$ m

$C_d = 0.6$

First Case. $H_2 = 1.0$

Time T is given by equation (7.13)

$$\therefore \quad T = \frac{\pi}{C_d \times a \times \sqrt{2g}} \left[\frac{4}{3} R \left(H_1^{3/2} - H_2^{3/2} \right) - \frac{2}{5} \left(H_1^{5/2} - H_2^{5/2} \right) \right]$$

$$= \frac{\pi}{0.6 \times .001963 \times \sqrt{2 \times 9.81}} \times \left[\frac{4}{3} \times 2.0 \left(1.5^{3/2} - 1.0^{3/2} \right) - \frac{2}{5} \left(1.5^{5/2} - 1.0^{5/2} \right) \right]$$

$= 602.189$ [2.2323 – 0.7022] = 921.4 second

$=$ **15 min 21.4 sec. Ans.**

Second Case. $H_2 = 0$ and hence time T is given by equation (7.14)

$$\therefore \quad T = \frac{\pi}{C_d \,.\, a \,.\, \sqrt{2g}} \left[\frac{4}{3} R H_1^{3/2} - \frac{2}{5} H_1^{5/2} \right]$$

$$= \frac{\pi}{0.6 \times .001963 \sqrt{2 \times 9.81}} \left[\frac{4}{3} \times 2.0 \times 1.5^{3/2} - \frac{2}{5} \times 1.5^{5/2} \right]$$

$= 602.189\ [4.8989 - 1.1022]$ sec $= 2286.33$ sec

$=$ **38 min 6.33 sec. Ans.**

Problem 7.20 *A hemispherical cistern of 6 m radius is full of water. It is fitted with a 75 mm diameter sharp edged orifice at the bottom. Calculate the time required to lower the level in the cistern by 2 metres. Assume co-efficient of discharge for the orifice is 0.6.*

Solution. Given :

Radius of hemispherical cistern, $R = 6$ m

Initial height of water, $H_1 = 6$ m

Dia. of orifice, $d = 75$ mm $= 0.075$ m

$\therefore$ Area, $a = \frac{\pi}{4}(.075)^2 = .004418\ \text{m}^2$

Fall of height of water $= 2$ m

$\therefore$ Final height of water, $H_2 = 6 - 2 = 4$ m

$C_d = 0.6$

The time T is given by equation (7.31)

$$T = \frac{\pi}{C_d \times a \times \sqrt{2g}}\left[\frac{4}{3}R\left(H_1^{3/2} - H_2^{3/2}\right) - \frac{2}{5}\left(H_1^{5/2} - H_2^{5/2}\right)\right]$$

$$= \frac{\pi}{0.6 \times .004418 \times \sqrt{2 \times 9.81}} \times \left[\frac{4}{3} \times 6\left(6.0^{3/2} - 4.0^{3/2}\right) - \frac{2}{5}\left(6.0^{5/2} - 4.0^{5/2}\right)\right]$$

$= 267.56\ [8(14.6969 - 8.0) - 0.4\ (88.18 - 32.0)]$

$= 267.56\ [53.575 - 22.472]$ sec

$= 8321.9$ sec $=$ **2hrs 18 min 42 sec. Ans.**

Problem 7.21 *A cylindrical tank is having a hemispherical base. The height of cylindrical portion is 5 m and diameter is 4 m. At the bottom of this tank an orifice of diameter 200 mm is fitted. Find the time required to completely emptying the tank. Take $C_d = 0.6$.*

Solution. Given :

Height of cylindrical portion (II) $= 5$ m

Dia. of tank $= 4.0$ m

$\therefore$ Area, $A = \frac{\pi}{4}(4)^2 = 12.566\ \text{m}^2$

Dia. of orifice, $d = 200$ mm $= 0.2$ m

$\therefore$ Area, $a = \frac{\pi}{4}(.2)^2 = 0.0314\ \text{m}^2$

$C_d = 0.6$

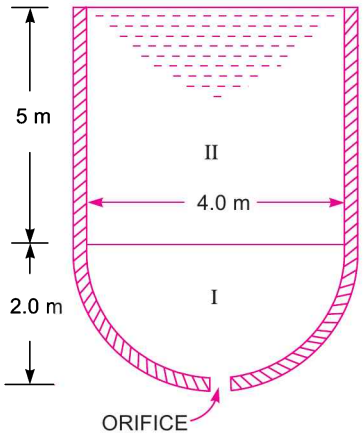


Fig. 7.11

The tank is splitted in two portions. First portion is a hemispherical tank and second portion is cylindrical tank.

Let T_1 = time for emptying hemispherical portion I.

T_2 = time for emptying cylindrical portion II.

Then total time $T = T_1 + T_2$.

For Portion I. $H_1 = 2.0$ m, $H_2 = 0$. Then T_1 is given by equation (7.14) as

$$T_1 = \frac{\pi}{C_d \times a \times \sqrt{2g}} \left[\frac{4}{3} R H_1^{3/2} - \frac{2}{5} H_1^{5/2} \right]$$

$$= \frac{\pi}{0.6 \times .0314 \times \sqrt{2 \times 9.81}} \left[\frac{4}{3} \times 2.0 \times 2.0^{3/2} - \frac{2}{5} \times 2.0^{5/2} \right]$$

$$= 37.646\ [7.5424 - 2.262]\ \text{sec} = 198.78\ \text{sec}.$$

For Portion II. $H_1 = 2.0 + 5.0 = 7.0$ m, $H_2 = 2.0$. Then T_2 is given by equation (7.11) as

$$T_2 = \frac{2A\left[\sqrt{H_1} - \sqrt{H_2}\right]}{C_d \times a \times \sqrt{2g}} = \frac{2 \times 12.566 \left[\sqrt{7} - \sqrt{2.0}\right]}{0.6 \times .0314 \times \sqrt{2 \times 9.81}} \text{sec} = 370.92\ \text{sec}$$

$\therefore$ Total time, $T = T_1 + T_2 = 198.78 + 370.92 = 569.7$ sec

= 9 min 29 sec. Ans.

▶ 7.11 TIME OF EMPTYING A CIRCULAR HORIZONTAL TANK

Consider a circular horizontal tank of length L and radius R, containing liquid upto a height of H_1. Let an orifice of area 'a' is fitted at the bottom of the tank. Then the time required to bring the liquid level from H_1 to H_2 is obtained as :

Let at any time, the height of liquid over orifice is 'h' and in time dT, let the height falls by an height of 'dh'. Let at this time, the width of liquid surface = AC as shown in Fig. 7.12.

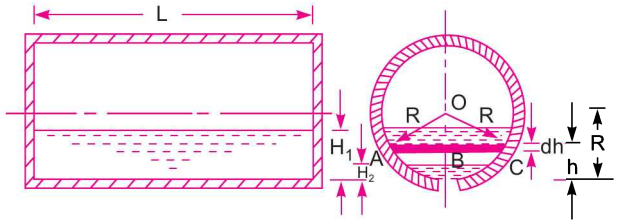


Fig. 7.12

$\therefore$ Surface area of liquid = $L \times AC$

But $$AC = 2 \times AB = 2\left[\sqrt{AO^2 - OB^2}\right] = 2\left[\sqrt{R^2 - (R-h)^2}\right]$$

$$= 2\sqrt{R^2 - \left(R^2 + h^2 - 2Rh\right)} = 2\sqrt{2Rh - h^2}$$

$\therefore$ Surface area, $A = L \times 2\sqrt{2Rh - h^2}$

$\therefore$ Volume of liquid leaving tank in time dT

$$= A \times dh = 2L\sqrt{2Rh - h^2} \times dh \qquad ...(i)$$

Also the volume of liquid flowing through orifice in time dT

$$= C_d \times \text{Area of orifice} \times \text{Velocity} \times dT$$

But the velocity of liquid at the time considered = $\sqrt{2gh}$

$\therefore$ Volume of liquid flowing through orifice in time dT

$$= C_d \times a \times \sqrt{2gh} \times dT \qquad ...(ii)$$

Equating (i) and (ii), we get

$$2L\sqrt{2Rh - h^2} \times (-dh) = C_d \times a \times \sqrt{2gh \times dT}$$

– ve sign is introduced as with the increase of T, the height h decreases,

$\therefore$ $$dT = \frac{-2L\sqrt{2Rh - h^2}\, dh}{C_d \times a \times \sqrt{2gh}} = \frac{-2L\sqrt{(2R - h)}\, dh}{C_d \times a \times \sqrt{2g}}$$

[Taking $\sqrt{h}$ common]

$\therefore$ Total time, $$T = \int_{H_1}^{H_2} \frac{-2L(2R - h)^{1/2}\, dh}{C_d \times a \times \sqrt{2g}}$$

$$= \frac{-2L}{C_d \times a \times \sqrt{2g}} \int_{H_1}^{H_2} [2R - h]^{1/2}\, dh$$

$$= \frac{-2L}{C_d \times a \times \sqrt{2g}} \left[\frac{(2R - h)^{1/2+1}}{\frac{1}{2} + 1} \times (-1) \right]_{H_1}^{H_2}$$

$$= \frac{2L}{C_d \times a \times \sqrt{2g}} \times \frac{2}{3} \times \left[(2R - h)^{3/2} \right]_{H_1}^{H_2}$$

$$= \frac{4L}{3C_d \times a \times \sqrt{2g}} \left[(2R - H_2)^{3/2} - (2R - H_1)^{3/2} \right] \qquad ...(7.15)$$

For completely emptying the tank, $H_2 = 0$ and hence

$$T = \frac{4L}{3C_d \times a \times \sqrt{2g}} \left[(2R)^{3/2} - (2R - H_1)^{3/2} \right]. \qquad ...(7.16)$$

Problem 7.22 *An orifice of diameter 100 mm is fitted at the bottom of a boiler drum of length 5 m and of diameter 2 m. The drum is horizontal and half full of water. Find the time required to empty the boiler, given the value of C_d = 0.6.*

Solution. Given :

Dia. of orifice, $d = 100 \text{ mm} = 0.1 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4}(.1)^2 = .007854 \text{ m}^2$

Length, $L = 5 \text{ m}$

Dia. of drum, $D = 2 \text{ m}$

$\therefore$ Radius, $R = 1 \text{ m}$

Initial height of water, $H_1 = 1 \text{ m}$

Final height of water, $H_2 = 0$

$C_d = 0.6$

For completely emptying the tank, T is given by equation (7.16)

$$\therefore \quad T = \frac{4L}{3 \times C_d \times a \times \sqrt{2g}} [(2R)^{3/2} - (2R - H_1)^{3/2}]$$

$$= \frac{4 \times 5.0}{3 \times .06 \times .007854 \times \sqrt{2 \times 9.81}} [(2 \times 1)^{3/2} - (2 \times 1 - 1)^{3/2}]$$

$$= 319.39 \, [2.8284 - 1.0] = 583.98 \text{ sec} = \textbf{9 min 44 sec. Ans.}$$

Problem 7.23 *An orifice of diameter 150 mm is fitted at the bottom of a boiler drum of length 8 m and of diameter 3 metres. The drum is horizontal and contains water upto a height of 2.4 m. Find the time required to empty the boiler. Take $C_d = 0.6$.*

Solution. Given :

Dia. of orifice, $d = 150 \text{ mm} = 0.15 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4}(.15)^2 = 0.01767 \text{ m}^2$

Length, $L = 8.0 \text{ m}$

Dia. of boiler, $D = 3.0 \text{ m}$

$\therefore$ Radius, $R = 1.5 \text{ m}$

Initial height of water, $H_1 = 2.4 \text{ m}$

Find height of water, $H_2 = 0$

$C_d = 0.6$.

For completely emptying the tank, T is given by equation (7.16) as

$$T = \frac{4L}{3C_d \times a \times \sqrt{2g}} [(2\text{R})^{3/2} - (2R - H_1)^{3/2}]$$

$$= \frac{4 \times 8.0}{3 \times .6 \times .01767 \times \sqrt{2 \times 9.81}} [(2 \times 1.5)^{3/2} - (2 \times 1.5 - 2.4)^{3/2}]$$

$$= 227.14 \, [5.196 - 0.4647] = 1074.66 \text{ sec}$$

$$= \textbf{17 min 54.66 sec. Ans.}$$

▶ 7.12 CLASSIFICATION OF MOUTHPIECES

1. The mouthpieces are classified as (*i*) External mouthpiece or (*ii*) Internal mouthpiece depending upon their position with respect to the tank or vessel to which they are fitted.

2. The mouthpiece are classified as (*i*) Cylindrical mouthpiece or (*ii*) Convergent mouthpiece or (*iii*) Convergent-divergent mouthpiece depending upon their shapes.

3. The mouthpieces are classified as (*i*) Mouthpieces running full or (*ii*) Mouthpieces running free, depending upon the nature of discharge at the outlet of the mouthpiece. This classification is only for internal mouthpieces which are known Borda's or Re-entrant mouthpieces. A mouthpiece is said to be running free if the jet of liquid after contraction does not touch the sides of the mouthpiece. But if the jet after contraction expands and fills the whole mouthpiece it is known as running full.

▶ 7.13 FLOW THROUGH AN EXTERNAL CYLINDRICAL MOUTHPIECE

A mouthpiece is a short length of a pipe which is two or three times its diameter in length. If this pipe is fitted externally to the orifice, the mouthpiece is called external cylindrical mouthpiece and the discharge through orifice increases.

Consider a tank having an external cylindrical mouthpiece of cross-sectional area a_1, attached to one of its sides as shown in Fig. 7.13. The jet of liquid entering the mouthpiece contracts to form a vena-contracta at a section *C*-*C*. Beyond this section, the jet again expands and fill the mouthpiece completely.

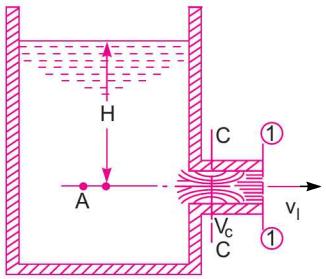


Fig. 7.13 *External cylindrical mouthpieces.*

Let H = Height of liquid above the centre of mouthpiece
v_c = Velocity of liquid at *C*-*C* section
a_c = Area of flow at vena-contracta
v_1 = Velocity of liquid at outlet
a_1 = Area of mouthpiece at outlet
C_c = Co-efficient of contraction.

Applying continuity equation at *C*-*C* and (1)-(1), we get

$$a_c \times v_c = a_1 v_1$$

$$\therefore \quad v_c = \frac{a_1 v_1}{a_c} = \frac{v_1}{a_c / a_1}$$

But $$\frac{a_c}{a_1} = C_c = \text{Co-efficient of contraction}$$

Taking $C_c = 0.62$, we get $\frac{a_c}{a_1} = 0.62$

$$\therefore \quad v_c = \frac{v_1}{0.62}$$

The jet of liquid from section *C*-*C* suddenly enlarges at section (1)-(1). Due to sudden enlargement, there will be a loss of head, h_L* which is given as $h_L = \frac{(v_c - v_1)^2}{2g}$

* Please refer Art. 11.4.1 for loss of head due to sudden enlargement.

But $v_c = \dfrac{v_1}{0.62}$ hence $h_L = \dfrac{\left(\dfrac{v_1}{0.62} - v_1\right)^2}{2g} = \dfrac{v_1^2}{2g}\left[\dfrac{1}{0.62} - 1\right]^2 = \dfrac{0.375\, v_1^2}{2g}$

Applying Bernoulli's equation to point A and (1)-(1)

$$\frac{p_A}{\rho g} + \frac{v_A^2}{2g} + z_A = \frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 + h_L$$

where $z_A = z_1$, v_A is negligible,

$\dfrac{p_1}{\rho g}$ = atmospheric pressure = 0

$$\therefore \quad H + 0 = 0 + \frac{v_1^2}{2g} + .375\,\frac{v_1^2}{2g}$$

$$\therefore \quad H = 1.375\,\frac{v_1^2}{2g}$$

$$\therefore \quad v_1 = \sqrt{\frac{2gH}{1.375}} = 0.855\,\sqrt{2gH}$$

Theoretical velocity of liquid at outlet is $v_{th} = \sqrt{2gH}$

$\therefore$ Co-efficient of velocity for mouthpiece

$$C_v = \frac{\text{Actual velocity}}{\text{Theoretical velocity}} = \frac{0.855\sqrt{2gH}}{\sqrt{2gH}} = 0.855.$$

C_c for mouthpiece = 1 as the area of jet of liquid at outlet is equal to the area of mouthpiece at outlet.

Thus $C_d = C_c \times C_v = 1.0 \times .855 = 0.855$

Thus the value of C_d for mouthpiece is more than the value of C_d for orifice, and so discharge through mouthpiece will be more.

Problem 7.24 *Find the discharge from a 100 mm diameter external mouthpiece, fitted to a side of a large vessel if the head over the mouthpiece is 4 metres.*

Solution. Given :

Dia. of mouthpiece = 100 m = 0.1 m

$\therefore$ Area, $a = \dfrac{\pi}{4}(0.1)^2 = 0.007854\ \text{m}^2$

Head, $H = 4.0$ m

C_d for mouthpiece $= 0.855$

$\therefore$ Discharge $= C_d \times \text{Area} \times \text{Velocity} = 0.855 \times a \times \sqrt{2gH}$

$= .855 \times .007854 \times \sqrt{2 \times 9.81 \times 4.0}$ = **.05948 m^3/s. Ans.**

Problem 7.25 *An external cylindrical mouthpiece of diameter 150 mm is discharging water under a constant head of 6 m. Determine the discharge and absolute pressure head of water at vena-contracta. Take C_d = 0.855 and C_c for vena-contracta = 0.62. Atmospheric pressure head = 10.3 m of water.*

Solution. Given :

Dia. of mouthpiece, $d = 150 \text{ mm} = 0.15 \text{ cm}$

$\therefore$ Area, $a = \frac{\pi}{4}(.15)^2 = 0.01767 \text{ m}^2$

Head, $H = 6.0 \text{ m}$

$C_d = 0.855$

C_c at vena-contracta = 0.62

Atmospheric pressure head, $H_a = 10.3 \text{ m}$

$\therefore$ Discharge $= C_d \times a \times \sqrt{2gH}$

$= 0.855 \times .01767 \times \sqrt{2 \times 9.81 \times 6.0} =$ **0.1639 m^3/s. Ans.**

Pressure Head at Vena-contracta

Applying Bernoulli's equation at A and C-C, we get

$$\frac{p_A}{\rho g} + \frac{v_A^2}{2g} + z_A = \frac{p_c}{\rho g} + \frac{v_c^2}{2g} + z_c$$

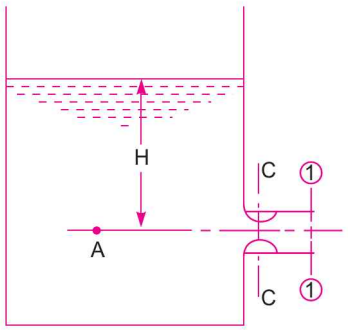


Fig. 7.14

But $\frac{p_A}{\rho g} = H_a + H$, $v_A = 0$,

$z_A = z_c$

$\therefore$ $H_a + H + 0 = \frac{p_c}{\rho g} + \frac{v_c^2}{2g} = H_c + \frac{v_c^2}{2g}$

$\therefore$ $H_c = H_a + H - \frac{v_c^2}{2g}$

But $v_c = \frac{v_1}{0.62}$

$\therefore$ $H_c = H_a + H - \left(\frac{v_1}{.62}\right)^2 \times \frac{1}{2g} = H_a + H - \frac{v_1^2}{2g} \times \frac{1}{(.62)^2}$

But $H = 1.375 \ \frac{v_1^2}{2g}$

$\therefore$ $\frac{v_1^2}{2g} = \frac{H}{1.375} = 0.7272 \ H$

$\therefore$ $H_c = H_a + H - .7272 \ H \times \frac{1}{(.62)^2}$

$= H_a + H - 1.89 \ H = H_a - .89 \ H$

$= 10.3 - .89 \times 6.0$ $\{\because H_a = 10.3 \text{ and } H = 6.0\}$

$= 10.3 - 5.34 =$ **4.96 m (Absolute). Ans.**

▶ 7.14 FLOW THROUGH A CONVERGENT-DIVERGENT MOUTHPIECE

If a mouthpiece converges upto vena-contracta and then diverges as shown in Fig. 7.15 then that type of mouthpiece is called Convergent-Divergent Mouthpiece. As in this mouthpiece there is no sudden enlargement of the jet, the loss of energy due to sudden enlargement is eliminated. The co-efficient of discharge for this mouthpiece is unity. Let H is the head of liquid over the mouthpiece.

Applying Bernoulli's equation to the free surface of water in tank and section C-C, we have

$$\frac{p}{\rho g} + \frac{v^2}{2g} + z = \frac{p_c}{\rho g} + \frac{v_c^2}{2g} + z_c$$

Taking datum passing through the centre of orifice, we get

$$\frac{p}{\rho g} = H_a,\ v = 0,\ z = H,\ \frac{p_c}{\rho g} = H_c,\ z_c = 0$$

$$\therefore \quad H_a + 0 + H = H_c + \frac{v_c^2}{2g} + 0 \qquad ...(i)$$

$$\therefore \quad \frac{v_c^2}{2g} = H_a + H - H_c \qquad ...(ii)$$

or $$v_c = \sqrt{2g(H_a + H - H_c)}$$

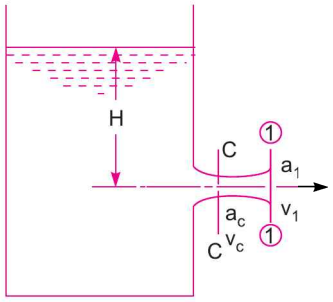


Fig. 7.15 *Convergent-divergent mouthpiece.*

Now applying Bernoulli's equation at sections C-C and (1)-(1)

$$\frac{p_c}{\rho g} + \frac{v_c^2}{2g} + z_c = \frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1$$

But $$z_c = z_1 \text{ and } \frac{p_1}{\rho g} = H_a$$

$$\therefore \quad H_c + \frac{v_c^2}{2g} = H_a + \frac{v_1^2}{2g}$$

Also from (i), $\quad H_c + v_c^2/2g = H + H_a$

$$\therefore \quad H_a + v_1^2/2g = H + H_a$$

$$\therefore \quad v_1 = \sqrt{2gH} \qquad ...(iii)$$

Now by continuity equation, $a_c\, v_c = v_1 \times a_1$

$$\therefore \quad \frac{a_1}{a_c} = \frac{v_c}{v_1} = \frac{\sqrt{2g(H_a + H - H_c)}}{\sqrt{2gH}} = \sqrt{\frac{H_a}{H} + 1 - \frac{H_c}{H}}$$

$$= \sqrt{1 + \frac{H_a - H_c}{H}} \qquad ...(7.17)$$

The discharge, Q is given as $Q = a_c \times \sqrt{2gH}$ ...(7.18)

where a_c = area at vena-contracta.

Problem 7.26 *A convergent-divergent mouthpiece having throat diameter of 4.0 cm is discharging water under a constant head of 2.0 m, determine the maximum outer diameter for maximum discharge. Find maximum discharge also. Take H_a = 10.3 m of water and H_{sep} = 2.5 m of water (absolute).*

Solution. Given :

Dia. of throat, $d_c = 4.0$ cm

$\therefore$ Area, $a_c = \frac{\pi}{4}(4)^2 = 12.566 \text{ cm}^2$

Constant head, $H = 2.0$ m

Find max. dia. at outlet, d_1 and Q_{max}

$H_a = 10.3$ m

$H_{sep} = 2.5$ m (absolute)

The discharge, Q in convergent-divergent mouthpiece depends on the area at throat.

$\therefore$ $Q_{max} = a_c \times \sqrt{2gH} = 12.566 \times \sqrt{2 \times 9.81 \times 200}$ = **7871.5 cm^3/s. Ans.**

Now ratio of areas at outlet and throat is given by equation (7.17) as

$$\frac{a_1}{a_c} = \sqrt{1 + \frac{H_a - H_c}{H}} = \sqrt{1 + \frac{10.3 - 2.5}{2.0}} \qquad \{\because H_c = H_{sep} = 2.5\}$$

$$= 2.2135$$

$$\frac{\pi}{4} d_1^2 \Big/ \frac{\pi}{4} d_c^2 = 2.2135 \text{ or } \left(\frac{d_1}{d_c}\right)^2 = 2.2135$$

$\therefore$ $$\frac{d_1}{d_c} = \sqrt{2.2135} = 1.4877$$

$\therefore$ $d_1 = 1.4877 \times d_c = 1.4877 \times 4.0$ = **5.95 cm. Ans.**

Problem 7.27 *The throat and exit diameters of convergent-divergent mouthpiece are 5 cm and 10 cm respectively. It is fitted to the vertical side of a tank, containing water. Find the maximum head of a water for steady flow. The maximum vacuum pressure is 8 m of water and take atmospheric pressure = 10.3 m water.*

Solution. Given :

Dia. at throat, $d_c = 5$ cm

Dia. at exit, $d_1 = 10$ cm

Atmospheric pressure head, $H_a = 10.3$ m

The maximum vacuum pressure will be at a throat only

$\therefore$ Pressure head at throat = 8 m (vacuum)

or $H_c = H_a - 8.0$ (absolute)

$= 10.3 - 8.0 = 2.3$ m (abs.)

Let maximum head of water over mouthpiece = H m of water.

The ratio of areas at outlet and throat of a convergent-divergent mouthpiece is given by equation (7.17).

$$\frac{a_1}{a_c} = \sqrt{1 + \frac{H_a - H_c}{H}} \quad \text{or} \quad \frac{\frac{\pi}{4}(d_1)^2}{\frac{\pi}{4}(d_c)^2} = \sqrt{1 + \frac{10.3 - 2.3}{H}}$$

or $$\frac{10^2}{5^2} = 4 = \sqrt{1 + \frac{8}{H}} \quad \text{or} \quad 16 = 1 + \frac{8}{H} \quad \text{or} \quad 15 = \frac{8}{H}$$

$\therefore$ $$H = \frac{8}{15} = 0.5333 \text{ m of water}$$

$\therefore$ Maximum head of water = **0.533 m. Ans.**

Problem 7.28 *A convergent-divergent mouthpiece is fitted to the side of a tank. The discharge through mouthpiece under a constant head of 1.5 m is 5 litres/s. The head loss in the divergent portion is 0.10 times the kinetic head at outlet. Find the throat and exit diameters, if separation pressure is 2.5 m and atmospheric pressure head = 10.3 m of water.*

Solution. Given :

Constant head, $H = 1.5$ m

Discharge, $Q = 5$ litres $= .005$ m^3/s

h_L or Head loss in divergent = $0.1 \times$ kinetic head at outlet

H_c or $H_{sep} = 2.5$ (abs.)

$H_a = 10.3$ m of water

Find (*i*) Dia. at throat, d_c

(*ii*) Dia. at outlet, d_1

(*i*) **Dia. at throat (d_c).** Applying Bernoulli's equation to the free water surface and throat section, we get (See Fig. 7.15).

$$\frac{p}{\rho g} + \frac{v^2}{2g} + z = \frac{p_c}{\rho g} + \frac{v_c^2}{2g} + z_c$$

Taking the centre line of mouthpiece as datum, we get

$$H_a + 0 + H = H_c + \frac{v_c^2}{2g}$$

$\therefore$ $$\frac{v_c^2}{2g} = H_a + H - H_c = 10.3 + 1.5 - 2.5 = 9.3 \text{ m of water}$$

$\therefore$ $$v_c = \sqrt{2 \times 9.81 \times 9.3} = 13.508 \text{ m/s}$$

Now $$Q = a_c \times v_c \text{ or } .005 = \frac{\pi}{4} d_c^2 \times 13.508$$

$\therefore$ $$d_c = \sqrt{\frac{.005 \times 4}{\pi \times 13.508}} = \sqrt{.00047} = .0217 \text{ m} = \mathbf{2.17 \text{ cm. Ans.}}$$

(*ii*) **Dia. at outlet (d_1).** Applying Bernoulli's equation to the free water surface and outlet of mouthpiece (See Fig. 7.15), we get

$$\frac{p}{\rho g} + \frac{v^2}{2g} + z = \frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 + h_L$$

$$H_a + 0 + H = H_a + \frac{v_1^2}{2g} + 0 + 0.1 \times \frac{v_1^2}{2g} \qquad \left\{\because \frac{p_1}{\rho g} = H_a\right\}$$

$$\therefore \quad H = \frac{v_1^2}{2g} + .1 \times \frac{v_1^2}{2g} = 1.1\ \frac{v_1^2}{2g}$$

$$\therefore \quad v_1 = \sqrt{\frac{2gH}{1.1}} = \sqrt{\frac{2 \times 9.81 \times 1.5}{1.1}} = 5.1724$$

Now $$Q = A_1 v_1 \text{ or } .005 = \frac{\pi}{4} d_1^2 \times v_1$$

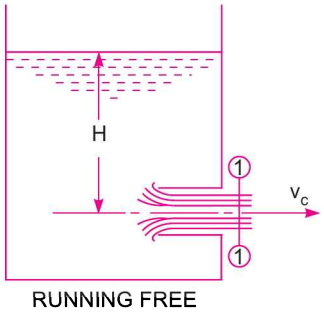
$$\therefore \quad d_1 = \sqrt{\frac{4 \times .005}{\pi \times v_1}} = \sqrt{\frac{4 \times .005}{\pi \times 5.1724}} = 0.035 \text{ m} = \mathbf{3.5\ cm.\ Ans.}$$

▶ 7.15 FLOW THROUGH INTERNAL OR RE-ENTRANT ON BORDA'S MOUTHPIECE

A short cylindrical tube attached to an orifice in such a way that the tube projects inwardly to a tank, is called an internal mouthpiece. It is also called Re-entrant or Borda's mouthpiece. If the length of the tube is equal to its diameter, the jet of liquid comes out from mouthpiece without touching the sides of the tube as shown in Fig. 7.16. The mouthpiece is known as *running free*. But if the length of the tube is about 3 times its diameter, the jet comes out with its diameter equal to the diameter of mouthpiece at outlet as shown in Fig. 7.17. The mouthpiece is said to be *running full.*

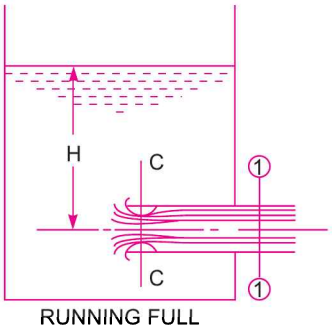
(*i*) **Borda's Mouthpiece Running Free.** Fig. 7.16 shows the Borda's mouthpiece running free.

Let H = height of liquid above the mouthpiece,
a = area of mouthpiece,
a_c = area of contracted jet in the mouthpiece,
v_c = velocity through mouthpiece.



RUNNING FREE

Fig. 7.16



RUNNING FULL

Fig. 7.17

The flow of fluid through mouthpiece is taking place due to the pressure force exerted by the fluid on the entrance section of the mouthpiece. As the area of the mouthpiece is 'a' hence total pressure force on entrance

$$= \rho g \,.\, a.\, h$$

where h = distance of C.G. of area 'a' from free surface = H.

$$= \rho g \,.\, a \,.\, H \qquad \ldots(i)$$

According to Newton's second law of motion, the net force is equal to the rate of change of momentum.

Now mass of liquid flowing/sec = $\rho \times a_c \times v_c$

The liquid is initially at rest and hence initial velocity is zero but final velocity of fluid is v_c.

$\therefore$ Rate of change of momentum = mass of liquid flowing/sec × [final velocity – initial velocity]

$$= \rho a_c \times v_c[v_c - 0] = \rho a_c v_c^2 \quad ...(ii)$$

Equating (*i*) and (*ii*), we get

$$\rho g \ .a.H. = \rho a_c . v_c^2 \quad ...(iii)$$

Applying Bernoulli's equation to free surface of liquid and section (1)-(1) of Fig. 7.16

$$\frac{p}{\rho g} + \frac{v^2}{2g} + z = \frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1$$

Taking the centre line of mouthpiece as datum, we have

$$z = H,\ z_1 = 0,\ \frac{p}{\rho g} = \frac{p_1}{\rho g} = p_{atmosp.} = 0,$$

$$v_1 = v_c, \qquad v = 0$$

$\therefore$ $$0 + 0 + H = 0 + \frac{v_c^2}{2g} + 0 \text{ or } H = \frac{v_c^2}{2g}$$

$\therefore$ $$v_c = \sqrt{2gH}$$

Substituting the value of v_c in (*iii*), we get

$$\rho g \ . \ a \ . \ H. = \rho \ . \ a_c \ . \ 2g \ . \ H$$

or $$a = 2a_c \text{ or } \frac{a_c}{a} = \frac{1}{2} = 0.5$$

$\therefore$ Co-efficient of contraction, $C_c = \frac{a_c}{a} = 0.5$

Since there is no loss of head, co-efficient of velocity, $C_v = 1.0$

$\therefore$ Co-efficient of discharge, $C_d = C_c \times C_v = 0.5 \times 1.0 = 0.5$

$\therefore$ Discharge $$Q = C_d a\sqrt{2gH} \quad ...(7.19)$$

$$= 0.5 \times a\sqrt{2gH}$$

(*ii*) **Borda's Mouthpiece Running Full.** Fig. 7.17 shows Borda's mouthpiece running full.

Let H = height of liquid above the mouthpiece,

v_1 = velocity at outlet or at (1)-(1) of mouthpiece,

a = area of mouthpiece,

a_c = area of the flow at *C*-*C*,

v_c = velocity of liquid at vena-contracta or at *C*-*C*.

The jet of liquid after passing through *C*-*C*, suddenly enlarges at section (1)-(1). Thus there will be a loss of head due to sudden enlargement.

$\therefore$ $$h_L = \frac{(v_c - v_1)^2}{2g} \quad ...(i)$$

Now from continuity, we have $a_c \times v_c = a_1 \times v_1$

$$\therefore \quad v_c = \frac{a_1}{a_c} \times v_1 = \frac{v_1}{a_c / a_1} = \frac{v_1}{C_c} = \frac{v_1}{0.5} \qquad \{\because C_c = 0.5\}$$

or $$v_c = 2v_1$$

Substituting this value of v_c in (*i*), we get $h_L = \frac{(2v_1 - v_1)^2}{2g} = \frac{v_1^2}{2g}$

Applying Bernoulli's equation to free surface of water in tank and section (1)-(1), we get

$$\frac{p}{\rho g} + \frac{v^2}{2g} + z = \frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 + h_L$$

Taking datum line passing through the centre line of mouthpiece

$$0 + 0 + H = 0 + \frac{v_1^2}{2g} + 0 + \frac{v_1^2}{2g}$$

$$\therefore \quad H = \frac{v_1^2}{2g} + \frac{v_1^2}{2g} = \frac{v_1^2}{g}$$

$$\therefore \quad v_1 = \sqrt{gH}$$

Here v_1 is actual velocity as losses have been taken into consideration,

But theoretical velocity, $v_{th} = \sqrt{2gH}$

$$\therefore \text{ Co-efficient of velocity, } C_v = \frac{v_1}{v_{th}} = \frac{\sqrt{gH}}{\sqrt{2gH}} = \frac{1}{\sqrt{2}} = 0.707$$

As the area of the jet at outlet is equal to the area of the mouthpiece, hence co-efficient of contraction = 1

$$\therefore \quad C_d = C_c \times C_v = 1.0 \times .707 = 0.707$$

$$\therefore \text{ Discharge, } \quad Q = C_d \times a \times \sqrt{2gH} = 0.707 \times a \times \sqrt{2gH} \qquad \text{...(7.20)}$$

Problem 7.29 *An internal mouthpiece of 80 mm diameter is discharging water under a constant head of 8 metres. Find the discharge through mouthpiece, when*

(i) The mouthpiece is running free, and (ii) The mouthpiece is running full.

Solution. Given :

Dia. of mouthpiece, $d = 80$ mm $= 0.08$ m

$$\therefore \text{ Area, } \quad a = \frac{\pi}{4}(.08)^2 = .005026 \text{ m}^2$$

Constant head, $H = 4$ m.

(*i*) **Mouthpiece running free.** The discharge, Q is given by equation (7.19) as

$$Q = 0.5 \times a \times \sqrt{2gH}$$

$$= 0.5 \times .005026 \times \sqrt{2 \times 9.81 \times 4.0}$$

$$= 0.02226 \text{ m}^3/\text{s} = \mathbf{22.26 \text{ litres/s. Ans.}}$$

(*ii*) **Mouthpiece running full.** The discharge, Q is given by equation (7.20) as

$$Q = 0.707 \times a \times \sqrt{2gH}$$
$$= 0.707 \times .005026 \times \sqrt{2 \times 9.81 \times 4.0}$$
$$= 0.03147 \text{ m}^3/\text{s} = \textbf{31.47 litre/s. Ans.}$$

HIGHLIGHTS

1. Orifice is a small opening on the side or at the bottom of a tank while mouthpiece is a short length of pipe which is two or three times its diameter in length.
2. Orifices as well as mouthpieces are used for measuring the rate of flow of liquid.
3. Theoretical velocity of jet of water from orifice is given by
$$V = \sqrt{2gH}$$, where H = Height of water from the centre of orifice.
4. There are three hydraulic co-efficients namely :

   (*a*) Co-efficient of velocity, $C_v = \dfrac{\text{Actual velocity at vena - contracta}}{\text{Theoretical velocity}} = \dfrac{x}{\sqrt{4yH}}$

   (*b*) Co-efficient of contraction, $C_c = \dfrac{\text{Area of jet at vena - contracta}}{\text{Area of orifice}}$

   (*c*) Co-efficient of discharge, $C_d = \dfrac{\text{Actual discharge}}{\text{Theoretical discharge}} = C_v \times C_c$

   where x and y are the co-ordinates of any point of jet of water from vena-contracta.
5. A large orifice is one, where the head of liquid above the centre of orifice is less than 5 times the depth of orifice. The discharge through a large rectangular orifice is
$$Q = \frac{2}{3} C_d \times b \times \sqrt{2g} [H_2^{3/2} - H_1^{3/2}]$$
   where b = Width of orifice,
   C_d = Co-efficient of discharge for orifice,
   H_1 = Height of liquid above top edge of orifice, and
   H_2 = Height of liquid above bottom edge of orifice.
6. The discharge through fully sub-merged orifice, $Q = C_d \times b \times (H_2 - H_1) \times \sqrt{2gH}$
   where b = Width of orifice,
   C_d = Co-efficient of discharge for orifice,
   H_2 = Height of liquid above bottom edge of orifice on upstream side,
   H_1 = Height of liquid above top edge of orifice on upstream side,
   H = Difference of liquid levels on both sides of the orifice.
7. Discharge through partially sub-merged orifice,
$$Q = Q_1 + Q_2$$
$$= C_d b (H_2 - H) \times \sqrt{2gH} + 2/3\ C_d b \times \sqrt{2g}\ [H^{3/2} - H_1^{3/2}]$$
   where b = Width of orifice
   C_d, H_1, H_2 and H are having their usual meaning.
8. Time of emptying a tank through an orifice at its bottom is given by,
$$T = \frac{2A\left[\sqrt{H_1} - \sqrt{H_2}\right]}{C_d \,.\, a \,.\, \sqrt{2g}}$$
   where H_1 = Initial height of liquid in tank,
   H_2 = Final height of liquid in tank,

A = Area of tank,
a = Area of orifice,
C_d = Co-efficient of discharge.

If the tank is to be completely emptied, then time T,

$$T = \frac{2A\sqrt{H}}{C_d \,.\, a \,.\, \sqrt{2g}}.$$

9. Time of emptying a hemispherical tank by an orifice fitted at its bottom,

$$T = \frac{\pi}{C_d \,.\, a \,.\, \sqrt{2g}}\left[\frac{4}{3} R\left(H_1^{3/2} - H_2^{3/2}\right) - \frac{2}{5}\left(H_1^{5/2} - H_2^{5/2}\right)\right]$$

and for completely emptying the tank, $T = \dfrac{\pi}{C_d \,.\, a \,.\, \sqrt{2g}}\left[\dfrac{4}{3} RH_1^{3/2} - \dfrac{2}{5} H_1^{5/2}\right]$

where R = Radius of the hemispherical tank,
H_1 = Initial height of liquid,
H_2 = Final height of liquid,
a = Area of orifice, and
C_d = Co-efficient of discharge.

10. Time of emptying a circular horizontal tank by an orifice at the bottom of the tank,

$$T = \frac{4L}{3C_d \,.\, a \,.\, \sqrt{2g}}\ [(2R - \mathrm{H}_2)^{3/2} - (2R - H_1)^{3/2}]$$

and for completely emptying the tank, $T = \dfrac{4L}{3C_d \,.\, a \,.\, \sqrt{2g}}\ [(2R)^{3/2} - (2R - H_1)^{3/2}]$

where L = Length of horizontal tank.

11. Co-efficient of discharge for,

(*i*) External mouthpiece, $C_d = 0.855$
(*ii*) Internal mouthpiece, running full, $C_d = 0.707$
(*iii*) Internal mouthpiece, running free, $C_d = 0.50$
(*iv*) Convergent or convergent-divergent, $C_d = 1.0$.

12. For an external mouthpiece, absolute pressure head at vena-contracta

$$H_c = H_a - 0.89\ H$$

where H_a = atmospheric pressure head = 10.3 m of water
H = head of liquid above the mouthpiece.

13. For a convergent-divergent mouthpiece, the ratio of areas at outlet and at vena-contracta is

$$\frac{a_1}{a_c} = \sqrt{1 + \frac{H_a - H_c}{H}}$$

where a_1 = Area of mouthpiece at outlet
a_c = Area of mouthpiece at vena-contracta
H_a = Atmospheric pressure head
H_c = Absolute pressure head at vena-contracta
H = Height of liquid above mouthpiece.

14. In case of internal mouthpieces, if the jet of liquid comes out from mouthpiece without touching its sides it is known as running free. But if the jet touches the sides of the mouthpiece, it is known as running full.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define an orifice and a mouthpiece. What is the difference between the two ?
2. Explain the classification of orifices and mouthpieces based on their shape, size and sharpness ?
3. What are hydraulic co-efficients ? Name them.
4. Define the following co-efficients : (*i*) Co-efficient of velocity, (*ii*) Co-efficient of contraction and (*iii*) Co-efficient of discharge.
5. Derive the expression $C_d = C_v \times C_c$.
6. Define vena-contracta.
7. Differentiate between a large and a small orifice. Obtain an expression for discharge through a large rectangular orifice.
8. What do you understand by the terms wholly sub-merged orifice and partially sub-merged orifice ?
9. Prove that the expression for discharge through an external mouthpiece is given by

$$Q = .855 \times a \times v$$

where a = Area of mouthpiece at outlet and
v = Velocity of jet of water at outlet.

10. Distinguish between : (*i*) External mouthpiece and internal mouthpiece, (*ii*) Mouthpiece running free and mouthpiece running full.
11. Obtain an expression for absolute pressure head at vena-contracta for an external mouthpiece.
12. What is a convergent-divergent mouthpiece ? Obtain an expression for the ratio of diameters at outlet and at vena-contracta for a convergent-divergent 'mouthpiece' in terms of absolute pressure head at vena-contracta, head of liquid above mouthpiece and atmospheric pressure head.
13. The length of the divergent outlet part in a venturimeter is usually made longer compared with that of the converging inlet part. Why ?
14. Justify the statement, "In a convergent-divergent mouthpiece the loss of head is practically eliminated".

(B) NUMERICAL PROBLEMS

1. The head of water over an orifice of diameter 50 mm is 12 m. Find the actual discharge and actual velocity of jet at vena-contracta. Take $C_d = 0.6$ and $C_v = 0.98$. [**Ans.** .018 m^3/s ; 15.04 m/s]
2. The head of water over the centre of an orifice of diameter 30 mm is 1.5 m. The actual discharge through the orifice is 2.35 litres/sec. Find the co-efficient of discharge. [**Ans.** 0.613]
3. A jet of water, issuing from a sharp edged vertical orifice under a constant head of 60 cm, has the horizontal and vertical co-ordinates measured from the vena-contracta at a certain point as 10.0 cm and 0.45 cm respectively. Find the value of C_v. Also find the value of C_v if $C_d = 0.60$. [**Ans.** 0.962, 0.623]
4. The head of water over an orifice of diameter 100 mm is 5 m. The water coming out from orifice is collected in a circular tank of diameter 2 m. The rise of water level in circular tank is .45 m in 30 seconds. Also the co-ordinates of a certain point on the jet, measured from vena-contracta are 100 cm horizontal and 5.2 cm vertical. Find the hydraulic co-efficients C_d, C_v and C_c. [**Ans.** 0.605, 0.98, 0.617]
5. A tank has two identical orifices in one of its vertical sides. The upper orifice is 4 m below the water surface and lower one 6 m below the water surface. If the value of C_v for each orifice is 0.98, find the point of intersection of the two jets. [**Ans.** At a horizontal distance of 9.60 cm]
6. A closed vessel contains water upto a height of 2.0 m and over the water surface there is air having pressure 8.829 N/cm^2 above atmospheric pressure. At the bottom of the vessel there is an orifice of diameter 15 cm. Find the rate of flow of water from orifice. Take $C_d = 0.6$. [**Ans.** 0.15575 m^3/s]

7. A closed tank partially filled with water upto a height of 1 m, having an orifice of diameter 20 mm at the bottom of the tank. Determine the pressure required for a discharge of 3.0 litres/s through the orifice. Take $C_d = 0.62$. [**Ans.** 10.88 N/cm^2]

8. Find the discharge through a rectangular orifice 3.0 m wide and 2 m deep fitted to a water tank. The water level in the tank is 4 m above the top edge of the orifice. Take $C_d = 0.62$ [**Ans.** 36.77 m^3/s]

9. A rectangular orifice, 2.0 m wide and 1.5 m deep is discharging water from a tank. If the water level in the tank is 3.0 m above the top edge of the orifice, find the discharge through the orifice. Take $C_d = 0.6$. [**Ans.** 15.40 m^3/s]

10. A rectangular orifice, 1.0 m wide and 1.5 m deep is discharging water from a vessel. The top edge of the orifice is 0.8 m below the water surface in the vessel. Calculate the discharge through the orifice if $C_d = 0.6$. Also calculate the percentage error if the orifice is treated as a small orifice. [**Ans.** 1.058%]

11. Find the discharge through a fully sub-merged orifice of width 2 m if the difference of water levels on both the sides of the orifice be 800 mm. The height of water from top and bottom of the orifice are 2.5 m and 3 m respectively. Take $C_d = 0.6$. [**Ans.** 2.377 m^3/s]

12. Find the discharge through a totally drowned orifice 1.5 m wide and 1 m deep, if the difference of water levels on both the sides of the orifice be 2.5 m. Take $C_d = 0.62$. [**Ans.** 6.513 m^3/s]

13. A rectangular orifice of 1.5 m wide and 1.2 m deep is fitted in one side of a large tank. The water level on one side of the orifice is 2 m above the top edge of the orifice, while on the other side of the orifice, the water level is 0.4 m below its top edge. Calculate the discharge through the orifice if $C_d = 0.62$. [**Ans.** 7.549 m^3/s]

14. A circular tank of diameter 3 m contains water upto a height of 4 m. The tank is provided with an orifice of diameter 0.4 m at the bottom. Find the time taken by water : (*i*) to fall from 4 m to 2 m and (*ii*) for completely emptying the tank. Take $C_d = 0.6$. [**Ans.** (*i*) 24.8 s, (*ii*) 84.7 s]

15. A circular tank of diameter 1.5 m contains water upto a height of 4 m. An orifice of 40 mm diameter is provided at its bottom. If $C_d = 0.62$, find the height of water above the orifice after 10 minutes. [**Ans.** 2 m]

16. A hemispherical tank of diameter 4 m contains water upto a height of 2.0 m. An orifice of diameter 50 mm is provided at the bottom. Find the time required by water (*i*) to fall from 2.0 m to 1.0 m (*ii*) for completely emptying the tank. Take $C_d = 0.6$ [**Ans.** (*i*) 30 min 14.34 s, (*ii*) 52 min 59 s]

17. A hemispherical cistern of 4 m radius is full of water. It is fitted with a 60 mm diameter sharp edged orifice at the bottom. Calculate the time required to lower the level in the cistern by 2 metres. Take $C_d = 0.6$. [**Ans.** 1 hr 58 min 45.9 s]

18. A cylindrical tank is having a hemispherical base. The height of cylindrical portion is 4 m and diameter is 3 m. At the bottom of this tank an orifice of diameter 300 mm is fitted. Find the time required to completely emptying the tank. Take $C_d = 0.6$. [**Ans.** 2 min 7.37 s]

19. An orifice of diameter 200 mm is fitted at the bottom of a boiler drum of length 6 m and of diameter 2 m. The drum is horizontal and half full of water. Find the time required to empty the boiler, given the value of $C_d = 0.6$ [**Ans.** 2 min 55.20 s]

20. An orifice of diameter 150 mm is fitted at the bottom of a boiler drum of length 6 m and of diameter 2 m. The drum is horizontal and contains water upto a height of 1.8 m. Find the time required to empty the boiler. Take $C_d = 0.6$. [**Ans.** 7 min 46.64 s]

21. Find the discharge from a 80 mm diameter external mouthpiece, fitted to a side of a large vessel if the head over the mouthpiece is 6 m. [**Ans.** 0.0466 m^3/s]

22. An external cylindrical mouthpiece of diameter 100 mm is discharging water under a constant head of 8 m. Determine the discharge and absolute pressure head of water at vena-contracta. Take $C_d = 0.855$ and C_c for vena-contracta = 0.62. Take atmospheric pressure head = 10.3 m of water. [**Ans.** 0.084 m^3/s ; 3.18 m]

23. A convergent-divergent mouthpiece having throat diameter of 60 mm is discharging water under a constant head of 3.0 m. Determine the maximum outlet diameter for maximum discharge. Find maximum discharge also. Take atmospheric pressure head = 10.3 m of water and separation pressure head = 2.5 m of water absolute. [**Ans.** 6.88 cm, $Q_{max} = 0.01506$ m^3/s]

24. The throat and exit diameter of a convergent-divergent mouthpiece are 40 mm and 80 mm respectively. It is fitted to the vertical side of a tank, containing water. Find the maximum head of water for steady flow. The maximum vacuum pressure is 8 m of water. Take atmospheric pressure head = 10.3 m of water.
[**Ans.** 0.533 m]

25. The discharge through a convergent-divergent mouthpiece fitted to the side of a tank under a constant head of 2 m is 7 litres/s. The head loss in the divergent portion is 0.10 times the kinetic head at outlet. Find the throat and exit diameters, if separation pressure head = 2.5 m and atmospheric pressure head = 10.3 m of water. [**Ans.** 25.3 mm ; 38.6 mm]

26. An internal mouthpiece of 100 mm diameter is discharging water under a constant head of 5 m. Find the discharge through mouthpiece, when
(*i*) the mouthpiece is running free, and (*ii*) the mouthpiece is running full.
[**Ans.** (*i*) 38.8 litres/s, (*ii*) 54.86 litres/s]

8 CHAPTER

NOTCHES AND WEIRS

▶ 8.1 INTRODUCTION

A **notch** is a device used for measuring the rate of flow of a liquid through a small channel or a tank. It may be defined as an opening in the side of a tank or a small channel in such a way that the liquid surface in the tank or channel is below the top edge of the opening.

A **weir** is a concrete or masonary structure, placed in an open channel over which the flow occurs. It is generally in the form of vertical wall, with a sharp edge at the top, running all the way across the open channel. The notch is of small size while the weir is of a bigger size. The notch is generally made of metallic plate while weir is made of concrete or masonary structure.

1. **Nappe or Vein.** The sheet of water flowing through a notch or over a weir is called Nappe or Vein.
2. **Crest or Sill.** The bottom edge of a notch or a top of a weir over which the water flows, is known as the sill or crest.

▶ 8.2 CLASSIFICATION OF NOTCHES AND WEIRS

The notches are classified as :

1. According to the shape of the opening :
   - (*a*) Rectangular notch,
   - (*b*) Triangular notch,
   - (*c*) Trapezoidal notch, and
   - (*d*) Stepped notch.
2. According to the effect of the sides on the nappe :
   - (*a*) Notch with end contraction.
   - (*b*) Notch without end contraction or suppressed notch.

Weirs are classified according to the shape of the opening, the shape of the crest, the effect of the sides on the nappe and nature of discharge. The following are important classifications.

(*a*) According to the shape of the opening :

(*i*) Rectangular weir, (*ii*) Triangular weir, and

(*iii*) Trapezoidal weir (Cipolletti weir)

(*b*) According to the shape of the crest :

(*i*) Sharp-crested weir, (*ii*) Broad-crested weir,

(*iii*) Narrow-crested weir, and (*iv*) Ogee-shaped weir.

(*c*) According to the effect of sides on the emerging nappe :
(*i*) Weir with end contraction, and (*ii*) Weir without end contraction.

▶ 8.3 DISCHARGE OVER A RECTANGULAR NOTCH OR WEIR

The expression for discharge over a rectangular notch or weir is the same.

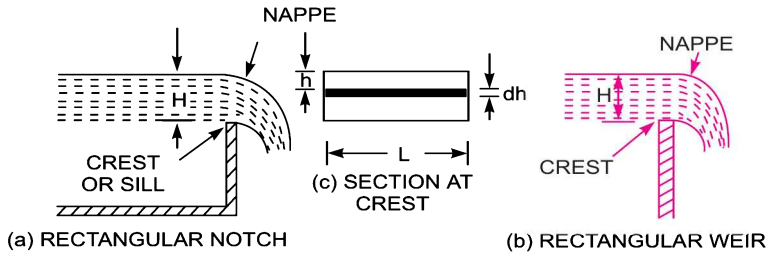


Fig. 8.1 *Rectangular notch and weir.*

Consider a rectangular notch or weir provided in a channel carrying water as shown in Fig. 8.1.

Let H = Head of water over the crest

L = Length of the notch or weir

For finding the discharge of water flowing over the weir or notch, consider an elementary horizontal strip of water of thickness dh and length L at a depth h from the free surface of water as shown in Fig. 8.1(*c*).

The area of strip $= L \times dh$

and theoretical velocity of water flowing through strip $= \sqrt{2gh}$

The discharge dQ, through strip is

$$dQ = C_d \times \text{Area of strip} \times \text{Theoretical velocity}$$

$$= C_d \times L \times dh \times \sqrt{2gh} \qquad ...(i)$$

where C_d = Co-efficient of discharge.

The total discharge, Q , for the whole notch or weir is determined by integrating equation (*i*) between the limits 0 and H.

$$\therefore \quad Q = \int_0^H C_d \,.\, L \,.\, \sqrt{2gh} \,.\, dh = C_d \times L \times \sqrt{2g} \int_0^H h^{1/2}\, dh$$

$$= C_d \times L \times \sqrt{2g} \left[\frac{h^{1/2+1}}{\frac{1}{2}+1} \right]_0^H = C_d \times L \times \sqrt{2g} \left[\frac{h^{3/2}}{3/2} \right]_0^H$$

$$= \frac{2}{3} C_d \times L \times \sqrt{2g}\ [H]^{3/2}. \qquad ...(8.1)$$

Problem 8.1 *Find the discharge of water flowing over a rectangular notch of 2 m length when the constant head over the notch is 300 mm. Take C_d = 0.60.*

Solution. Given :

Length of the notch, L = 2.0 m

Head over notch, $H = 300 \text{ m} = 0.30 \text{ m}$

$C_d = 0.60$

Discharge, $Q = \frac{2}{3} C_d \times L \times \sqrt{2g} \left[H^{3/2}\right]$

$= \frac{2}{3} \times 0.6 \times 2.0 \times \sqrt{2 \times 9.81} \times [0.30]^{1.5} \text{ m}^3/\text{s}$

$= 3.5435 \times 0.1643 = \mathbf{0.582 \text{ m}^3/\text{s. Ans.}}$

Problem 8.2 *Determine the height of a rectangular weir of length 6 m to be built across a rectangular channel. The maximum depth of water on the upstream side of the weir is 1.8 m and discharge is 2000 litres/s. Take $C_d = 0.6$ and neglect end contractions.*

Solution. Given :

Length of weir, $L = 6 \text{ m}$

Depth of water, $H_1 = 1.8 \text{ m}$

Discharge, $Q = 2000 \text{ lit/s} = 2 \text{ m}^3/\text{s}$

$C_d = 0.6$

Let H is height of water above the crest of weir, and H_2 = height of weir (Fig. 8.2)

The discharge over the weir is given by the equation (8.1) as

$$Q = \frac{2}{3} C_d \times L \times \sqrt{2g}\ H^{3/2}$$

or $$2.0 = \frac{2}{3} \times 0.6 \times 6.0 \times \sqrt{2 \times 9.81} \times H^{3/2}$$

$$= 10.623\ H^{3/2}$$

$$\therefore \quad H^{3/2} = \frac{2.0}{10.623}$$

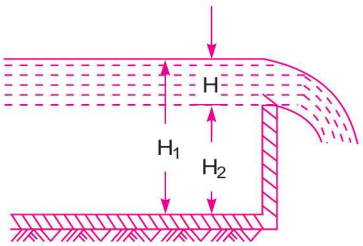


Fig. 8.2

$$\therefore \quad H = \left(\frac{2.0}{10.623}\right)^{2/3} = 0.328 \text{ m}$$

$\therefore$ Height of weir, $H_2 = H_1 - H$

= Depth of water on upstream side $- H$

$= 1.8 - .328 = \mathbf{1.472 \text{ m. Ans.}}$

Problem 8.3 *The head of water over a rectangular notch is 900 mm. The discharge is 300 litres/s. Find the length of the notch, when $C_d = 0.62$.*

Solution. Given :

Head over notch, $H = 90 \text{ cm} = 0.9 \text{ m}$

Discharge, $Q = 300 \text{ lit/s} = 0.3 \text{ m}^3/\text{s}$

$C_d = 0.62$

Let length of notch $= L$

Using equation (8.1), we have

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$

or $$0.3 = \frac{2}{3} \times 0.62 \times L \times \sqrt{2 \times 9.81} \times (0.9)^{3/2}$$

$$= 1.83 \times L \times 0.8538$$

$\therefore$ $$L = \frac{0.3}{1.83 \times .8538} = .192 \text{ m} = \textbf{192 mm. Ans.}$$

▶ 8.4 DISCHARGE OVER A TRIANGULAR NOTCH OR WEIR

The expression for the discharge over a triangular notch or weir is the same. It is derived as :

Let H = head of water above the V- notch

θ = angle of notch

Consider a horizontal strip of water of thickness 'dh' at a depth of h from the free surface of water as shown in Fig. 8.3.

From Fig. 8.3 (b), we have

$$\tan \frac{\theta}{2} = \frac{AC}{OC} = \frac{AC}{(H-h)}$$

$\therefore$ $$AC = (H-h) \tan \frac{\theta}{2}$$

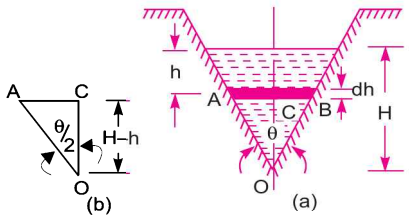


Fig. 8.3 *The triangular notch.*

Width of strip $$= AB = 2AC = 2 (H - h) \tan \frac{\theta}{2}$$

$\therefore$ Area of strip $$= 2 (H - h) \tan \frac{\theta}{2} \times dh$$

The theoretical velocity of water through strip = $\sqrt{2gh}$

$\therefore$ Discharge, through the strip,

$$dQ = C_d \times \text{Area of strip} \times \text{ Velocity (theoretical)}$$

$$= C_d \times 2 (H - h) \tan \frac{\theta}{2} \times dh \times \sqrt{2gh}$$

$$= 2C_d (H - h) \tan \frac{\theta}{2} \times \sqrt{2gh} \times dh$$

$\therefore$ Total discharge, $$Q = \int_0^H 2C_d (H - h) \tan \frac{\theta}{2} \times \sqrt{2gh} \times dh$$

$$= 2C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \int_0^H (H - h) h^{1/2} \, dh$$

$$= 2 \times C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \int_0^H (Hh^{1/2} - h^{3/2}) \, dh$$

$$= 2 \times C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \left[\frac{Hh^{3/2}}{3/2} - \frac{h^{5/2}}{5/2} \right]_0^H$$

$$= 2 \times C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \left[\frac{2}{3} H.H^{3/2} - \frac{2}{5} H^{5/2} \right]$$

$$= 2 \times C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \left[\frac{2}{3} H^{5/2} - \frac{2}{5} H^{5/2} \right]$$

$$= 2 \times C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \left[\frac{4}{15} H^{5/2} \right]$$

$$= \frac{8}{15} C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2} \quad ...(8.2)$$

For a right-angled V-notch, if $C_d = 0.6$

$$\theta = 90°, \quad \therefore \quad \tan \frac{\theta}{2} = 1$$

Discharge, $$Q = \frac{8}{15} \times 0.6 \times 1 \times \sqrt{2 \times 9.81} \times H^{5/2} \quad ...(8.3)$$

$$= 1.417 \ H^{5/2}.$$

Problem 8.4 *Find the discharge over a triangular notch of angle 60° when the head over the V-notch is 0.3 m. Assume $C_d = 0.6$.*

Solution. Given :

Angle of V-notch, $\theta = 60°$

Head over notch, $H = 0.3$ m

$C_d = 0.6$

Discharge, Q over a V-notch is given by equation (8.2)

$$Q = \frac{8}{15} \times C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2}$$

$$= \frac{8}{15} \times 0.6 \tan \frac{60°}{2} \times \sqrt{2 \times 9.81} \times (0.3)^{5/2}$$

$$= 0.8182 \times 0.0493 = \mathbf{0.040\ m^3/s.\ Ans.}$$

Problem 8.5 *Water flows over a rectangular weir 1 m wide at a depth of 150 mm and afterwards passes through a triangular right-angled weir. Taking C_d for the rectangular and triangular weir as 0.62 and 0.59 respectively, find the depth over the triangular weir.*

Solution. Given :

For rectangular weir, length, $L = 1$ m

Depth of water, $H = 150$ mm $= 0.15$ m

$C_d = 0.62$

For triangular weir, $\theta = 90°$

$C_d = 0.59$

Let depth over triangular weir $= H_1$

The discharge over the rectangular weir is given by equation (8.1) as

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$

$$= \frac{2}{3} \times 0.62 \times 1.0 \times \sqrt{2 \times 9.81} \times (.15)^{3/2} \text{ m}^3/\text{s} = 0.10635 \text{ m}^3/\text{s}$$

The same discharge passes through the triangular right-angled weir. But discharge, Q, is given by equation (8.2) for a triangular weir as

$$Q = \frac{8}{15} \times C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2}$$

$$\therefore \quad 0.10635 = \frac{8}{15} \times .59 \times \tan \frac{90^\circ}{2} \times \sqrt{2g} \times H_1^{5/2} \qquad \{\because \theta = 90^\circ \text{ and } H = H_1\}$$

$$= \frac{8}{15} \times .59 \times 1 \times 4.429 \times H_1^{5/2} = 1.3936 \, H_1^{5/2}$$

$$\therefore \quad H_1^{5/2} = \frac{0.10635}{1.3936} = 0.07631$$

$$\therefore \quad H_1 = (.07631)^{0.4} = \mathbf{0.3572 \text{ m. Ans.}}$$

Problem 8.5A *Water flows through a triangular right-angled weir first and then over a rectangular weir of 1 m width. The discharge co-efficients of the triangular and rectangular weirs are 0.6 and 0.7 respectively. If the depth of water over the triangular weir is 360 mm, find the depth of water over the rectangular weir.*

Solution. Given :

For triangular weir : $\theta = 90^\circ$, $C_d = 0.6$, $H = 360$ mm $= 0.36$ m

For rectangular weir : $L = 1$ m, $C_d = 0.7$, $H = ?$

The discharge for a triangular weir is given by equation (8.2) as

$$Q = \frac{8}{15} \times C_d \times \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2}$$

$$= \frac{8}{15} \times 0.6 \times \tan \left(\frac{90^\circ}{2} \right) \times \sqrt{2 \times 9.81} \times (0.36)^{5/2} = 0.1102 \text{ m}^3/\text{s}$$

The same discharge is passing through the rectangular weir. But discharge for a rectangular weir is given by equation (8.1) as

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$

or $$0.1102 = \frac{2}{3} \times 0.7 \times 1 \times \sqrt{2 \times 9.81} \times H^{3/2} = 2.067 \, H^{3/2}$$

or $$H^{3/2} = \frac{0.1102}{2.067} = 0.0533$$

$$\therefore \quad H = (0.0533)^{2/3} = 0.1415 \text{ m} = \mathbf{141.5 \text{ mm. Ans.}}$$

Problem 8.6 *A rectangular channel 2.0 m wide has a discharge of 250 litres per second, which is measured by a right-angled V-notch weir. Find the position of the apex of the notch from the bed of the channel if maximum depth of water is not to exceed 1.3 m. Take C_d = 0.62.*

Solution. Given :

Width of rectangular channel, L = 2.0 m

Discharge, Q = 250 lit/s = 0.25 m^3/s

Depth of water in channel = 1.3 m

Let the height of water over V-notch = H

The rate of flow through V-notch is given by equation (8.2) as

$$Q = \frac{8}{15} \times C_d \times \sqrt{2g} \times \tan\frac{\theta}{2} \times H^{5/2}$$

where C_d = 0.62, $\theta = 90°$

$$\therefore \quad Q = \frac{8}{15} \times .62 \times \sqrt{2 \times 9.81} \times \tan\frac{90°}{2} \times H^{5/2}$$

or $$0.25 = \frac{8}{15} \times .62 \times 4.429 \times 1 \times H^{5/2}$$

or $$H^{5/2} = \frac{.25 \times 15}{8 \times .62 \times 4.429} = 0.1707$$

$$\therefore \quad H = (.1707)^{2/5} = (.1707)^{0.4} = 0.493 \text{ m}$$

Position of apex of the notch from the bed of channel

= depth of water in channel–height of water over V-notch

= 1.3 – .493 = **0.807 m. Ans.**

▶ 8.5 ADVANTAGES OF TRIANGULAR NOTCH OR WEIR OVER RECTANGULAR NOTCH OR WEIR

A triangular notch or weir is preferred to a rectangular weir or notch due to following reasons :

1. The expression for discharge for a right-angled V-notch or weir is very simple.
2. For measuring low discharge, a triangular notch gives more accurate results than a rectangular notch.
3. In case of triangular notch, only one reading, *i.e.*, H is required for the computation of discharge.
4. Ventilation of a triangular notch is not necessary.

▶ 8.6 DISCHARGE OVER A TRAPEZOIDAL NOTCH OR WEIR

As shown in Fig. 8.4, a trapezoidal notch or weir is a combination of a rectangular and triangular notch or weir. Thus the total discharge will be equal to the sum of discharge through a rectangular weir or notch and discharge through a triangular notch or weir.

Let H = Height of water over the notch

L = Length of the crest of the notch

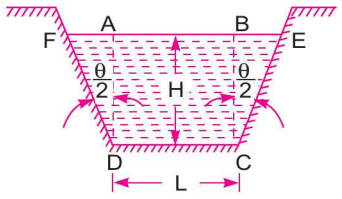


Fig. 8.4 *The trapezoidal notch.*

C_{d_1} = Co-efficient of discharge for rectangular portion *ABCD* of Fig. 8.4.

C_{d_2} = Co-efficient of discharge for triangular portion [*FAD* and *BCE*]

The discharge through rectangular portion *ABCD* is given by (8.1)

or
$$Q_1 = \frac{2}{3} \times C_{d_1} \times L \times \sqrt{2g} \times H^{3/2}$$

The discharge through two triangular notches *FDA* and *BCE* is equal to the discharge through a single triangular notch of angle θ and it is given by equation (8.2) as

$$Q_2 = \frac{8}{15} \times C_{d_2} \times \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2}$$

∴ Discharge through trapezoidal notch or weir *FDCEF* = $Q_1 + Q_2$

$$= \frac{2}{3} C_{d_1} L \sqrt{2g} \times H^{3/2} + \frac{8}{15} C_{d_2} \times \tan \theta/2 \times \sqrt{2g} \times H^{5/2}. \quad ...(8.4)$$

Problem 8.7 *Find the discharge through a trapezoidal notch which is 1 m wide at the top and 0.40 m at the bottom and is 30 cm in height. The head of water on the notch is 20 cm. Assume C_d for rectangular portion = 0.62 while for triangular portion = 0.60.*

Solution. Given :

Top width, $AE = 1$ m

Base width, $CD = L = 0.4$ m

Head of water, $H = 0.20$ m

For rectangular portion, $C_{d_1} = 0.62$

For triangular portion, $C_{d_2} = 0.60$

From ΔABC, we have

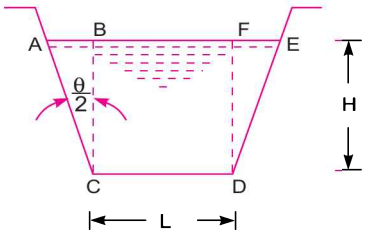


Fig. 8.5

$$\tan \frac{\theta}{2} = \frac{AB}{BC} = \frac{(AE - CD)/2}{H}$$

$$= \frac{(1.0 - 0.4)/2}{0.3} = \frac{0.6/2}{0.3} = \frac{0.3}{0.3} = 1$$

Discharge through trapezoidal notch is given by equation (8.4)

$$Q = \frac{2}{3} C_{d_1} \times L \times \sqrt{2g} \times H^{3/2} + \frac{8}{15} C_{d_2} \times \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2}$$

$$= \frac{2}{3} \times 0.62 \times 0.4 \times \sqrt{2 \times 9.81} \times (0.2)^{3/2} + \frac{8}{15} \times .60 \times 1 \times \sqrt{2 \times 9.81} \times (0.2)^{5/2}$$

$= 0.06549 + 0.02535 = 0.09084$ m^3/s = **90.84 litres/s. Ans.**

▶ 8.7 DISCHARGE OVER A STEPPED NOTCH

A stepped notch is a combination of rectangular notches. The discharge through stepped notch is equal to the sum of the discharges through the different rectangular notches.

Consider a stepped notch as shown in Fig. 8.6.

Let H_1 = Height of water above the crest of notch 1,

L_1 = Length of notch 1,

H_2, L_2 and H_3, L_3 are corresponding values for notches 2 and 3 respectively.

C_d = Co-efficient of discharge for all notches

∴ Total discharge $Q = Q_1 + Q_2 + Q_3$

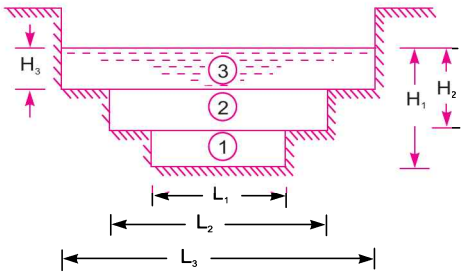


Fig. 8.6 *The stepped notch.*

or $$Q = \frac{2}{3} \times C_d \times L_1 \times \sqrt{2g}\ [H_1^{3/2} - H_2^{3/2}]$$

$$+ \frac{2}{3}\ C_d \times L_2 \times \sqrt{2g}\,[H_2^{3/2} - H_3^{3/2}] + \frac{2}{3}\ C_d \times L_3 \times \sqrt{2g}\ \times H_3^{3/2}. \quad ...(8.5)$$

Problem 8.8 *Fig. 8.7 shows a stepped notch. Find the discharge through the notch if C_d for all section = 0.62.*

Solution. Given :

$L_1 = 40$ cm, $L_2 = 80$ cm,

$L_3 = 120$ cm

$H_1 = 50 + 30 + 15 = 95$ cm,

$H_2 = 80$ cm, $H_3 = 50$ cm,

$C_d = 0.62$

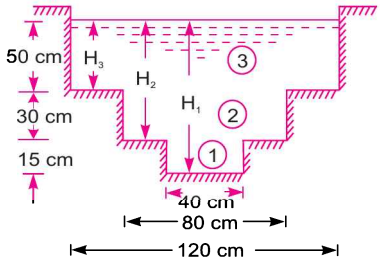


Fig. 8.7

Total discharge, $Q = Q_1 + Q_2 + Q_3$

where $$Q_1 = \frac{2}{3} \times C_d \times L_1 \times \sqrt{2g}\ [H_1^{3/2} - H_2^{3/2}]$$

$$= \frac{2}{3} \times 0.62 \times 40 \times \sqrt{2 \times 981} \times [95^{3/2} - 80^{3/2}]$$

$$= 732.26[925.94 - 715.54] = 154067 \text{ cm}^3/\text{s} = 154.067 \text{ lit/s}$$

$$Q_2 = \frac{2}{3} \times C_d \times L_2 \times \sqrt{2g} \times [H_2^{3/2} - H_3^{3/2}]$$

$$= \frac{2}{3} \times 0.62 \times 80 \times \sqrt{2 \times 981} \times [80^{3/2} - 50^{3/2}]$$

$$= 1464.52[715.54 - 353.55] \text{ cm}^3/\text{s} = 530141 \text{ cm}^3/\text{s} = 530.144 \text{ lit/s}$$

and $$Q_3 = \frac{2}{3} \times C_d \times L_3 \times \sqrt{2g} \times H_3^{3/2}$$

$$= \frac{2}{3} \times 0.62 \times 120 \times \sqrt{2 \times 981} \times 50^{3/2} = 776771 \text{ cm}^3/\text{s} = 776.771 \text{ lit/s}$$

∴ $Q = Q_1 + Q_2 + Q_3 = 154.067 + 530.144 + 776.771$

$$= \mathbf{1460.98 \text{ lit/s. Ans.}}$$

▶ 8.8 EFFECT ON DISCHARGE OVER A NOTCH OR WEIR DUE TO ERROR IN THE MEASUREMENT OF HEAD

For an accurate value of the discharge over a weir or notch, an accurate measurement of head over the weir or notch is very essential as the discharge over a triangular notch is proportional to $H^{5/2}$ and in case of rectangular notch it is proportional to $H^{3/2}$. A small error in the measurement of head, will affect the discharge considerably. The following cases of error in the measurement of head will be considered :

(*i*) For Rectangular Weir or Notch.

(*ii*) For Triangular Weir or Notch.

8.8.1 For Rectangular Weir or Notch. The discharge for a rectangular weir or notch is given by equation (8.1) as

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$
$$= KH^{3/2} \qquad ...(i)$$

where $K = \frac{2}{3} C_d \times L \times \sqrt{2g}$

Differentiating the above equation, we get

$$dQ = K \times \frac{3}{2} H^{1/2} \, dH \qquad ...(ii)$$

Dividing (*ii*) by (*i*), $$\frac{dQ}{Q} = \frac{K \times \frac{3}{2} \times H^{1/2} dH}{KH^{3/2}} = \frac{3}{2} \frac{dH}{H} \qquad ...(8.6)$$

Equation (8.6) shows that an error of 1% in measuring H will produce 1.5% error in discharge over a rectangular weir or notch.

8.8.2 For Triangular Weir or Notch. The discharge over a triangular weir or notch is given by equation (8.2) as

$$Q = \frac{8}{15} C_d . \tan \frac{\theta}{2} \sqrt{2g} \times H^{5/2}$$
$$= KH^{5/2} \qquad ...(iii)$$

where $K = \frac{8}{15} C_d . \tan \frac{\theta}{2} \sqrt{2g}$

Differentiating equation (*iii*), we get

$$dQ = K \frac{5}{2} H^{3/2} \times dH \qquad ...(iv)$$

Dividing (*iv*) by (*iii*), we get $$\frac{dQ}{Q} = \frac{K \frac{5}{2} H^{3/2} \, dH}{KH^{5/2}} = \frac{5}{2} \frac{dH}{H} \qquad ...(8.7)$$

Equation (8.7) shows that an error of 1% in measuring H will produce 2.5% error in discharge over a triangular weir or notch.

Problem 8.9 *A rectangular notch 40 cm long is used for measuring a discharge of 30 litres per second. An error of 1.5 mm was made, while measuring the head over the notch. Calculate the percentage error in the discharge. Take C_d = 0.60.*

Solution. Given :

Length of notch, $L = 40$ cm

Discharge, $Q = 30$ lit/s $= 30000$ cm^3/s

Error in head, $dH = 1.5$ mm $= 0.15$ cm

$C_d = 0.60$

Let the height of water over rectangular notch = H

The discharge through a rectangular notch is given by (8.1)

or $$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$

or $$30000 = \frac{2}{3} \times 0.60 \times 40 \times \sqrt{2 \times 981} \times H^{3/2}$$

or $$H^{3/2} = \frac{3 \times 30000}{2 \times .60 \times 40 \times \sqrt{2 \times 981}} = 42.33$$

$\therefore$ $$H = (42.33)^{2/3} = 12.16 \text{ cm}$$

Using equation (8.6), we get

$$\frac{dQ}{Q} = \frac{3}{2}\frac{dH}{H} = \frac{3}{2} \times \frac{0.15}{12.16} = 0.0185 = \mathbf{1.85\%}. \textbf{ Ans.}$$

Problem 8.10 *A right-angled V-notch is used for measuring a discharge of 30 litres/s. An error of 1.5 mm was made while measuring the head over the notch. Calculate the percentage error in the discharge. Take C_d = 0.62.*

Solution. Given :

Angle of V-notch, $\theta = 90°$

Discharge, $Q = 30$ lit/s $= 30000$ cm^3/s

Error in head, $dH = 1.5$ mm $= 0.15$ cm

$C_d = 0.62$

Let the head over the V-notch $= H$

The discharge Q through a triangular notch is given by equation (8.2)

$$Q = \frac{8}{15} C_d \,.\, \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2}$$

or $$30000 = \frac{8}{15} \times 0.62 \times \tan\left(\frac{90°}{2}\right) \times \sqrt{2 \times 981} \times H^{5/2}$$

$$= \frac{8}{15} \times .62 \times 1 \times 44.29 \times H^{5/2}$$

$\therefore$ $$H^{5/2} = \frac{30000 \times 15}{8 \times .62 \times 44.29} = 2048.44$$

$\therefore$ $$H = (2048.44)^{2/5} = 21.11 \text{ cm}$$

Using equation (8.7), we get

$$\frac{dQ}{Q} = \frac{5}{2}\frac{dH}{H} = 2.5 \times \frac{0.15}{21.11} = 0.01776 = \mathbf{1.77\%}. \quad \textbf{Ans.}$$

Problem 8.11 *The head of water over a triangular notch of angle 60° is 50 cm and co-efficient of discharge is 0.62. The flow measured by it is to be within an accuracy of 1.5% up or down. Find the limiting values of the head.*

Solution. Given :

Angle of V-notch, $\theta = 60°$

Head of water, $H = 50$ cm

$C_d = 0.62$

$$\frac{dQ}{Q} = \pm 1.5\% = \pm 0.015$$

The discharge Q over a triangular notch is

$$Q = \frac{8}{15} C_d \sqrt{2g} \tan\frac{\theta}{2} H^{5/2}$$

$$= \frac{8}{15} \times 0.62 \times \sqrt{2 \times 981} \times \tan\frac{60°}{2} \times (50)^{5/2}$$

$$= 14.64 \times 0.5773 \times 17677.67 = 149405.86 \text{ cm}^3/\text{s}$$

Now applying equation (8.7), we get

$$\frac{dQ}{Q} = \frac{5}{2}\frac{dH}{H} \text{ or } \pm .015 = 2.5\frac{dH}{H} \text{ or } \frac{dH}{H} = \pm\frac{.015}{2.5}$$

$$\therefore \quad dH = \pm\frac{.015}{2.5} \times H = \pm\frac{.015}{2.5} \times 50 = \pm 0.3$$

$\therefore$ The limiting values of the head

$$= H \pm dH = 50 \pm 0.3 = 50.3 \text{ cm}, 49.7 \text{ cm}$$

$$= \mathbf{50.3 \text{ cm and } 49.7 \text{ cm. Ans.}}$$

▶ 8.9. (a) TIME REQUIRED TO EMPTY A RESERVOIR OR A TANK WITH A RECTANGULAR WEIR OR NOTCH

Consider a reservoir or tank of uniform cross-sectional area A. A rectangular weir or notch is provided in one of its sides.

Let L = Length of crest of the weir or notch

C_d = Co-efficient of discharge

H_1 = Initial height of liquid above the crest of notch

H_2 = Final height of liquid above the crest of notch

T = Time required in seconds to lower the height of liquid from H_1 to H_2.

Let at any instant, the height of liquid surface above the crest of weir or notch be h and in a small time dT, let the liquid surface falls by 'dh'. Then,

$$-Adh = Q \times dT$$

–ve sign is taken, as with the increase of T, h decreases.

But $$Q = \frac{2}{3} C_d \times L \times \sqrt{2g} \times h^{3/2}$$

$\therefore$ $$-Adh = \frac{2}{3} C_d \times L \times \sqrt{2g} \cdot h^{3/2} \times dT \text{ or } dT = \frac{-Adh}{\frac{2}{3} C_d \times L \times \sqrt{2g} \times h^{3/2}}$$

The total time T is obtained by integrating the above equation between the limits H_1 and H_2.

$$\int_0^T dT = \int_{H_1}^{H_2} \frac{-Adh}{\frac{2}{3} C_d \times L \times \sqrt{2g} \times h^{3/2}}$$

or $$T = \frac{-A}{\frac{2}{3} C_d \times L \times \sqrt{2g}} \int_{H_1}^{H_2} h^{-3/2}\, dh = \frac{-3A}{2\, C_d \times L \times \sqrt{2g}} \left[\frac{h^{-3/2+1}}{-\frac{3}{2}+1}\right]_{H_1}^{H_2}$$

$$= \frac{-3A}{2\, C_d \times L \times \sqrt{2g}} \left[\frac{h^{-1/2}}{-\frac{1}{2}}\right]_{H_1}^{H_2} = \frac{-3A}{2\, C_d \times L \times \sqrt{2g}} \left(-\frac{2}{1}\right) \left[\frac{1}{\sqrt{h}}\right]_{H_1}^{H_2}$$

$$= \frac{3A}{C_d \times L \times \sqrt{2g}} \left[\frac{1}{\sqrt{H_2}} - \frac{1}{\sqrt{H_1}}\right]. \quad \text{...(8.8)}$$

(b) TIME REQUIRED TO EMPTY A RESERVOIR OR A TANK WITH A TRIANGULAR WEIR OR NOTCH

Consider a reservoir or tank of uniform cross-sectional area A, having a triangular weir or notch in one of its sides.

Let θ = Angle of the notch

C_d = Co-efficient of discharge

H_1 = Initial height of liquid above the apex of notch

H_2 = Final height of liquid above the apex of notch

T = Time required in seconds, to lower the height from H_1 to H_2 above the apex of the notch.

Let at any instant, the height of liquid surface above the apex of weir or notch be h and in a small time dT, let the liquid surface falls by 'dh'. Then

$$-Adh = Q \times dT$$

–ve sign is taken, as with the increase of T, h decreases.

And Q for a triangular notch is

$$Q = \frac{8}{15} \times C_d \times \tan\frac{\theta}{2} \sqrt{2g} \times h^{5/2}$$

$\therefore$ $$-Adh = \frac{8}{15} \times C_d \times \tan\frac{\theta}{2} \times \sqrt{2g} \times h^{5/2} \times dT$$

$$\therefore \qquad dT = \frac{Adh}{\frac{8}{15} \times C_d \times \tan\frac{\theta}{2} \times \sqrt{2g} \times h^{5/2}}$$

The total time T is obtained by integrating the above equation between the limits H_1 and H_2.

$$\therefore \qquad \int_0^T dT = \int_{H_1}^{H_2} \frac{-Adh}{\frac{8}{15} C_d \tan\frac{\theta}{2} \sqrt{2g}\, h^{5/2}}$$

or
$$T = \frac{-A}{\frac{8}{15} C_d \times \tan\frac{\theta}{2} \times \sqrt{2g}} \int_{H_1}^{H_2} h^{-5/2}\, dh$$

$$= \frac{-15A}{8 \times C_d \times \tan\frac{\theta}{2} \times \sqrt{2g}} \left[\frac{h^{-3/2}}{-\frac{3}{2}}\right]_{H_1}^{H_2}$$

$$= \frac{-15A}{8 \times C_d \times \tan\frac{\theta}{2} \times \sqrt{2g}} \times \left(-\frac{2}{3}\right)\left[\frac{1}{h^{3/2}}\right]_{H_1}^{H_2}$$

$$= \frac{5A}{4 \times C_d \times \tan\frac{\theta}{2} \times \sqrt{2g}} \left[\frac{1}{{H_2}^{3/2}} - \frac{1}{{H_1}^{3/2}}\right]. \qquad \text{...(8.9)}$$

Problem 8.12 *Find the time required to lower the water level from 3 m to 2 m in a reservoir of dimension 80 m × 80 m, by a rectangular notch of length 1.5 m. Take C_d = 0.62.*

Solution. Given :

Initial height of water, $H_1 = 3$ m

Final height of water, $H_2 = 2$ m

Dimension of reservoir = 80 m × 80 m

or Area, $A = 80 \times 80 = 6400\ \text{m}^2$

Length of notch, $L = 1.5$ m, $C_d = 0.62$

Using the relation given by the equation (8.8)

$$T = \frac{3A}{C_d \times L \times \sqrt{2g}} \left[\frac{1}{\sqrt{H_2}} - \frac{1}{\sqrt{H_1}}\right]$$

$$= \frac{3 \times 6400}{0.62 \times 1.5 \times \sqrt{2 \times 9.81}} \left[\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{3}}\right]$$

$$= 4661.35\ [0.7071 - 0.5773] \text{ seconds}$$

$$= 605.04 \text{ seconds} = \textbf{10 min 5 sec. Ans.}$$

Problem 8.13 *If in problem 8.12, instead of a rectangular notch, a right-angled V-notch is used, find the time required. Take all other data same.*

Solution. Given :

Angle of notch, $\theta = 90°$

Initial height of water, $H_1 = 3$ m

Final height of water, $H_2 = 2$ m

Area of reservoir, $A = 80 \times 80 = 6400$ m^2

$C_d = 0.62$

Using the relation given by equation (8.9)

$$T = \frac{5A}{4 \times C_d \times \tan\frac{\theta}{2} \times \sqrt{2g}} \left[\frac{1}{H_2^{3/2}} - \frac{1}{H_1^{3/2}}\right]$$

$$= \frac{5 \times 6400}{4 \times .62 \times \tan\frac{90°}{2} \times \sqrt{2 \times 9.81}} \left[\frac{1}{2^{1.5}} - \frac{1}{3^{1.5}}\right] \qquad \left\{\because \tan\frac{90°}{2} = 1\right\}$$

$$= 2913.34 \times \left[\frac{1}{2.8284} - \frac{1}{5.1961}\right]$$

$$= 2913.34 \ [0.3535 - 0.1924] \text{ seconds}$$

$$= 469.33 \text{ seconds} = \textbf{7 min 49.33 sec. Ans.}$$

Problem 8.14 *A right-angled V-notch is inserted in the side of a tank of length 4 m and width 2.5 m. Initial height of water above the apex of the notch is 30 cm. Find the height of water above the apex if the time required to lower the head in tank from 30 cm to final height is 3 minutes. Take $C_d = 0.60$.*

Solution. Given :

Angle of notch, $\theta = 90°$

Area of tank, $A = \text{Length} \times \text{width} = 4 \times 2.5 = 10.0$ m^2

Initial height of water, $H_1 = 30$ cm $= 0.3$ m

Time, $T = 3$ min $= 3 \times 60 = 180$ seconds

$C_d = 0.60$

Let the final height of water above the apex of notch $= H_2$

Using the relation given by equation (8.9)

$$T = \frac{5A}{4 \times C_d \times \tan\frac{\theta}{2} \times \sqrt{2g}} \left[\frac{1}{H_2^{3/2}} - \frac{1}{H_1^{3/2}}\right]$$

$$180 = \frac{5 \times 10}{4 \times .60 \times \tan\left(\frac{90°}{2}\right) \times \sqrt{2 \times 9.81}} \left[\frac{1}{H_2^{3/2}} - \frac{1}{(0.3)^{3/2}}\right]$$

$$= \frac{50}{4 \times .60 \times 1 \times 4.429} \left[\frac{1}{H_2^{3/2}} - \frac{1}{(0.3)^{3/2}}\right]$$

or $$\frac{1}{H_2^{1.5}} - \frac{1}{0.3^{1.5}} = \frac{180 \times 4 \times 0.60 \times 4.429}{50} = 38.266.$$

or $$\frac{1}{H_2^{1.5}} - 6.0858 = 38.266$$

$$\therefore \quad \frac{1}{H_2^{1.5}} = 38.266 + 6.0858 = 44.35 \text{ or } H_2^{1.5} = \frac{1}{44.35} = 0.0225$$

$$\therefore \quad H_2 = (0.0225)^{1/1.5} = (0.0225)^{.6667} = 0.0822 \text{ m} = \mathbf{8.22 \text{ cm. Ans.}}$$

▶ 8.10 VELOCITY OF APPROACH

Velocity of approach is defined as the velocity with which the water approaches or reaches the weir or notch before it flows over it. Thus if V_a is the velocity of approach, then an additional head h_a equal to $\frac{V_a^2}{2g}$ due to velocity of approach, is acting on the water flowing over the notch. Then initial height of water over the notch becomes (H+ h_a) and final height becomes equal to h_a. Then all the formulae are changed taking into consideration of velocity of approach.

The velocity of approach, V_a is determined by finding the discharge over the notch or weir neglecting velocity of approach. Then dividing the discharge by the cross-sectional area of the channel on the upstream side of the weir or notch, the velocity of approach is obtained. Mathematically,

$$V_a = \frac{Q}{\text{Area of channel}}$$

This velocity of approach is used to find an additional head $\left(h_a = \frac{V_a^2}{2g}\right)$. Again the discharge is calculated and above process is repeated for more accurate discharge.

Discharge over a rectangular weir, with velocity of approach

$$= \frac{2}{3} \times C_d \times L \times \sqrt{2g}\ [(H_1 + h_a)^{3/2} - h_a^{3/2}] \qquad ...(8.10)$$

Problem 8.15 *Water is flowing in a rectangular channel of 1 m wide and 0.75 m deep. Find the discharge over a rectangular weir of crest length 60 cm, if the head of water over the crest of weir is 20 cm and water from channel flows over the weir. Take C_d = 0.62. Neglect end contractions. Take velocity of approach into consideration.*

Solution. Given :

Area of channel, A = Width × depth = 1.0 × 0.75 = 0.75 m^2

Length of weir, L = 60 cm = 0.6 m

Head of water, H_1 = 20 cm = 0.2 m

C_d = 0.62

Discharge over a rectangular weir without velocity of approach is given by

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H_1^{3/2}$$

$$= \frac{2}{3} \times 0.62 \times 0.6 \times \sqrt{2 \times 9.81} \times (0.2)^{3/2} \text{ m}^3/\text{s}$$

$= 1.098 \times 0.0894 = 0.0982 \text{ m}^3/\text{s}$

Velocity of approach, $V_a = \dfrac{Q}{A} = \dfrac{.0982}{0.75} = 0.1309 \text{ m/s}$

$\therefore$ Additional head, $h_a = \dfrac{V_a^2}{2g} = (.1309)^2/2 \times 9.81 = .0008733 \text{ m}$

Then discharge with velocity of approach is given by equation (8.10)

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g}\ [(H_1 + h_a)^{3/2} - h_a^{3/2}]$$

$$= \frac{2}{3} \times 0.62 \times 0.6 \times \sqrt{2 \times 9.81}\ [(0.2 + .00087)^{3/2} - (.00087)^{3/2}]$$

$$= 1.098\ [0.09002 - .00002566]$$

$$= 1.098 \times 0.09017 = \textbf{.09881 m}^3\textbf{/s. Ans.}$$

Problem 8.16 *Find the discharge over a rectangular weir of length 100 m. The head of water over the weir is 1.5 m. The velocity of approach is given as 0.5 m/s. Take $C_d = 0.60$.*

Solution. Given :

Length of weir, $L = 100 \text{ m}$

Head of water, $H_1 = 1.5 \text{ m}$

Velocity of approach, $V_a = 0.5 \text{ m/s}$

$C_d = 0.60$

$\therefore$ Additional head, $h_a = \dfrac{V_a^2}{2g} = \dfrac{0.5 \times 0.5}{2 \times 9.81} = 0.0127 \text{ m}$

The discharge, Q over a rectangular weir due to velocity of approach is given by equation (8.10)

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g}\ [(H_1 + h_a)^{3/2} - h_a^{3/2}]$$

$$= \frac{2}{3} \times 0.6 \times 100 \times \sqrt{2 \times 9.81}\ [(1.5 + .0127)^{3/2} - .0127^{3/2}]$$

$$= 177.16\ [\ 1.5127^{3/2} - .0127^{3/2}]$$

$$= 177.16\ [1.8605 - .00143] = \textbf{329.35 m}^3\textbf{/s. Ans.}$$

Problem 8.17 *A rectangular weir of crest length 50 cm is used to measure the rate of flow of water in a rectangular channel of 80 cm wide and 70 cm deep. Determine the discharge in the channel if the water level is 80 mm above the crest of weir. Take velocity of approach into consideration and value of $C_d = 0.62$.*

Solution. Given :

Length of weir, $L = 50 \text{ cm} = 0.5 \text{ m}$

Area of channel, $A = \text{Width} \times \text{depth} = 80 \text{ cm} \times 70 \text{ cm} = 0.80 \times 0.70 = 0.56 \text{ m}^2$

Head over weir, $H = 80 \text{ mm} = 0.08 \text{ m}$

$C_d = 0.62$

The discharge over a rectangular weir without velocity of approach is given by equation (8.1)

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$

$$= \frac{2}{3} \times 0.62 \times 0.5 \times \sqrt{2 \times 9.81} \times (0.08)^{3/2} \text{ m}^3/\text{s}$$

$$= 0.9153 \times .0226 = .0207 \text{ m}^3/\text{s}$$

Velocity of approach, $V_a = \frac{Q}{A} = \frac{.0207}{0.56} = .0369$ m/s

$\therefore$ Head due to V_a, $h_a = V_a^2/2g = \frac{(.0369)^2}{2 \times 9.81} = .0000697$ m

Discharge with velocity of approach is

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \ [(H_1 + h_a)^{3/2} - h_a^{3/2}]$$

$$= \frac{2}{3} \times 0.62 \times 0.5 \times \sqrt{2 \times 9.81} \ [(.08 + .0000697)^{3/2} - .0000697^{3/2}]$$

$$= 0.9153 \times [.0800697^{1.5} - .0000697^{1.5}]$$

$$= .9153 \ [.02265 - .000000582] = \mathbf{0.2073 \ m^3/s. \ Ans.}$$

Problem 8.18 *A suppressed rectangular weir is constructed across a channel of 0.77 m width with a head of 0.39 m and the crest 0.6 m above the bed of the channel. Estimate the discharge over it. Consider velocity of approach and assume $C_d = 0.623$.*

Solution. Given :

Width of channel, $b = 0.77$ m

Head over weir, $H = 0.39$ m

Height of crest from bed of channel = 0.6 m

$\therefore$ Depth of channel = 0.6 + 0.39 = 0.99

Value of $C_d = 0.623$

Suppressed weir means that the width of channel is equal to width of weir *i.e.*, there is no end contraction.

$\therefore$ Width of channel = Width of weir = 0.77 m

Now area of channel, A = Width of channel × Depth of channel

$= 0.77 \times 0.99$

The discharge over a rectangular weir without velocity of approach is given by equation (8.1).

$\therefore$ $$Q = \frac{2}{3} \times C_d \times b \times \sqrt{2g} \times H^{3/2}$$ ($\because$ Here $b = L$)

$$= \frac{2}{3} \times 0.623 \times 0.77 \times \sqrt{2 \times 9.81} \times 0.39^{3/2} = 0.345 \text{ m}^3/\text{s}$$

Now velocity of approach, $V_a = \frac{Q}{\text{Area of channel}} = \frac{0.345}{0.77 \times 0.99} = 0.4526$ m/s

Head due to velocity of approach,

$$h_a = \frac{V_a^2}{2g} = \frac{0.4526^2}{2 \times 9.81} = 0.0104 \text{ m}$$

Now the discharge with velocity of approach is given by,

$$Q = \frac{2}{3} \times C_d \times b \times \sqrt{2g}\ [(H + h_a)^{3/2} - h_a^{3/2}]$$

$$= \frac{2}{3} \times 0.623 \times 0.77 \times \sqrt{2 \times 9.81}\ [(0.39 + 0.0104)^{3/2} - (0.0104)^{3/2}]$$

$$= \frac{2}{3} \times 0.623 \times 0.77 \times 4.43\ [0.2533 - 0.00106]$$

$$= \mathbf{0.3573\ m^3/s.\ Ans.}$$

Problem 8.19 *A sharp crested rectangular weir of 1 m height extends across a rectangular channel of 3 m width. If the head of water over the weir is 0.45 m, calculate the discharge. Consider velocity of approach and assume $C_d = 0.623$.*

Solution. Given :

Width of channel, $b = 3$ m

Height of weir $= 1$ m

Head of water over weir, $H = 0.45$ m

$\therefore$ Depth of channel = Height of weir + Head of water over weir

$= 1 + 0.45 = 1.45$ m

Value of $C_d = 0.623$

The discharge over a rectangular weir without velocity of approach is given by equation (8.1) as

$$Q = \frac{2}{3} \times C_d \times b \times \sqrt{2g} \times H^{3/2}$$

$$= \frac{2}{3} \times 0.623 \times 3 \times \sqrt{2 \times 9.81} \times 0.45^{3/2} = 1.665 \text{ m}^3/\text{s}$$

Now velocity of approach is given by

$$V_a = \frac{Q}{\text{Area of channel}}$$

$$= \frac{1.665}{\text{Width of channel} \times \text{Depth of channel}} = \frac{1.665}{3 \times 1.45} = 0.382 \text{ m/s}$$

Head due to velocity of approach is given by,

$$h_a = \frac{V_a^2}{2g} = \frac{0.382^2}{2 \times 9.81} = 0.0074 \text{ m}$$

Now the discharge with velocity of approach is given by,

$$Q = \frac{2}{3} \times C_d \times b \times \sqrt{2g}\ [(H + h_a)^{3/2} - (h_a)^{3/2}]$$

$$= \frac{2}{3} \times 0.623 \times 3 \times \sqrt{2 \times 9.81}\ [(0.45 + 0.0074)^{3/2} - (0.0074)^{3/2}]$$

$$= \mathbf{1.703\ m^3/s.\ Ans.}$$

▶ 8.11 EMPIRICAL FORMULAE FOR DISCHARGE OVER RECTANGULAR WEIR

The discharge over a rectangular weir is given by

$$Q = \frac{2}{3} C_d \sqrt{2g} \times L \times [H^{3/2}] \text{ without velocity of approach} \quad ...(i)$$

$$= \frac{2}{3} C_d \sqrt{2g} \times L \times [(H + h_a)^{3/2} - h_a^{3/2}] \text{ with velocity of approach} \quad ...(ii)$$

Equations (*i*) and (*ii*) are applicable to the weir or notch for which the crest length is equal to the width of the channel. This type of weir is called *Suppressed weir*. But if the weir is not suppressed, the effect of end contraction will be taken into account.

(*a*) **Francis's Formula.** Francis on the basis of his experiments established that end contraction decreases the effective length of the crest of weir and hence decreases the discharge. Each end contraction reduces the crest length by $0.1 \times H$, where H is the head over the weir. For a rectangular weir there are two end contractions only and hence effective length

$$L = (L - 0.2\ H)$$

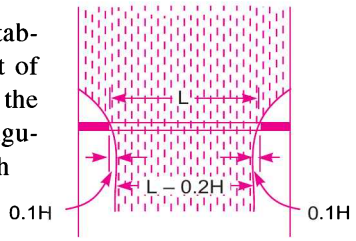


Fig. 8.8

and $$Q = \frac{2}{3} \times C_d \times [L - 0.2 \times H] \times \sqrt{2g}\ H^{3/2}$$

If $C_d = 0.623$, $g = 9.81$ m/s^2, then

$$Q = \frac{2}{3} \times .623 \times \sqrt{2 \times 9.81} \times [L - 0.2 \times H] \times H^{3/2}$$

$$= 1.84\ [L - 0.2 \times H] H^{3/2} \quad ...(8.11)$$

If end contractions are suppressed, then

$$H = 1.84\ LH^{3/2} \quad ...(8.12)$$

If velocity of approach is considered, then

$$Q = 1.84\ L\ [(H + h_a)^{3/2} - h_a^{3/2}] \quad ...(8.13)$$

(*b*) **Bazin's Formula.** On the basis of results of a series of experiments, Bazin's proposed the following formula for the discharge over a rectangular weir as

$$Q = m \times L \times \sqrt{2g} \times H^{3/2} \quad ...(8.14)$$

where $m = \frac{2}{3} \times C_d = 0.405 + \frac{.003}{H}$

H = height of water over the weir

If velocity of approach is considered, then

$$Q = m_1 \times L \times \sqrt{2g}\ [(H + h_a)^{3/2}] \quad ...(8.15)$$

where $m_1 = 0.405 + \frac{.003}{(H + h_a)}$.

Problem 8.20 *The head of water over a rectangular weir is 40 cm. The length of the crest of the weir with end contraction suppressed is 1.5 m. Find the discharge using the following formulae : (i) Francis's Formula and (ii) Bazin's Formula.*

Solution. Given :

Head of water, $H = 40 \text{ cm} = 0.40 \text{ m}$

Length of weir, $L = 1.5 \text{ m}$

(*i*) Francis's Formula for end contraction suppressed is given by equation (8.12).

$$Q = 1.84\ L \times H^{3/2} = 1.84 \times 1.5 \times (.40)^{3/2}$$
$$= 0.6982 \text{ m}^3/\text{s}$$

(*ii*) Bazin's Formula is given by equation (8.14)

$$Q = m \times L \times \sqrt{2g} \times H^{3/2}$$

where $m = 0.405 + \dfrac{.003}{H} = 0.405 + \dfrac{.003}{.40} = 0.4125$

$\therefore$ $$Q = .4125 \times 1.5 \times \sqrt{2 \times 9.81} \times (.4)^{3/2}$$
$$= \mathbf{0.6932\ m^3/s.\ Ans.}$$

Problem 8.21 *A weir 36 metres long is divided into 12 equal bays by vertical posts, each 60 cm wide. Determine the discharge over the weir if the head over the crest is 1.20 m and velocity of approach is 2 metres per second.*

Solution. Given :

Length of weir, $L_1 = 36 \text{ m}$

Number of bays, $= 12$

For 12 bays, no. of vertical post $= 11$

Width of each post $= 60 \text{ cm} = 0.6 \text{ m}$

$\therefore$ Effective length, $L = L_1 - 11 \times 0.6 = 36 - 6.6 = 29.4 \text{ m}$

Head on weir, $H = 1.20 \text{ m}$

Velocity of approach, $V_a = 2 \text{ m/s}$

$\therefore$ Head due to V_a, $h_a = \dfrac{V_a^2}{2g} = \dfrac{2^2}{2 \times 9.81} = 0.2038 \text{ m}$

Number of end contraction, $n = 2 \times 12$ {Each bay has two end contractions}

$= 24$

$\therefore$ Discharge by Francis Formula with end contraction and velocity of approach is

$$Q = 1.84\ [L - 0.1 \times n(H + h_a)][(H + h_a)^{3/2} - h_a^{3/2}]$$
$$= 1.84[29.4 - 0.1 \times 24(1.20 + .2038)] \times [(1.2 + .2038)^{1.5} - .2038^{1.5}]$$
$$= 1.84[29.4 - 3.369][1.663 - .092]$$
$$= \mathbf{75.246\ m^3/s.\ Ans.}$$

Problem 8.22 *A discharge of 2000 m^3/s is to pass over a rectangular weir. The weir is divided into a number of openings each of span 10 m. If the velocity of approach is 4 m/s, find the number of openings needed in order the head of water over the crest is not to exceed 2 m.*

Solution. Given :

Total discharge, $Q = 2000 \text{ m}^3/\text{s}$

Length of each opening, $L = 10$

Velocity of approach, $V_a = 4$ m/s
Head over weir, $H = 2$ m
Let number of openings $= N$
Head due to velocity of approach,

$$h_a = \frac{V_a^2}{2g} = \frac{4 \times 4}{2 \times 9.81} = 0.8155 \text{ m}$$

For each opening, number of end contractions are two. Hence discharge for each opening considering velocity of approach is given by Francis Formula

i.e.,
$$Q = 1.84[L - 0.1 \times 2 \times (H + h_a)][(H + h_a)^{3/2} - h_a^{3/2}]$$
$$= 1.84[10.0 - 0.2 \times (2 + .8155)][2.8155^{1.5} - .8155^{1.5}]$$
$$= 17.363[4.7242 - 0.7364] = 69.24 \text{ m}^3/\text{s}$$

$\therefore$ Number of opening $= \dfrac{\text{Total discharge}}{\text{Discharge for one opening}} = \dfrac{2000}{69.24}$

$= 28.88$ (say 29) = **29. Ans.**

▶ 8.12 CIPOLLETTI WEIR OR NOTCH

Cipolletti weir is a trapezoidal weir, which has side slopes of 1 horizontal to 4 vertical as shown in Fig. 8.9. Thus in ΔABC,

$$\tan \frac{\theta}{2} = \frac{AB}{BC} = \frac{H/4}{H} = \frac{1}{4}$$

$\therefore$
$$\frac{\theta}{2} = \tan \frac{1}{4} = 14° 2'.$$

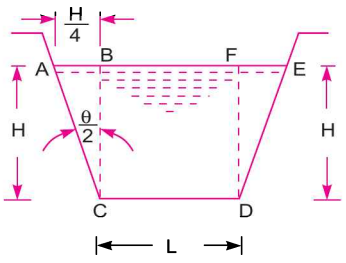


Fig. 8.9 *The cipolletti weir.*

By giving this slope to the sides, an increase in discharge through the triangular portions *ABC* and *DEF* of the weir is obtained. If this slope is not provided the weir would be a rectangular one, and due to end contraction, the discharge would decrease. Thus in case of cipolletti weir, the factor of end contraction is not required which is shown below.

The discharge through a rectangular weir with two end contractions is

$$Q = \frac{2}{3} \times C_d \times (L - 0.2\,H) \sqrt{2g} \times H^{3/2}$$
$$= \frac{2}{3} \times C_d \times L \times \sqrt{2g}\ H^{3/2} - \frac{2}{15} \times C_d \times \sqrt{2g} \times H^{5/2}$$

Thus due to end contraction, the discharge decreases by $\dfrac{2}{15} \times C_d \times \sqrt{2g} \times H^{5/2}$. This decrease in discharge can be compensated by giving such a slope to the sides that the discharge through two triangular portions is equal to $\dfrac{2}{15} \times C_d \times \sqrt{2g} \times H^{5/2}$. Let the slope is given by $\theta/2$. The discharge through a *V*-notch of angle θ is given by

$$= \frac{8}{15} \times C_d \times \sqrt{2g} \times \tan \frac{\theta}{2} H^{5/2}$$

Thus $$\frac{8}{15} \times C_d \times \sqrt{2g} \times \tan \frac{\theta}{2} H^{5/2} = \frac{2}{15} \times C_d \times \sqrt{2g} \times H^{5/2}$$

$\therefore$ $$\tan \frac{\theta}{2} = \frac{2}{15} \times \frac{15}{8} = \frac{1}{4} \quad \text{or} \quad \theta/2 = \tan^{-1} \frac{1}{4} = 14° 2'.$$

Thus discharge through cipolletti weir is

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g}\ H^{3/2} \qquad ...(8.16)$$

If velocity of approach, V_a is to be taken into consideration,

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g}\ [(H + h_a)^{3/2} - h_a^{3/2}] \qquad ...(8.17)$$

Problem 8.23 *Find the discharge over a cipolletti weir of length 2.0 m when the head over the weir is 1 m. Take C_d = 0.62.*

Solution. Given :

Length of weir, L = 20 m

Head over weir, H = 1.0 m

C_d = 0.62

Using equation (8.16), the discharge is given as

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$

$$= \frac{2}{3} \times 0.62 \times 2.0 \times \sqrt{2 \times 9.81} \times (1)^{3/2} = \mathbf{3.661\ m^3/s.\ Ans.}$$

Problem 8.24 *A cipolletti weir of crest length 60 cm discharges water. The head of water over the weir is 360 mm. Find the discharge over the weir if the channel is 80 cm wide and 50 cm deep. Take C_d = 0.60.*

Solution. Given :

C_d = 0.60

Length of weir, L = 60 cm = 0.60 m

Head of water, H = 360 mm = 0.36 m

Channel width = 80 cm = 0.80 m

Channel depth = 50 cm = 0.50 m

A = cross-sectional area of channel = $0.8 \times 0.5 = 0.4\ m^2$

To find velocity of approach, first determine discharge over the weir as

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$

The velocity of approach, $V_a = \frac{Q}{A}$

$\therefore$ $$Q = \frac{2}{3} \times 0.60 \times 0.60 \times \sqrt{2 \times 9.81} \times (0.36)^{3/2}\ m^3/s = 0.2296\ m^3/s$$

$\therefore$ $$V_a = \frac{.2296}{0.40} = 0.574\ m/s$$

Head due to velocity of approach,

$$h_a = V_a^2/2g = \frac{(0.574)^2}{2 \times 9.81} = 0.0168 \text{ m}$$

Thus the discharge is given by equation (8.17) as

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g}\ [(H + h_a)^{1.5} - h_a^{1.5}]$$

$$= \frac{2}{3} \times 0.60 \times .6 \times \sqrt{2 \times 9.81}\ [(.36 + .0168)^{1.5} - (.0168)^{1.5}]$$

$$= 1.06296 \times [.2313 - .002177] = \mathbf{0.2435\ m^3/s.\ Ans.}$$

▶ 8.13 DISCHARGE OVER A BROAD-CRESTED WEIR

A weir having a wide crest is known as broad-crested weir.

Let H = height of water above the crest

L = length of the crest

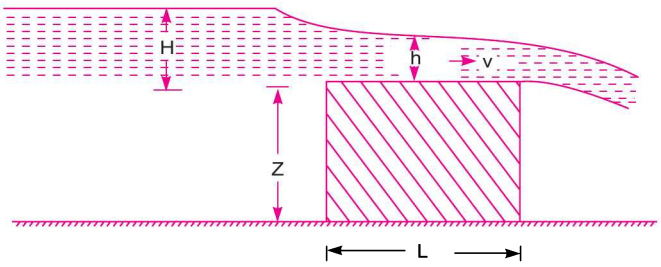


Fig. 8.10 *Broad-crested weir.*

If $2L > H$, the weir is called broad-crested weir

If $2L < H$, the weir is called a narrow-crested weir

Fig. 8.10 shows a broad-crested weir.

Let h = head of water at the middle of weir which is constant

v = velocity of flow over the weir

Applying Bernoulli's equation to the still water surface on the upstream side and running water at the end of weir,

$$0 + 0 + H = 0 + \frac{v^2}{2g} + h$$

$$\therefore \quad \frac{v^2}{2g} = H - h$$

$$\therefore \quad v = \sqrt{2g\,(H - h)}$$

$\therefore$ The discharge over weir $Q = C_d \times$ Area of flow $\times$ Velocity

$$= C_d \times L \times h \times \sqrt{2g\,(H - h)}$$

$$= C_d \times L \times \sqrt{2g\left(Hh^2 - h^3\right)} \qquad \text{...(8.18)}$$

The discharge will be maximum, if $(Hh^2 - h^3)$ is maximum

or $$\frac{d}{dh}(Hh^2 - h^3) = 0 \text{ or } 2h \times H - 3h^2 = 0 \text{ or } 2H = 3h$$

$$\therefore \quad h = \frac{2}{3}H$$

Q_{max} will be obtained by substituting this value of h in equation (8.18) as

$$Q_{max} = C_d \times L \times \sqrt{2g\left[H \times \left(\frac{2}{3}H\right)^2 - \left(\frac{2}{3}H\right)^3\right]}$$

$$= C_d \times L \times \sqrt{2g}\sqrt{H \times \frac{4}{9} \times H^2 - \frac{8}{27}H^3}$$

$$= C_d \times L \times \sqrt{2g}\sqrt{\frac{4}{9}H^3 - \frac{8}{27}H^3} = C_d \times L \times \sqrt{2g}\sqrt{\frac{(12-8)H^3}{27}}$$

$$= C_d \times L \times \sqrt{2g}\sqrt{\frac{4}{27}H^3} = C_d \times L \times \sqrt{2g} \times 0.3849 \times H^{3/2}$$

$$= .3849 \times \sqrt{2 \times 9.81} \times C_d \times L \times H^{3/2} = 1.7047 \times C_d \times L \times H^{3/2}$$

$$= 1.705 \times C_d \times L \times H^{3/2}. \qquad ...(8.19)$$

▶ 8.14 DISCHARGE OVER A NARROW-CRESTED WEIR

For a narrow-crested weir, $2L < H$. It is similar to a rectangular weir or notch hence, Q is given by

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2} \qquad ...(8.20)$$

▶ 8.15 DISCHARGE OVER AN OGEE WEIR

Fig. 8.11 shows an Ogee weir, in which the crest of the weir rises upto maximum height of $0.115 \times H$ (where H is the height of water above inlet of the weir) and then falls as shown in Fig. 8.11. The discharge for an Ogee weir is the same as that of a rectangular weir, and it is given by

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2} \qquad ...(8.21)$$

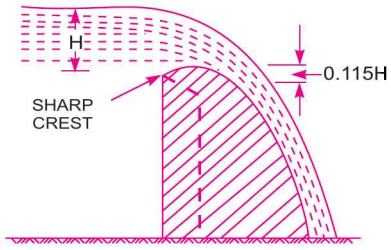


Fig. 8.11 *An Ogee weir.*

▶ 8.16 DISCHARGE OVER SUB-MERGED OR DROWNED WEIR

When the water level on the downstream side of a weir is above the crest of the weir, then the weir is called to be a sub-merged or drowned weir. Fig. 8.12 shows a sub-merged weir. The total discharge, over the weir is obtained by dividing the weir into two parts. The portion between upstream and downstream water surface may be treated as free weir and portion between downstream water surface and crest of weir as a drowned weir.

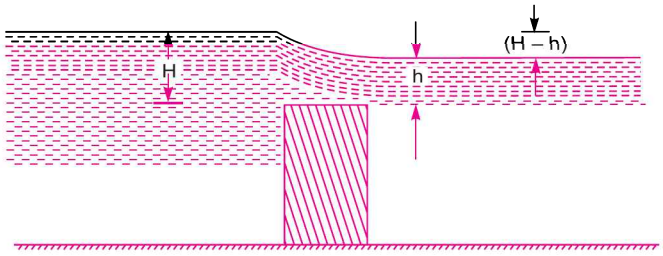


Fig. 8.12 *Sub-merged weir.*

Let H = height of water on the upstream side of the weir
h = height of water on the downstream side of the weir

Then Q_1 = discharge over upper portion

$$= \frac{2}{3} \times C_{d_1} \times L \times \sqrt{2g}\ [H - h]^{3/2}$$

Q_2 = discharge through drowned portion

$$= C_{d_2} \times \text{Area of flow} \times \text{Velocity of flow}$$

$$= C_{d_2} \times L \times h \times \sqrt{2g(H - h)}$$

$\therefore$ Total discharge, $Q = Q_1 + Q_2$

$$= \frac{2}{3} C_{d_1} \times L \times \sqrt{2g}\ [H - h]^{3/2} + C_{d_2} \times L \times h \times \sqrt{2g(H - h)} \quad ...(8.22)$$

Problem 8.25 *(a) A broad-crested weir of 50 m length, has 50 cm height of water above its crest. Find the maximum discharge. Take C_d = 0.60. Neglect velocity of approach. (b) If the velocity of approach is to be taken into consideration, find the maximum discharge when the channel has a cross-sectional area of 50 m^2 on the upstream side.*

Solution. Given :

Length of weir, $L = 50$ m

Head of water, $H = 50$ cm $= 0.5$ m

$C_d = 0.60$

(*i*) **Neglecting velocity of approach.** Maximum discharge is given by equation (8.19) as

$$Q_{max} = 1.705 \times C_d \times L \times H^{3/2}$$

$$= 1.705 \times 0.60 \times 50 \times (.5)^{3/2} = \mathbf{18.084\ m^3/s.\ Ans.}$$

(*ii*) **Taking velocity of approach into consideration**

Area of channel, $A = 50\ m^2$

Velocity of approach, $V_a = \frac{Q}{A} = \frac{18.084}{50} = 0.36$ m/s

$\therefore$ Head due to V_a, $h_a = \frac{V_a^2}{2g} = \frac{0.36 \times .36}{2 \times 9.81} = .0066$ m

Maximum discharge, Q_{max} is given by

$$Q_{max} = 1.705 \times C_d \times L \times [(H + h_a)^{3/2} - h_a^{3/2}]$$

$$= 1.705 \times 0.6 \times 50 \times [(.50 + .0066)^{1.5} - (.0066)^{1.5}]$$

$$= 51.15[0.3605 - .000536] = \mathbf{18.412\ m^3/s.\ Ans.}$$

Problem 8.26 *An Ogee weir 5 metres long has a head of 40 cm of water. If C_d = 0.6, find the discharge over the weir.*

Solution. Given :

Length of weir, $L = 5$ m

Head of water, $H = 40$ cm $= 0.40$ m

$C_d = 0.6$

Discharge over Ogee weir is given by equation (8.21) as

$$Q = \frac{2}{3} \times C_d \times L \times \sqrt{2g} \times H^{3/2}$$

$$= \frac{2}{3} \times 0.60 \times 5.0 \times \sqrt{2 \times 9.81} \times (0.4)^{3/2} = \mathbf{2.2409\ m^3/s.\ Ans.}$$

Problem 8.27 *The heights of water on the upstream and downstream side of a sub-merged weir of 3 m length are 20 cm and 10 cm respectively. If C_d for free and drowned portions are 0.6 and 0.8 respectively, find the discharge over the weir.*

Solution. Given :

Height of water on upstream side, $H = 20$ cm $= 0.20$ m

Height of water on downstream side, $h = 10$ cm $= 0.10$ m

Length of weir, $L = 3$ m

$C_{d_1} = 0.6$

$C_{d_2} = 0.8$

Total discharge Q is the sum of discharge through free portion and discharge through the drowned portion. This is given by equation (8.22) as

$$Q = \frac{2}{3} \times C_{d_1} \times L \times \sqrt{2g}\ [H - h]^{3/2} + C_{d_2} \times L \times h \times \sqrt{2g\,(H - h)}$$

$$= \frac{2}{3} \times 0.6 \times 3 \times \sqrt{2 \times 9.81}\ [.20 - .10]^{1.5} + 0.8 \times 3 \times .10 \times \sqrt{2 \times 9.81 (.2 - .1)}$$

$$= 0.168 + 0.336 = \mathbf{0.504\ m^3/s.\ Ans.}$$

HIGHLIGHTS

1. A notch is a device used for measuring the rate of flow of a liquid through a small channel. A weir is a concrete or masonary structure placed in the open channel over which the flow occurs.
2. The discharge through a rectangular notch or weir is given by

$$Q = \frac{2}{3} C_d \times L \times H^{3/2}$$

where C_d = Co-efficient of discharge,
L = Length of notch or weir,
H = Head of water over the notch or weir.

3. The discharge over a triangular notch or weir is given by

$$Q = \frac{8}{15} C_d \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2}$$

where θ = total angle of triangular notch.

4. The discharge through a trapezoidal notch or weir is equal to the sum of discharge through a rectangular notch and the discharge through a triangular notch. It is given as

$$Q = \frac{2}{3} C_{d_1} \times L \times \sqrt{2g} \times H^{3/2} + \frac{8}{15} C_{d_2} \times \tan \frac{\theta}{2} \times \sqrt{2g} \times H^{5/2}$$

where C_{d_1} = co-efficient of discharge for rectangular notch,

C_{d_2} = co-efficient of discharge for triangular notch,

$\theta/2$ = slope of the side of trapezoidal notch.

5. The error in discharge due to the error in the measurement of head over a rectangular and triangular notch or weir is given by

$$\frac{dQ}{Q} = \frac{3}{2}\frac{dH}{H} \quad \text{... For a rectangular weir or notch}$$

$$= \frac{5}{2}\frac{dH}{H} \quad \text{... For a triangular weir or notch}$$

where Q = discharge through rectangular or triangular notch or weir

H = head over the notch or weir.

6. The time required to empty a reservoir or a tank by a rectangular or a triangular notch is given by

$$H = \frac{3A}{C_d L \sqrt{2g}} \left[\frac{1}{\sqrt{H_2}} - \frac{1}{\sqrt{H_1}} \right] \quad \text{... By a rectangular notch}$$

$$= \frac{5A}{4C_d \tan \frac{\theta}{2} \times \sqrt{2g}} \left[\frac{1}{H_2^{3/2}} - \frac{1}{H_1^{3/2}} \right] \quad \text{... By a triangular notch}$$

where A = cross-sectional area of a tank or a reservoir

H_1 = initial height of liquid above the crest or apex of notch

H_2 = final height of liquid above the crest or apex of notch.

7. The velocity with which the water approaches the weir or notch is called the velocity of approach. It is denoted by V_a and is given by

$$V_a = \frac{\text{Discharge over the notch or weir}}{\text{Cross - sectional area of channel}}.$$

8. The head due to velocity of approach is given by $h_a = \frac{V_a^2}{2g}$.

9. Discharge over a rectangular weir, with velocity of approach,

$$Q = \frac{2}{3} C_d L \sqrt{2g} \left[(H_1 + h_a)^{3/2} - h_a^{3/2} \right].$$

10. Francis's Formula for a rectangular weir is given by

$$Q = 1.84[L - 0.2\ H]\ H^{3/2} \quad \text{... For two end contractions}$$

$$= 1.84\ L\ H^{3/2} \quad \text{... If end contractions are suppressed}$$

$$= 1.84\ L\ [(H + h_a)^{3/2} - h_a^{3/2}] \quad \text{... If velocity of approach is considered}$$

where L = length of weir,

H = height of water above the crest of the weir,

h_a = head due to velocity of approach.

11. Bazin's Formula for discharge over a rectangular weir,

$$Q = m\,L\sqrt{2g}H^{3/2} \quad \text{... without velocity of approach}$$

$$= m\,L\sqrt{2g}\,[(H + h_a)^{3/2}] \quad \text{... with velocity of approach}$$

where $m = \frac{2}{3}\,C_d = 0.405 + \frac{.003}{H}$ ... without velocity of approach

$$= 0.405 + \frac{.003}{(H + h_a)} \quad \text{... with velocity of approach.}$$

12. *A* trapezoidal weir, with side slope or 1 horizontal to 4 vertical, is called Cipolletti weir. The discharge through Cipolletti weir is given by

$$Q = \frac{2}{3}\,C_d \times L \times \sqrt{2g}\;H^{3/2} \quad \text{... without velocity of approach}$$

$$= \frac{2}{3}\,C_d \times L \times \sqrt{2g}\;[(H + h_a)^{3/2} - h_a^{3/2}] \quad \text{... with velocity of approach.}$$

13. The discharge over a broad-crested weir is given by,

$$Q = C_d L\sqrt{2g\left(Hh^2 - h^3\right)}$$

where H = height of water above the crest
h = head of water at the middle of the weir which is constant
L = length of the weir.

14. The condition for maximum discharge over a broad-crested weir is $h = \frac{2}{3}\,H$

and maximum discharge is given by $Q_{max} = 1.705\,C_d\,L\,H^{3/2}$.

15. The discharge over an Ogee weir is given by $Q = \frac{2}{3}\,C_d L \times \sqrt{2g} \times H^{3/2}$.

16. The discharge over sub-merged or drowned weir is given by

Q = discharge over upper portion + discharge through downed portion

$$= \frac{2}{3}C_{d_1}\,L \times \sqrt{2g}\;(H - h)^{3/2} + C_{d_2}\,Lh \times \sqrt{2g\,(H - h)}$$

where H = height of water on the upstream side of the weir,
h = height of water on the downstream side of the weir.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define the terms : notch, weir, nappe and crest.
2. How are the weirs and notches classified ?
3. Find an expression for the discharge over a rectangular weir in terms of head of water over the crest of the weir.
4. Prove that the discharge through a triangular notch or weir is given by

$$Q = \frac{8}{15}\,C_d \times \tan\frac{\theta}{2} \times \sqrt{2g}\;H^{3/2}$$

where H = head of water over the notch or weir
θ = angle of notch or weir.

5. What are the advantages of triangular notch or weir over rectangular notch ?
6. Prove that the error in discharge due to the error in the measurement of head over a rectangular notch is given by

$$\frac{dQ}{Q} = \frac{3}{2}\frac{dH}{H}$$

where Q = discharge through rectangular notch
and H = head over the rectangular notch.

7. Find an expression for the time required to empty a tank of area of cross-section A, with a rectangular notch.
8. What do you understand by 'Velocity of Approach' ? Find an expression for the discharge over a rectangular weir with velocity of approach.
9. Define 'end contraction' of a weir. What is the effect of end contraction on the discharge through a weir ?
10. What is a Cipolletti Weir ? Prove that the discharge through Cipolletti weir is given by

$$Q = \frac{2}{3} C_d L \sqrt{2g}\ H^{3/2}$$

where L = length of weir, and H = head of water over weir.

11. Differentiate between Broad-crested weir and Narrow-crested weir. Find the condition for maximum discharge over a Broad-crested weir and hence derive an expression for maximum discharge over a broad-crested weir.
12. What do you mean by a drowned weir ? How will you determine the discharge for the downed weir ?
13. Discuss 'end contraction' of a weir.
14. State the different devices that can be used to measure the discharge through a pipe also through an open channel. Describe one of such devices with a neat sketch and explain how one can obtain the actual discharge with its help.
15. What is the difference between a notch and a weir ?
16. Define velocity of approach. How does the velocity of approach affect the discharge over a weir ?

(B) NUMERICAL PROBLEMS

1. Find the discharge of water flowing over rectangular notch of 3 m length when the constant head of water over the notch is 40 cm. Take $C_d = 0.6$. [**Ans.** 1.344 m^3/s]
2. Determine the height of a rectangular weir of length 5 m to be built across a rectangular channel. The maximum depth of water on the upstream side of the weir is 1.5 m and discharge is 1.5 m^3 per second. Take $C_d = 0.6$ and neglect end contractions. [**Ans.** 1.194 m]
3. Find the discharge over a triangular notch of angle 60° when the head over the triangular notch is 0.20 m. Take $C_d = 0.6$. [**Ans.** 0.0164 m^3/s]
4. A rectangular channel 1.5 m wide has a discharge of 200 litres per second, which is measured by a right-angled V-notch weir. Find the position of the apex of the notch from the bed of the channel if maximum depth of water is not be exceed 1 m. Take $C_d = 0.62$. [**Ans.** .549 m]
5. Find the discharge through a trapezoidal notch which is 1.2 m wide at the top and 0.50 m at the bottom and is 40 cm in height. The head of water on the notch is 30 cm. Assume C_d for rectangular portion as 0.62 while for triangular portion = 0.60. [**Ans.** 0.22 m^3/s]
6. A rectangular notch 50 cm long is used for measuring a discharge of 40 litres per second. An error of 2 mm was made in measuring the head over the notch. Calculate the percentage error in the discharge. Take $C_d = 0.6$. [**Ans.** 2.37%]
7. A right-angled V-notch is used for measuring a discharge of 30 litres/s. An error of 2 mm was made in measuring the head over the notch. Calculate the percentage error in the discharge. Take $C_d = 0.62$. [**Ans.** 2.37%]

8. Find the time required to lower the water level from 3 m to 1.5 m in a reservoir of dimension 70 m × 70 m, by a rectangular notch of length 2.0 m. Take $C_d = 0.60$. [**Ans.** 11 min 1 s]

9. If in the problem 8, instead of a rectangular notch, a right angled *V*-notch is used, find the time required. Take all other data same. [**Ans.** 13 min 31 s]

10. Water is flowing in a rectangular channel of 1.2 m wide and 0.8 m deep. Find the discharge over a rectangular weir of crest length 70 cm if the head of water over the crest of weir is 25 cm and water from channel flows over the weir. Take $C_d = 0.60$. Neglect end contractions but consider velocity of approach. [**Ans.** 0.1557 m^3/s]

11. Find the discharge over a rectangular weir of length 80 m. The head of water over the weir is 1.2 m. The velocity of approach is given as 1.5 m/s. Take $C_d = 3.6$. [**Ans.** 208.11 m^3/s]

12. The head of water over a rectangular weir is 50 cm. The length of the crest of the weir with end contraction suppressed is 1.4 m. Find the discharge using following formulae : (*i*) Francis's Formula and (*ii*) Bazin's Formula. [**Ans.** (*i*) 0.91 m^3/s, (*ii*) .901 m^3/s]

13. A discharge of 1500 m^3/s is to pass over a rectangular weir. The weir is divided into a number of openings each of span 7.5 m. If the velocity of approach is 3 m/s, find the number of openings needed in order the head of water over the crest is not to exceed 1.8. [**Ans.** 37.5 say 38]

14. Find the discharge over a cipolletti weir of length 1.8 m when the head over the weir is 1.2 m. Take $C_d = 0.62$ [**Ans.** 4.331 m^3/s]

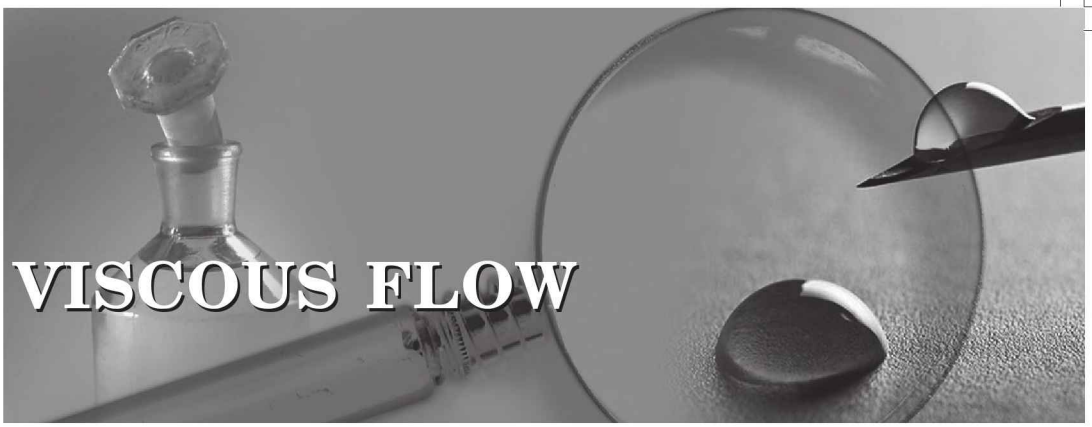
15. (*a*) A broad-crested weir of length 40 m, has 400 mm height of water above its crest. Find the maximum discharge. Take $C_d = 0.6$. Neglect velocity of approach. [**Ans.** 10.352 m^3/s]

(*b*) If the velocity of approach is to be taken into consideration, find the maximum discharge when the channel has a cross-sectional area of 40 m^2 on the upstream side. [**Ans.** 10.475 m^3/s]

16. An Ogee weir 4 m long has a head of 500 mm of water. If $C_d = 0.6$, find the discharge over the weir. [**Ans.** 2.505 m^3/s]

17. The heights of water on the upstream and downstream side of a sub-merged weir of length 3.5 m are 300 mm and 150 mm respectively. If C_d for free and drowned portion are 0.6 and 0.8 respectively, find the discharge over the weir. [**Ans.** 1.0807 m^3/s]

9 CHAPTER



VISCOUS FLOW

▶ 9.1 INTRODUCTION

This chapter deals with the flow of fluids which are viscous and flowing at very low velocity. At low velocity the fluid moves in layers. Each layer of fluid slides over the adjacent layer. Due to relative velocity between two layers the velocity gradient $\frac{du}{dy}$ exists and hence a shear stress $\tau = \mu \frac{du}{dy}$ acts on the layers.

The following cases will be considered in this chapter :

1. Flow of viscous fluid through circular pipe.
2. Flow of viscous fluid between two parallel plates.
3. Kinetic energy correction and momentum correction factors.
4. Power absorbed in viscous flow through
   (*a*) Journal bearings, (*b*) Foot-step bearings, and (*c*) Collar bearings.

▶ 9.2 FLOW OF VISCOUS FLUID THROUGH CIRCULAR PIPE

For the flow of viscous fluid through circular pipe, the velocity distribution across a section, the ratio of maximum velocity to average velocity, the shear stress distribution and drop of pressure for a given length is to be determined. The flow through the circular pipe will be viscous or laminar, if the Reynolds number (R_e*) is less than 2000. The expression for Reynold number is given by

$$R_e = \frac{\rho V D}{\mu}$$

where ρ = Density of fluid flowing through pipe
V = Average velocity of fluid
D = Diameter of pipe and
μ = Viscosity of fluid.

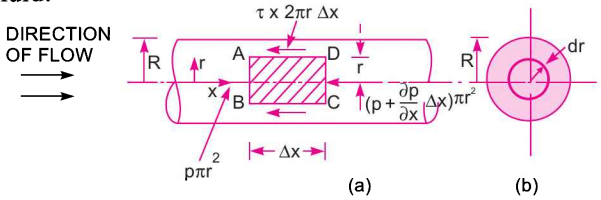


Fig. 9.1 *Viscous flow through a pipe.*

* For derivation, please refer to Art. 12.8.1.

Consider a horizontal pipe of radius R. The viscous fluid is flowing from left to right in the pipe as shown in Fig. 9.1 (a). Consider a fluid element of radius r, sliding in a cylindrical fluid element of radius $(r + dr)$. Let the length of fluid element be Δx. If 'p' is the intensity of pressure on the face AB, then the intensity of pressure on face CD will be $\left(p + \frac{\partial p}{\partial x}\Delta x\right)$. Then the forces acting on the fluid element are :

1. The pressure force, $p \times \pi r^2$ on face AB.
2. The pressure force, $\left(p + \frac{\partial p}{\partial x}\Delta x\right)\pi r^2$ on face CD.
3. The shear force, $\tau \times 2\pi r \Delta x$ on the surface of fluid element. As there is no acceleration, hence the summation of all forces in the direction of flow must be zero *i.e.*,

$$p\pi r^2 - \left(p + \frac{\partial p}{\partial x}\Delta x\right)\pi r^2 - \tau \times 2\pi r \times \Delta x = 0$$

or
$$-\frac{\partial p}{\partial x}\Delta x \pi r^2 - \tau \times 2\pi r \times \Delta x = 0$$

or
$$-\frac{\partial p}{\partial x} \cdot r - 2\tau = 0$$

$\therefore$
$$\tau = -\frac{\partial p}{\partial x}\frac{r}{2} \qquad ...(9.1)$$

The shear stress τ across a section varies with 'r' as $\frac{\partial p}{\partial x}$ across a section is constant. Hence shear stress distribution across a section is linear as shown in Fig. 9.2 (a).

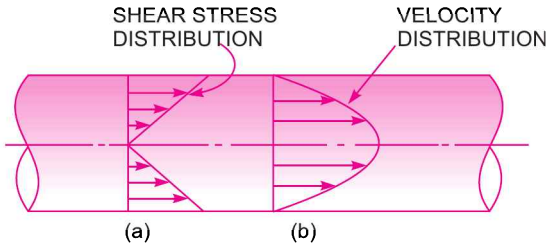


Fig. 9.2 *Shear stress and velocity distribution across a section.*

(i) **Velocity Distribution.** To obtain the velocity distribution across a section, the value of shear stress $\tau = \mu \frac{du}{dy}$ is substituted in equation (9.1).

But in the relation $\tau = \mu \frac{du}{dy}$, y is measured from the pipe wall. Hence

$$y = R - r \quad \text{and} \quad dy = -dr$$

$\therefore$
$$\tau = \mu \frac{du}{-dr} = -\mu \frac{du}{dr}$$

Substituting this value in (9.1), we get

$$-\mu \frac{du}{dr} = -\frac{\partial p}{\partial x}\frac{r}{2} \quad \text{or} \quad \frac{du}{dr} = \frac{1}{2\mu}\frac{\partial p}{\partial x} r$$

Integrating this above equation w.r.t. 'r', we get

$$u = \frac{1}{4\mu}\frac{\partial p}{\partial x} r^2 + C \qquad ...(9.2)$$

where C is the constant of integration and its value is obtained from the boundary condition that at $r = R$, $u = 0$.

$$\therefore \qquad 0 = \frac{1}{4\mu}\frac{\partial p}{\partial x} R^2 + C$$

$$\therefore \qquad C = -\frac{1}{4\mu}\frac{\partial p}{\partial x} R^2$$

Substituting this value of C in equation (9.2), we get

$$u = \frac{1}{4\mu}\frac{\partial p}{\partial x} r^2 - \frac{1}{4\mu}\frac{\partial p}{\partial x} R^2$$

$$= -\frac{1}{4\mu}\frac{\partial p}{\partial x}[R^2 - r^2] \qquad ...(9.3)$$

In equation (9.3), values of μ, $\frac{\partial p}{\partial x}$ and R are constant, which means the velocity, u varies with the square of r. Thus equation (9.3) is a equation of parabola. This shows that the velocity distribution across the section of a pipe is parabolic. This velocity distribution is shown in Fig. 9.2 (*b*).

(*ii*) **Ratio of Maximum Velocity to Average Velocity.** The velocity is maximum, when $r = 0$ in equation (9.3). Thus maximum velocity, U_{max} is obtained as

$$U_{max} = -\frac{1}{4\mu}\frac{\partial p}{\partial x} R^2 \qquad ...(9.4)$$

The average velocity, $\bar{u}$, is obtained by dividing the discharge of the fluid across the section by the area of the pipe (πR^2). The discharge (Q) across the section is obtained by considering the flow through a circular ring element of radius r and thickness dr as shown in Fig. 9.1 (*b*). The fluid flowing per second through this elementary ring

dQ = velocity at a radius $r \times$ area of ring element

$$= u \times 2\pi r\, dr$$

$$= -\frac{1}{4\mu}\frac{\partial p}{\partial x}[R^2 - r^2] \times 2\pi r\, dr$$

$$\therefore \qquad Q = \int_0^R dQ = \int_0^R -\frac{1}{4\mu}\frac{\partial p}{\partial x}(R^2 - r^2) \times 2\pi r\, dr$$

$$= \frac{1}{4\mu}\left(\frac{-\partial p}{\partial x}\right) \times 2\pi \int_0^R (R^2 - r^2)\, r dr$$

$$= \frac{1}{4\mu}\left(\frac{-\partial p}{\partial x}\right) \times 2\pi \int_0^R (R^2 r - r^3)\, dr$$

$$= \frac{1}{4\mu}\left(\frac{-\partial p}{\partial x}\right) \times 2\pi \left[\frac{R^2 r^2}{2} - \frac{r^4}{4}\right]_0^R = \frac{1}{4\mu}\left(\frac{-\partial p}{\partial x}\right) \times 2\pi \left[\frac{R^4}{2} - \frac{R^4}{4}\right]$$

$$= \frac{1}{4\mu}\left(\frac{-\partial p}{\partial x}\right) \times 2\pi \times \frac{R^4}{4} = \frac{\pi}{8\mu}\left(\frac{-\partial p}{\partial x}\right) R^4$$

$\therefore$ Average velocity, $\bar{u} = \dfrac{Q}{\text{Area}} = \dfrac{\dfrac{\pi}{8\mu}\left(\dfrac{-\partial p}{\partial x}\right) R^4}{\pi R^2}$

or $$\bar{u} = \frac{1}{8\mu}\left(\frac{-\partial p}{\partial x}\right) R^2 \qquad ...(9.5)$$

Dividing equation (9.4) by equation (9.5),

$$\frac{U_{max}}{\bar{u}} = \frac{-\dfrac{1}{4\mu}\dfrac{\partial p}{\partial x} R^2}{\dfrac{1}{8\mu}\left(-\dfrac{\partial p}{\partial x}\right) R^2} = 2.0$$

$\therefore$ Ratio of maximum velocity to average velocity = 2.0.

(*iii*) **Drop of Pressure for a given Length (*L*) of a pipe**

From equation (9.5), we have

$$\bar{u} = \frac{1}{8\mu}\left(\frac{-\partial p}{\partial x}\right) R^2 \quad \text{or} \quad \left(\frac{-\partial p}{\partial x}\right) = \frac{8\mu\bar{u}}{R^2}$$

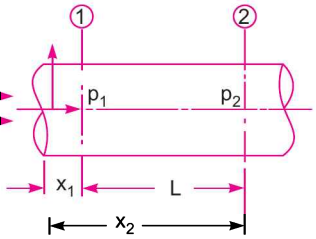


Fig. 9.3

Integrating the above equation w.r.t. x, we get

$$-\int_2^1 dp = \int_2^1 \frac{8\mu\bar{u}}{R^2}\, dx$$

$\therefore$ $$-[p_1 - p_2] = \frac{8\mu\bar{u}}{R^2}[x_1 - x_2] \text{ or } (p_1 - p_2) = \frac{8\mu\bar{u}}{R^2}[x_2 - x_1]$$

$$= \frac{8\mu\bar{u}}{R^2} L \qquad \{\because \quad x_2 - x_1 = L \text{ from Fig. 9.3}\}$$

$$= \frac{8\mu\bar{u}L}{(D/2)^2} \qquad \left\{\because \quad R = \frac{D}{2}\right\}$$

or $$(p_1 - p_2) = \frac{32\mu\bar{u}L}{D^2}, \quad \text{where } p_1 - p_2 \text{ is the drop of pressure.}$$

$\therefore$ Loss of pressure head $= \dfrac{p_1 - p_2}{\rho g}$

$\therefore$ $$\frac{p_1 - p_2}{\rho g} = h_f = \frac{32\mu\bar{u}L}{\rho g D^2} \qquad ...(9.6)$$

Equation (9.6) is called **Hagen Poiseuille Formula.**

Problem 9.1 *A crude oil of viscosity 0.97 poise and relative density 0.9 is flowing through a horizontal circular pipe of diameter 100 mm and of length 10 m. Calculate the difference of pressure at the two ends of the pipe, if 100 kg of the oil is collected in a tank in 30 seconds.*

Solution. Given : $\mu = 0.97 \text{ poise} = \frac{0.97}{10} = 0.097 \text{ Ns/m}^2$

Relative density $= 0.9$

$\therefore$ ρ_0, or density, $= 0.9 \times 1000 = 900 \text{ kg/m}^3$

Dia. of pipe, $D = 100 \text{ mm} = 0.1 \text{ m}$

$L = 10 \text{ m}$

Mass of oil collected, $M = 100 \text{ kg}$

Time, $t = 30 \text{ seconds}$

Calculate difference of pressure or $(p_1 - p_2)$.

The difference of pressure $(p_1 - p_2)$ for viscous or laminar flow is given by

$$p_1 - p_2 = \frac{32\mu\bar{u}L}{D^2}, \text{ where } \bar{u} = \text{average velocity} = \frac{Q}{\text{Area}}$$

Now, mass of oil/sec $= \frac{100}{30} \text{ kg/s}$

$= \rho_0 \times Q = 900 \times Q$ $(\because \quad \rho_0 = 900)$

$\therefore \quad \frac{100}{30} = 900 \times Q$

$\therefore \quad Q = \frac{100}{30} \times \frac{1}{900} = 0.0037 \text{ m}^3/\text{s}$

$\therefore \quad \bar{u} = \frac{Q}{\text{Area}} = \frac{.0037}{\frac{\pi}{4}D^2} = \frac{.0037}{\frac{\pi}{4}(.1)^2} = 0.471 \text{ m/s}.$

For laminar or viscous flow, the Reynolds number (R_e) is less than 2000. Let us calculate the Reynolds number for this problem.

Reynolds number, $R_e^* = \frac{\rho VD}{\mu}$

where $\rho = \rho_0 = 900$, $V = \bar{u} = 0.471$, $D = 0.1$ m, $\mu = 0.097$

$\therefore \quad R_e = 900 \times \frac{.471 \times 0.1}{0.097} = 436.91$

As Reynolds number is less than 2000, the flow is laminar.

$\therefore \quad p_1 - p_2 = \frac{32\mu\bar{u}L}{D^2} = \frac{32 \times 0.097 \times .471 \times 10}{(.1)^2} \text{ N/m}^2$

$= 1462.28 \text{ N/m}^2 = 1462.28 \times 10^{-4} \text{ N/cm}^2 = \mathbf{0.1462 \text{ N/cm}^2}$**. Ans.**

* For derivation, please refer to Art. 12.8.1

Problem 9.2 *An oil of viscosity 0.1 Ns/m^2 and relative density 0.9 is flowing through a circular pipe of diameter 50 mm and of length 300 m. The rate of flow of fluid through the pipe is 3.5 litres/s. Find the pressure drop in a length of 300 m and also the shear stress at the pipe wall.*

Solution. Given : Viscosity, $\mu = 0.1$ Ns/m^2

Relative density $= 0.9$

$\therefore$ ρ_0 or density of oil $= 0.9 \times 1000 = 900$ kg/m^3 ($\because$ Density of water = 1000 kg/m^3)

$D = 50$ mm $= .05$ m

$L = 300$ m

$$Q = 3.5 \text{ litres/s} = \frac{3.5}{1000} = .0035 \text{ m}^3/\text{s}$$

Find (*i*) Pressure drop, $p_1 - p_2$

(*ii*) Shear stress at pipe wall, τ_0

(*i*) **Pressure drop ($p_1 - p_2$)** $= \dfrac{32\mu\bar{u}L}{D^2}$, where $\bar{u} = \dfrac{Q}{\text{Area}} = \dfrac{.0035}{\frac{\pi}{4}D^2} = \dfrac{.0035}{\frac{\pi}{4}(.05)^2} = 1.782$ m/s

The Reynolds number (R_e) is given by, $R_e = \dfrac{\rho VD}{\mu}$

where $\rho = 900$ kg/m^3, V = average velocity $= \bar{u} = 1.782$ m/s

$\therefore$ $$R_e = 900 \times \frac{1.782 \times .05}{0.1} = 801.9$$

As Reynolds number is less than 2000, the flow is viscous or laminar

$\therefore$ $$p_1 - p_2 = \frac{32 \times 0.1 \times 1.782 \times 3000}{(.05)^2}$$

$$= 684288 \text{ N/m}^2 = 68428 \times 10^{-4} \text{ N/cm}^2 = \mathbf{68.43 \text{ N/cm}^2. \text{ Ans.}}$$

(*ii*) **Shear Stress at the pipe wall (τ_0)**

The shear stress at any radius r is given by the equation (9.1)

i.e., $$\tau = -\frac{\partial p}{\partial x}\frac{r}{2}$$

$\therefore$ Shear stress at pipe wall, where $r = R$ is given by

$$\tau_0 = \frac{-\partial p}{\partial x}\frac{R}{2}$$

Now $$\frac{-\partial p}{\partial x} = \frac{-(p_2 - p_1)}{x_2 - x_1} = \frac{p_1 - p_2}{x_2 - x_1} = \frac{p_1 - p_2}{L}$$

$$= \frac{684288}{300}\frac{\text{N/m}^2}{\text{m}} = 2280.96 \text{ N/m}^3$$

and $$R = \frac{D}{2} = \frac{.05}{2} = .025 \text{ m}$$

$$\tau_0 = 2280.96 \times \frac{.025}{2}\frac{\text{N}}{\text{m}^2} = \mathbf{28.512 \text{ N/m}^2. \text{ Ans.}}$$

Problem 9.3 *A laminar flow is taking place in a pipe of diameter 200 mm. The maximum velocity is 1.5 m/s. Find the mean velocity and the radius at which this occurs. Also calculate the velocity at 4 cm from the wall of the pipe.*

Solution. Given : Dia. of pipe, D = 200 mm = 0.20 m

U_{max} = 1.5 m/s

Find (*i*) Mean velocity, $\bar{u}$

(*ii*) Radius at which $\bar{u}$ occurs

(*iii*) Velocity at 4 cm from the wall.

(*i*) **Mean velocity,** $\bar{u}$

Ratio of $\frac{U_{max}}{\bar{u}} = 2.0$ or $\frac{1.5}{\bar{u}} = 2.0$ $\therefore$ $\bar{u} = \frac{1.5}{2.0}$ = **0.75 m/s. Ans.**

(*ii*) **Radius at which** $\bar{u}$ occurs

The velocity, u, at any radius 'r' is given by (9.3)

$$u = -\frac{1}{4\mu}\frac{\partial p}{\partial x}[R^2 - r^2] = -\frac{1}{4\mu}\frac{\partial p}{\partial x} R^2\left[1 - \frac{r^2}{R^2}\right]$$

But from equation (9.4) U_{max} is given by

$$U_{max} = -\frac{1}{4\mu}\frac{\partial p}{\partial x} R^2 \quad \therefore \quad u = U_{max}\left[1 - \left(\frac{r}{R}\right)^2\right] \qquad ...(1)$$

Now, the radius r at which $u = \bar{u}$ = 0.75 m/s

$\therefore$ $$0.75 = 1.5\left[1 - \left(\frac{r}{D/2}\right)^2\right]$$

$$= 1.5\left[1 - \left(\frac{r}{0.2/2}\right)^2\right] = 1.5\left[1 - \left(\frac{r}{0.1}\right)^2\right]$$

$\therefore$ $$\frac{0.75}{0.50} = 1 - \left(\frac{r}{0.1}\right)^2$$

$\therefore$ $$\left(\frac{r}{0.1}\right)^2 = 1 - \frac{.75}{1.50} = 1 - \frac{1}{2} = \frac{1}{2}$$

$\therefore$ $$\frac{r}{0.1} = \sqrt{\frac{1}{2}} = \sqrt{0.5}$$

$\therefore$ $$r = 0.1 \times \sqrt{.5} = 0.1 \times .707 = .0707 \text{ m}$$

= 70.7 mm. Ans.

(*iii*) **Velocity at 4 cm from the wall**

$$r = R - 4.0 = 10 - 4.0 = 6.0 \text{ cm} = 0.06 \text{ m}$$

$\therefore$ The velocity at a radius = 0.06 m

or 4 cm from pipe wall is given by equation (1)

$$= U_{max}\left[1-\left(\frac{r}{R}\right)^2\right] = 1.5\left[1-\left(\frac{.06}{.1}\right)^2\right]$$

$$= 1.5[1.0 - .36] = 1.5 \times .64 = \mathbf{0.96\ m/s.\ Ans.}$$

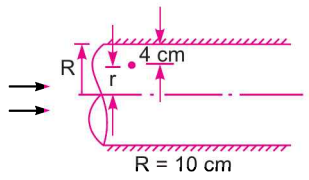


Fig. 9.4

Problem 9.4 *Crude oil of* $\mu = 1.5$ *poise and relative density 0.9 flows through a 20 mm diameter vertical pipe. The pressure gauges fixed 20 m apart read 58.86 N/cm² and 19.62 N/cm² as shown in Fig. 9.5. Find the direction and rate of flow through the pipe.*

Solution. Given : $\mu = 1.5$ poise $= \frac{1.5}{10} = 0.15$ Ns/m²

Relative density $= 0.9$

$\therefore$ Density of oil $= 0.9 \times 1000 = 900$ kg/m³

Dia. of pipe, $D = 20$ mm $= 0.02$ m

$L = 20$ m

$p_A = 58.86$ N/cm² $= 58.86 \times 10^4$ N/m²

$p_B = 19.62$ N/cm² $= 19.62 \times 10^4$ N/m².

Find (*i*) Direction of flow

(*ii*) Rate of flow.

(*i*) **Direction of flow.** To find the direction of flow, the total energy $\left(\frac{p}{\rho g}+\frac{v^2}{2g}+Z\right)$ at the lower end A and at the upper end B is to be calculated. The direction of flow will be given from the higher energy to the lower energy. As here the diameter of the pipe is same and hence kinetic energy at A and B will be same. Hence to find the direction of flow, calculate $\left(\frac{p}{\rho g}+Z\right)$ at A and B.

Taking the level at A as datum. The value of $\left(\frac{p_A}{\rho g}+Z\right)$ at

$$A = \frac{p_A}{\rho g} + Z_A$$

$$= \frac{6\times10^4\times9.81}{900\times9.81} + 0 \quad \{\because\ r = 900\ \text{kg/cm}^2\}$$

$$= 66.67\ \text{m}$$

The value of $\left(\frac{p}{\rho g}+Z\right)$ at $B = \frac{p_B}{\rho g} + Z_B$

$$= \frac{2\times10^4\times9.81}{900\times9.81} + 20 = 22.22 + 20 = 42.22\ \text{m}$$

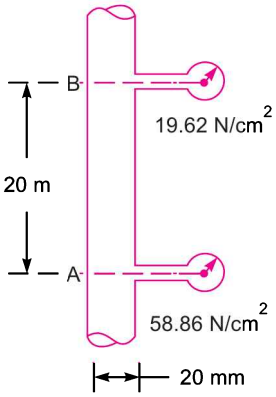


Fig. 9.5

As the value of $\left(\frac{p}{\rho g}+Z\right)$ is higher at A and hence flow takes place from A to B. **Ans.**

(*ii*) **Rate of flow.** The loss of pressure head for viscous flow through circular pipe is given by

$$h_f = \frac{32\mu\bar{u}L}{\rho g D^2}$$

For a vertical pipe h_f = Loss of peizometric head

$$= \left(\frac{p_A}{\rho g} + Z_A\right) - \left(\frac{p_B}{\rho g} + Z_B\right) = 66.67 - 42.22 = 24.45 \text{ m}$$

$$\therefore \quad 24.45 = \frac{32 \times 0.15 \times \bar{u} \times 20.0}{900 \times 9.81 \times (.02)^2}$$

or
$$\bar{u} = \frac{24.45 \times 900 \times 9.81 \times .0004}{32 \times 0.15 \times 20.0} = 0.889 \simeq 0.9 \text{ m/s.}$$

The Reynolds number should be calculated. If Reynolds number is less than 2000, the flow will be laminar and the above expression for loss of pressure head for laminar flow can be used.

Now Reynolds number $= \frac{\rho VD}{\mu}$

where $\rho = 900$ kg/m^3 and $V = \bar{u}$

$\therefore$ Reynolds number $= 900 \times \frac{0.9 \times .02}{0.15} = 108$

As Reynolds number is less than 2000, the flow is laminar.

$\therefore$ Rate of flow = average velocity × area

$$= \bar{u} \times \frac{\pi}{4} D^2 = 0.9 \times \frac{\pi}{4} \times (.02)^2 \text{ m}^3/\text{s} = 2.827 \times 10^{-4} \text{ m}^3/\text{s}$$

$$= \mathbf{0.2827 \text{ litres/s. Ans.}} \qquad (\because \quad 10^{-3} \text{ m}^3 = 1 \text{ litre})$$

Problem 9.5 *A fluid of viscosity 0.7 Ns/m^2 and specific gravity 1.3 is flowing through a circular pipe of diameter 100 mm. The maximum shear stress at the pipe wall is given as 196.2 N/m^2, find (i) the pressure gradient, (ii) the average velocity, and (iii) Reynolds number of the flow.*

Solution. Given : $\mu = 0.7 \frac{\text{Ns}}{\text{m}^2}$

Sp. gr. = 1.3

$\therefore$ Density $= 1.3 \times 1000 = 1300$ kg/m^3

Dia. of pipe, $D = 100$ mm $= 0.1$ m

Shear stress, $\tau_0 = 196.2$ N/m^2

Find (*i*) Pressure gradient, $\frac{dp}{dx}$

(*ii*) Average velocity, $\bar{u}$

(*iii*) Reynolds number, R_e

(*i*) Pressure gradient, $\frac{dp}{dx}$

The maximum shear stress (τ_0) is given by

$$\tau_0 = -\frac{\partial p}{\partial x}\frac{R}{2} \text{ or } 196.2 = -\frac{\partial p}{\partial x} \times \frac{D}{4} = -\frac{\partial p}{\partial x} \times \frac{0.1}{4}$$

$$\therefore \qquad \frac{\partial p}{\partial x} = -\frac{196.2 \times 4}{0.1} = -7848 \text{ N/m}^2 \text{ per m}$$

$\therefore$ Pressure Gradient = **– 7848 N/m² per m. Ans.**

(*ii*) **Average velocity,** $\bar{u}$

$$\bar{u} = \frac{1}{2} U_{max} = \frac{1}{2}\left[-\frac{1}{4\mu}\frac{\partial p}{\partial x} R^2\right] \qquad \left\{\because \quad U_{max} = -\frac{1}{8\mu}\frac{\partial p}{\partial x} R^2\right\}$$

$$= \frac{1}{8\mu} \times \left(-\frac{\partial p}{\partial x}\right) R^2$$

$$= \frac{1}{8 \times 0.7} \times (7848) \times (.05)^2 \qquad \left\{\because \quad R = \frac{D}{2} = \frac{1}{2} = .05\right\}$$

$$= 3.50 \text{ m/s}$$

(*iii*) **Reynolds number,** R_e

$$R_e = \frac{\bar{u} \times D}{\nu} = \frac{\bar{u} \times D}{\mu / \rho} = \frac{\rho \times \bar{u} \times D}{\mu}$$

$$= 1300 \times \frac{3.50 \times 0.1}{0.7} = \textbf{650.00. Ans.}$$

Problem 9.6 *What power is required per kilometre of a line to overcome the viscous resistance to the flow of glycerine through a horizontal pipe of diameter 100 mm at the rate of 10 litres/s ? Take* μ *= 8 poise and kinematic viscosity (v) = 6.0 stokes.*

Solution. Given :

Length of pipe, $L = 1 \text{ km} = 1000 \text{ m}$

Dia. of pipe, $D = 100 \text{ mm} = 0.1 \text{ m}$

Discharge, $Q = 10 \text{ litres/s} = \frac{10}{1000} \text{ m}^3/\text{s} = .01 \text{ m}^3/\text{s}$

Viscosity, $\mu = 8 \text{ poise} = \frac{8}{10}\frac{\text{Ns}}{\text{m}^2} = 0.8 \text{ N s/m}^2$

Kinematic Viscosity, $\nu = 6.0 \text{ stokes}$ $\qquad \left(\because \quad 1 \text{ poise} = \frac{1}{10}\text{Ns/m}^2\right)$

$$= 6.0 \text{ cm}^2/\text{s} = 6.0 \times 10^{-4} \text{ m}^2/\text{s}$$

Loss of pressure head is given by equation (9.6) as $h_f = \frac{32\mu\bar{u}L}{\rho g D^2}$

Power required = $W \times h_f$ watts ...(*i*)

where W = weight of oil flowing per sec = $\rho g \times Q$

Substituting the values of W and h_f in equation (*i*),

Power required $= (\rho g \times Q) \times \frac{(32\,\mu\bar{u}L)}{\rho g D^2}$ watts $= \frac{Q \times 32\,\mu\bar{u}L}{D^2}$ (cancelling ρg)

But $\bar{u} = \frac{Q}{\text{Area}} = \frac{.01}{\frac{\pi}{4}D^2} = \frac{.01 \times 4}{\pi \times (.1)^2} = 1.273 \text{ m/s}$

$\therefore$ Power required $= \dfrac{.01 \times 32 \times 0.8 \times 1.273 \times 1000}{(.1)^2}$

$= 32588.8$ W $=$ **32.588 kW. Ans.**

▶ 9.3 FLOW OF VISCOUS FLUID BETWEEN TWO PARALLEL PLATES

In this case also, the shear stress distribution, the velocity distribution across a section ; the ratio of maximum velocity to average velocity and difference of pressure head for a given length of parallel plates, are to be calculated.

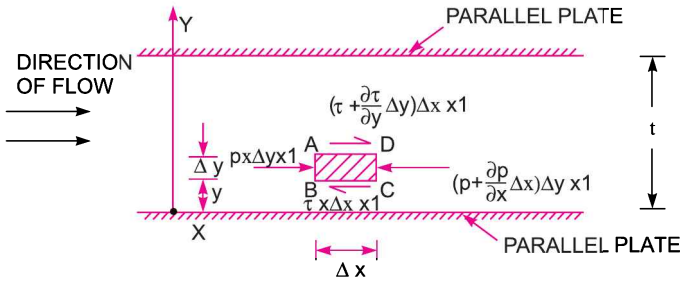


Fig. 9.6 *Viscous flow between two parallel plates.*

Consider two parallel fixed plates kept at a distance 't' apart as shown in Fig. 9.6. A viscous fluid is flowing between these two plates from left to right. Consider a fluid element of length Δx and thickness Δy at a distance y from the lower fixed plate. If p is the intensity of pressure on the face AB of the fluid element then intensity of pressure on the face CD will be $\left(p + \dfrac{\partial p}{\partial x}\Delta x\right)$. Let τ is the shear stress acting on the face BC then the shear stress on the face AD will be $\left(\tau + \dfrac{\partial \tau}{\partial y}\Delta y\right)$. If the width of the element in the direction perpendicular to the paper is unity then the forces acting on the fluid element are :

1. The pressure force, $p \times \Delta y \times 1$ on face AB.
2. The pressure force, $\left(p + \dfrac{\partial p}{\partial x}\Delta x\right)\Delta y \times 1$ on face CD.
3. The shear force, $\tau \times \Delta x \times 1$ on face BC.
4. The shear force, $\left(\tau + \dfrac{\partial \tau}{\partial y}\Delta y\right)\Delta x \times 1$ on face AD.

For steady and uniform flow, there is no acceleration and hence the resultant force in the direction of flow is zero.

$$\therefore \quad p\Delta y \times 1 - \left(p + \frac{\partial p}{\partial x}\Delta x\right)\Delta y \times 1 - \tau\Delta x \times 1 + \left(\tau + \frac{\partial \tau}{\partial y}\Delta y\right)\Delta x \times 1 = 0$$

or $$-\frac{\partial p}{\partial x}\Delta x\Delta y + \frac{\partial \tau}{\partial x}\Delta y\Delta x = 0$$

Dividing by $\Delta x \Delta y$, we get $-\dfrac{\partial p}{\partial x} + \dfrac{\partial \tau}{\partial y} = 0$ or $\dfrac{\partial p}{\partial x} = \dfrac{\partial \tau}{\partial y}$ ...(9.7)

(*i*) **Velocity Distribution.** To obtain the velocity distribution across a section, the value of shear stress $\tau = \mu \frac{du}{dy}$ from Newton's law of viscosity for laminar flow is substituted in equation (9.7).

$$\therefore \quad \frac{\partial p}{\partial x} = \frac{\partial}{\partial y}\left(\mu \frac{du}{dy}\right) = \mu \frac{\partial^2 u}{dy^2}$$

$$\therefore \quad \frac{\partial^2 u}{dy^2} = \frac{1}{\mu}\frac{\partial p}{\partial x}$$

Integrating the above equation w.r.t. y, we get

$$\frac{\partial u}{\partial y} = \frac{1}{\mu}\frac{\partial p}{\partial x} y + C_1 \qquad \left\{\because \frac{\partial p}{\partial x} \text{ is constant}\right\}$$

Integrating again

$$u = \frac{1}{\mu}\frac{\partial p}{\partial x}\frac{y^2}{2} + C_1 y + C_2 \qquad ...(9.8)$$

where C_1 and C_2 are constants of integration. Their values are obtained from the two boundary conditions that is (*i*) at $y = 0$, $u = 0$ (*ii*) at $y = t$, $u = 0$.

The substitution of $y = 0$, $u = 0$ in equation (9.8) gives

$$0 = 0 + C_1 \times 0 + C_2 \text{ or } C_2 = 0$$

The substitution of $y = t$, $u = 0$ in equation (9.8) gives

$$0 = \frac{1}{\mu}\frac{\partial p}{\partial x}\frac{t^2}{2} + C_1 \times t + 0$$

$$\therefore \quad C_1 = -\frac{1}{\mu}\frac{\partial p}{\partial x}\frac{t^2}{2 \times t} = -\frac{1}{2\mu}\frac{\partial p}{\partial x} t$$

Substituting the values of C_1 and C_2 in equation (9.8)

$$u = \frac{1}{2\mu}\frac{\partial p}{\partial x} y^2 + y\left(-\frac{1}{2\mu}\frac{\partial p}{\partial x} t\right)$$

or

$$u = -\frac{1}{2\mu}\frac{\partial p}{\partial x}[ty - y^2] \qquad ...(9.9)$$

In the above equation, μ, $\frac{\partial p}{\partial x}$ and t are constant. It means u varies with the square of y. Hence equation (9.9) is a equation of a parabola. Hence velocity distribution across a section of the parallel plate is parabolic. This velocity distribution is shown in Fig. 9.7 (*a*).

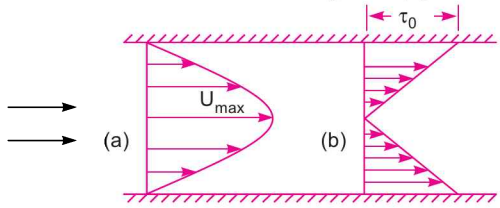


Fig. 9.7 *Velocity distribution and shear stress distribution across a section of parallel plates.*

(*ii*) **Ratio of Maximum Velocity to Average Velocity.** The velocity is maximum, when $y = t/2$. Substituting this value in equation (9.9), we get

$$U_{max} = -\frac{1}{2\mu}\frac{\partial p}{\partial x}\left[t \times \frac{t}{2} - \left(\frac{t}{2}\right)^2\right]$$

$$= -\frac{1}{2\mu}\frac{\partial p}{\partial x}\left[\frac{t^2}{2} - \frac{t^2}{4}\right] = -\frac{1}{2\mu}\frac{\partial p}{\partial x}\frac{t^2}{4} = -\frac{1}{8\mu}\frac{\partial p}{\partial x}t^2 \qquad ...(9.10)$$

The average velocity, $\bar{u}$, is obtained by dividing the discharge (Q) across the section by the area of the section ($t \times 1$). And the discharge Q is obtained by considering the rate of flow of fluid through the strip of thickness dy and integrating it. The rate of flow through strip is

$$dQ = \text{Velocity at a distance } y \times \text{Area of strip}$$

$$= -\frac{1}{2\mu}\frac{\partial p}{\partial x}[ty - y^2] \times dy \times 1$$

$\therefore$ $$Q = \int_0^t dQ = \int_0^t -\frac{1}{2\mu}\frac{\partial p}{\partial x}[ty - y^2]\, dy$$

$$= -\frac{1}{2\mu}\frac{\partial p}{\partial x}\left[\frac{ty^2}{2} - \frac{y^3}{3}\right]_0^t = \frac{1}{2\mu}\frac{\partial p}{\partial x}\left[\frac{t^3}{2} - \frac{t^3}{3}\right]$$

$$= -\frac{1}{2\mu}\frac{\partial p}{\partial x}\frac{t^3}{6} = -\frac{1}{12\mu}\frac{\partial p}{\partial x}t^3$$

$\therefore$ $$\bar{u} = \frac{Q}{\text{Area}} = -\frac{\frac{1}{12\mu}\frac{\partial p}{\partial x} \cdot t^3}{t \times 1} = -\frac{1}{12\mu}\frac{\partial p}{\partial x}t^2 \qquad ...(9.11)$$

Dividing equation (9.10) by equation (9.11), we get

$$\frac{U_{max}}{\bar{u}} = \frac{-\frac{1}{8\mu}\frac{\partial p}{\partial x}t^2}{-\frac{1}{12\mu}\frac{\partial p}{\partial x}t^2} = \frac{12}{8} = \frac{3}{2} \qquad ...(9.12)$$

(*iii*) **Drop of Pressure head for a given Length.** From equation (9.11), we have

$$\bar{u} = -\frac{1}{12\mu}\frac{\partial p}{\partial x}t^2 \quad \text{or} \quad \frac{\partial p}{\partial x} = -\frac{12\mu\bar{u}}{t^2}$$

Integrating this equation w.r.t. x, we get

$$\int_2^1 dp = \int_2^1 -\frac{12\mu\bar{u}}{t^2}\, dx$$

or $$p_1 - p_2 = -\frac{12\mu\bar{u}}{t^2}[x_1 - x_2] = \frac{12\mu\bar{u}}{t^2}[x_2 - x_1]$$

or $$p_1 - p_2 = \frac{12\mu \bar{u} L}{t^2} \qquad [\because x_1 - x_2 = L]$$

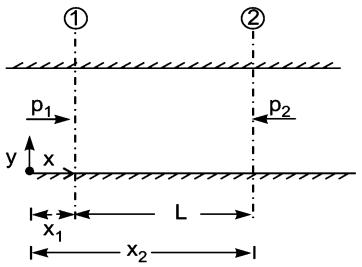


Fig. 9.8

If h_f is the drop of pressure head, then

$$h_f = \frac{p_1 - p_2}{\rho g} = \frac{12\mu \bar{u} L}{\rho g t^2} \qquad ...(9.13)$$

(*iv*) **Shear Stress Distribution.** It is obtained by substituting the value of u from equation (9.9) into

$$\tau = \mu \frac{\partial u}{\partial y}$$

$$\therefore \qquad \tau = \mu \frac{\partial u}{\partial y} = \mu \frac{\partial}{\partial y}\left[-\frac{1}{2\mu}\frac{\partial p}{\partial x}\left(ty - y^2\right)\right] = \mu\left[-\frac{1}{2\mu}\frac{\partial p}{\partial x}(t - 2y)\right]$$

$$\tau = -\frac{1}{2}\frac{\partial p}{\partial x}\,[\,t - 2y] \qquad ...(9.14)$$

In equation (9.14), $\frac{\partial p}{\partial x}$ and t are constant. Hence τ varies linearly with y. The shear stress distribution is shown in Fig. 9.7 (*b*). Shear stress is maximum, when $y = 0$ or t at the walls of the plates. Shear stress is zero, when $y = t/2$ that is at the centre line between the two plates. Max. shear stress (τ_0) is given by

$$\tau_0 = -\frac{1}{2}\frac{\partial p}{\partial x}\,t. \qquad ...(9.15)$$

Problem 9.7 *Calculate : (i) the pressure gradient along flow, (ii) the average velocity, and (iii) the discharge for an oil of viscosity 0.02 Ns/m² flowing between two stationary parallel plates 1 m wide maintained 10 mm apart. The velocity midway between the plates is 2 m/s.*

Solution. Given :

Viscosity, $\mu = .02$ N s/m²

Width, $b = 1$ m

Distance between plates, $t = 10$ mm $= .01$ m

Velocity midway between the plates, $U_{max} = 2$ m/s.

(*i*) **Pressure gradient** $\left(\frac{dp}{dx}\right)$

Using equation (9.10), $U_{max} = -\frac{1}{8\mu}\frac{dp}{dx}t^2$ or $2.0 = -\frac{1}{8 \times .02}\left(\frac{dp}{dx}\right)(.01)^2$

$$\therefore \qquad \frac{dp}{dx} = -\frac{2.0 \times 8 \times .02}{.01 \times .01} = \mathbf{-3200\ N/m^2\ per\ m.\ Ans.}$$

(*ii*) **Average velocity** ($\bar{u}$)

Using equation (9.12), $\frac{U_{max}}{\bar{u}} = \frac{3}{2}$ $\therefore$ $\bar{u} = \frac{2U_{max}}{3} = \frac{2 \times 2}{3} = \mathbf{1.33\ m/s.\ Ans.}$

(*iii*) **Discharge** (Q) = Area of flow $\times\ \bar{u} = b \times t \times \bar{u} = 1 \times .01 \times 1.33 = \mathbf{.0133\ m^3/s.\ Ans.}$

Problem 9.8 *Determine (i) the pressure gradient, (ii) the shear stress at the two horizontal parallel plates and (iii) the discharge per metre width for the laminar flow of oil with a maximum velocity of 2 m/s between two horizontal parallel fixed plates which are 100 mm apart. Given $\mu = 2.4525$ N s/m^2.*

Solution. Given :

$$U_{max} = 2 \text{ m/s}, \; t = 100 \text{ mm} = 0.1 \text{ m}, \; \mu = 2.4525 \text{ N/m}^2$$

Find (*i*) Pressure gradient, $\frac{dp}{dx}$

(*ii*) Shear stress at the wall, τ_0

(*iii*) Discharge per metre width, Q.

(*i*) **Pressure gradient**, $\frac{dp}{dx}$

Maximum velocity, U_{max}, is given by equation (9.10)

$$U_{max} = -\frac{1}{8\mu}\frac{dp}{dx}t^2$$

Substituting the values

or $$2.0 = -\frac{1}{8 \times 2.4525} \times \frac{dp}{dx} \times (.1)^2$$

$\therefore$ $$\frac{dp}{dx} = -\frac{2.0 \times 8 \times 2.4525}{.1 \times .1} = \mathbf{-\ 3924\ N/m^2\ per\ m.\ Ans.}$$

(*ii*) **Shear stress at the wall**, τ_0

τ_0 is given by equation (9.15) as $\tau_0 = -\frac{1}{2}\frac{dp}{dx} \times t = -\frac{1}{2}(-3924) \times 0.1 =$ **196.2 N/m^2. Ans.**

(*iii*) **Discharge per metre width**, Q

$$= \text{Mean velocity} \times \text{Area}$$

$$= \frac{2}{3} U_{max} \times (t \times 1) = \frac{2}{3} \times 2.0 \times 0.1 \times 1 = \mathbf{0.133\ m^3/s.\ Ans.}$$

Problem 9.9 *An oil of viscosity 10 poise flows between two parallel fixed plates which are kept at a distance of 50 mm apart. Find the rate of flow of oil between the plates if the drop of pressure in a length of 1.2 m be 0.3 N/cm^3. The width of the plates is 200 mm.*

Solution. Given :

$$\mu = 10 \text{ poise}$$

$$= \frac{10}{10} \text{ N s/m}^2 = 1 \text{ N s/m}^2 \qquad \left(\because \; 1 \text{ poise} = \frac{1}{10}\frac{\text{Ns}}{\text{m}^2}\right)$$

$$t = 50 \text{ mm} = 0.05 \text{ m}$$

$$p_1 - p_2 = 0.3 \text{ N/m}^2 = 0.3 \times 10^4 \text{ N/m}^2$$

$$L = 1.20 \text{ m}$$

Width, $B = 200$ mm $= 0.20$ m.

Find Q, rate of flow

The difference of pressure is given by equation (9.13)

$$p_1 - p_2 = \frac{12\mu\bar{u}L}{t^2}$$

Substituting the values, we get

$$0.3 \times 10^4 = 12 \times 1.0 \times \frac{\bar{u} \times 1.20}{.05 \times .05}$$

$$\therefore \quad \bar{u} = \frac{0.3 \times 10^4 \times 1.0 \times .05 \times .05}{12 \times 1.20} = 0.52 \text{ m/s}$$

$\therefore$ Rate of flow $= \bar{u} \times \text{Area} = 0.52 \times (B \times t)$

$= 0.52 \times 0.20 \times .05 \text{ m}^3/\text{s} = .0052 \text{ m}^3/\text{s}$

$= 0.0052 \times 10^3$ litre/s = **5.2 litre/s. Ans.**

Problem 9.10 *Water at 15°C flows between two large parallel plates at a distance of 1.6 mm apart. Determine (i) the maximum velocity (ii) the pressure drop per unit length and (iii) the shear stress at the walls of the plates if the average velocity is 0.2 m/s. The viscosity of water at 15°C is given as 0.01 poise.*

Solution. Given : $t = 1.6 \text{ mm} = 1.6 \times 10^{-3} \text{ m}$

$= 0.0016 \text{ m}$

$$\bar{u} = 0.2 \text{ m/sec}, \mu = .01 \text{ poise} = \frac{.01}{10} = 0.001 \text{ N s/m}^2$$

(*i*) **Maximum velocity**, U_{max} is given by equation (9.12)

i.e., $$U_{max} = \frac{3}{2}\bar{u} = 1.5 \times 0.2 = \textbf{0.3 m/s. Ans.}$$

(*ii*) **The pressure drop**, $(p_1 - p_2)$ is given by equation (9.13)

$$p_1 - p_2 = \frac{12\mu\bar{u}L}{t^2}$$

or pressure drop per unit length $= \dfrac{12\mu\bar{u}}{t^2}$

or $$\frac{\partial p}{\partial x} = 12 \times \frac{.01}{10} \times \frac{0.2}{(.0016)^2} = 937.44 \text{ N/m}^2 \text{ per m.}$$

(*iii*) **Shear stress at the walls** is given by equation (9.15)

$$\tau_0 = -\frac{1}{2}\frac{\partial p}{\partial x} \times t = \frac{1}{2} \times 937.44 \times .0016 = \textbf{0.749 N/m}^2\textbf{. Ans.}$$

Problem 9.11 *There is a horizontal crack 40 mm wide and 2.5 mm deep in a wall of thickness 100 mm. Water leaks through the crack. Find the rate of leakage of water through the crack if the difference of pressure between the two ends of the crack is 0.02943 N/cm^2. Take the viscosity of water equal to 0.01 poise.*

Solution. Given :

Width of crack, $b = 40 \text{ mm} = 0.04 \text{ m}$

Depth of crack, $t = 2.5 \text{ mm} = .0025 \text{ m}$

Length of crack, $L = 100 \text{ mm} = 0.1 \text{ m}$

$$p_1 - p_2 = 0.02943 \text{ N/cm}^2 = 0.02943 \times 10^4 \text{ N/m}^2 = 294.3 \text{ N/m}^2$$

$$\mu = .01 \text{ poise} = \frac{.01}{10}\frac{\text{Ns}}{\text{m}^2}$$

Find rate of leakage (Q)

($p_1 - p_2$) is given by equation (9.13) as

$$p_1 - p_2 = \frac{12\mu \bar{u} L}{t^2} \quad \text{or} \quad 294.3 = 12 \times \frac{.01}{10} \times \frac{\bar{u} \times 0.1}{(.0025 \times .0025)}$$

$$\therefore \qquad \bar{u} = \frac{294.3 \times 10 \times .0025 \times .0025}{12 \times .01 \times 0.1} = 1.5328 \text{ m/s}$$

$$\therefore \qquad \text{Rate of leakage} = \bar{u} \times \text{area of cross-section of crack}$$

$$= 1.538 \times (b \times t)$$

$$= 1.538 \times .04 \times .0025 \text{ m}^3/\text{s} = 1.538 \times 10^{-4} \text{ m}^3/\text{s}$$

$$= 1.538 \times 10^{-4} \times 10^3 \text{ litre/s} = \mathbf{0.1538 \text{ litre/s. Ans.}}$$

Problem 9.12 *The radial clearance between a hydraulic plunger and the cylinder walls is 0.1 mm; the length of the plunger is 300 mm and diameter 100 mm. Find the velocity of leakage and rate of leakage past the plunger at an instant when the difference of the pressure between the two ends of the plunger is 9 m of water. Take $\mu = 0.0127$ poise.*

Solution. Given :

The flow through the clearance area will be the same as the flow between two parallel surfaces,

$$t = 0.1 \text{ mm} = 0.0001 \text{ m}$$

$$L = 300 \text{ mm} = 0.3 \text{ m}$$

Diameter, $D = 100 \text{ mm} = 0.1 \text{ m}$

Difference of pressure $= \dfrac{p_1 - p_2}{\rho g} = 9 \text{ m of water}$

$$\therefore \qquad p_1 - p_2 = 9 \times 1000 \times 9.81 \text{ N/m}^2 = 88290 \text{ N/m}^2$$

Viscosity, $\mu = .0127 \text{ poise} = \dfrac{.0127}{10}\dfrac{\text{Ns}}{\text{m}^2}$

Find (*i*) Velocity of leakage, *i.e.*, mean velocity $\bar{u}$

(*ii*) Rate of leakage, Q

(*i*) **Velocity of leakage** $(\bar{u})$. The average velocity $(\bar{u})$ is given by equation (9.11)

$$\bar{u} = -\frac{1}{12\mu}\frac{\partial p}{\partial x} t^2$$

$$= \frac{1}{12 \times \dfrac{.0127}{10}} \times \frac{p_1 - p_2}{L} \times (.0001) \times (.0001)$$

$$= \frac{1}{12 \times .0127} \times \frac{88290}{0.3} \times (.0001) \times (.0001)$$

$$= .193 \text{ m/s} = \mathbf{19.3 \text{ cm/s. Ans.}}$$

(*ii*) **Rate of leakage**, Q

$$Q = \bar{u} \times \text{area of flow}$$
$$= 0.193 \times \pi D \times t \text{ m}^3/\text{s} = 0.193 \times \pi \times .1 \times .0001 \text{ m}^3/\text{s}$$
$$= 6.06 \times 10^{-6} \text{ m}^3/\text{s} = 6.06 \times 10^{-6} \times 10^3 \text{ litre/s}$$
$$= \mathbf{6.06 \times 10^{-3} \text{ litre/s. Ans.}}$$

▶ 9.4 KINETIC ENERGY CORRECTION AND MOMENTUM CORRECTION FACTORS

Kinetic energy correction factor is defined as the ratio of the kinetic energy of the flow per second based on actual velocity across a section to the kinetic energy of the flow per second based on average velocity across the same section. It is denoted by α. Hence mathematically,

$$\alpha = \frac{\text{K.E./sec based on actual velocity}}{\text{K.E./sec based on average velocity}} \qquad ...(9.16)$$

Momentum Correction Factor. It is defined as the ratio of momentum of the flow per second based on actual velocity to the momentum of the flow per second based on average velocity across a section. It is denoted by β. Hence mathematically,

$$\beta = \frac{\text{Momentum per second based on actual velocity}}{\text{Momentum per second based on average velocity}}. \qquad ...(9.17)$$

Problem 9.13 *Show that the momentum correction factor and energy correction factor for laminar flow through a circular pipe are 4/3 and 2.0 respectively.*

Solution. (*i*) **Momentum Correction Factor or β**

The velocity distribution through a circular pipe for laminar flow at any radius r is given by equation (9.3)

or
$$u = \frac{1}{4\mu}\left(-\frac{\partial p}{\partial x}\right)(R^2 - r^2) \qquad ...(1)$$

Consider an elementary area dA in the form of a ring at a radius r and of width dr, then

$$dA = 2\pi r\, dr$$

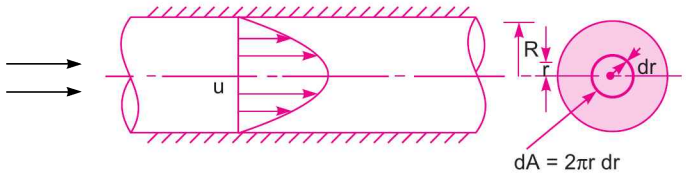


Fig. 9.9

Rate of fluid flowing through the ring

$$= dQ = \text{velocity} \times \text{area of ring element}$$
$$= u \times 2\pi r\, dr$$

Momentum of the fluid through ring per second

$$= \text{mass} \times \text{velocity}$$
$$= \rho \times dQ \times u = \rho \times 2\pi r\, dr \times u \times u = 2\pi\rho\, u^2 r\, dr$$

$\therefore$ Total actual momentum of the fluid per second across the section

$$= \int_0^R 2\pi\rho\, u^2 r\, dr$$

Substituting the value of u from (1)

$$= 2\pi\rho \int_0^R \left[\frac{1}{4\mu}\left(\frac{-\partial p}{\partial x}\right)\left(R^2 - r^2\right)\right]^2 rdr$$

$$= 2\pi\rho\left[\frac{1}{4\mu}\left(\frac{-\partial p}{\partial x}\right)\right]^2 \int_0^R [R^2 - r^2]^2\, rdr$$

$$= 2\pi\rho \frac{1}{\left(16\mu^2\right)}\left(\frac{\partial p}{\partial x}\right)^2 \int_0^R (R^4 + r^4 - 2R^2r^2)\, rdr$$

$$= \frac{\pi\rho}{8\mu^2}\left(\frac{\partial p}{\partial x}\right)^2 \int_0^R (R^4 r + r^5 - 2R^2r^3)\, dr$$

$$= \frac{\pi\rho}{8\mu^2}\left(\frac{\partial p}{\partial x}\right)^2\left[\frac{R^4r^2}{2} + \frac{r^6}{6} - \frac{2R^2r^4}{4}\right]_0^R = \frac{\pi\rho}{8\mu^2}\left(\frac{\partial p}{\partial x}\right)^2\left[\frac{R^6}{2} + \frac{R^6}{6} - \frac{2R^6}{4}\right]$$

$$= \frac{\pi\rho}{8\mu^2}\left(\frac{\partial p}{\partial x}\right)^2 \frac{6R^6 + 2R^6 - 6R^6}{12}$$

$$= \frac{\pi\rho}{8\mu^2}\left(\frac{\partial p}{\partial x}\right)^2 \times \frac{R^6}{6} = \frac{\pi\rho}{48\mu^2}\left(\frac{\partial p}{\partial x}\right)^2 R^6 \qquad \ldots(2)$$

Momentum of the fluid per second based on average velocity

$$= \frac{\text{mass of fluid}}{\text{sec}} \times \text{average velocity}$$

$$= \rho A\bar{u} \times \bar{u} = \rho A\bar{u}^2$$

where A= Area of cross-section = πR^2, $\bar{u}$ = average velocity = $\frac{U_{max}}{2}$

$$= \frac{1}{2} \times \frac{1}{4\mu}\left(-\frac{\partial p}{\partial x}\right) R^2 \qquad \left\{\because \quad U_{max} = \frac{1}{4\mu}\left(-\frac{\partial p}{\partial x}\right) R^2\right\}$$

$$= \frac{1}{8\mu}\left(-\frac{\partial p}{\partial x}\right) R^2$$

$\therefore$ Momentum/sec based on average velocity

$$= \rho \times \pi R^2 \times \left[\frac{1}{8\mu}\left(-\frac{\partial p}{\partial x}\right) R^2\right]^2 = \rho \times \pi R^2 \times \frac{1}{64\mu^2}\left(-\frac{\partial p}{\partial x}\right)^2 R^4$$

$$= \frac{\rho\pi\left(-\frac{\partial p}{\partial x}\right)^2 R^6}{64\mu^2} \qquad \ldots(3)$$

$\therefore$ $$\beta = \frac{\text{Momentum / sec based on actual velocity}}{\text{Momentum / sec based on average velocity}}$$

$$= \frac{\frac{\pi\rho}{48\mu^2}\left(\frac{\partial p}{\partial x}\right)^2 R^6}{\frac{\pi\rho}{64\mu^2}\left(-\frac{\partial p}{\partial x}\right)^2 R^6} = \frac{64}{48} = \frac{\mathbf{4}}{\mathbf{3}}. \textbf{ Ans.}$$

(*ii*) **Energy Correction Factor**, α**.** Kinetic energy of the fluid flowing through the elementary ring of radius 'r' and of width 'dr' per sec

$$= \frac{1}{2} \times \text{mass} \times u^2 = \frac{1}{2} \times \rho dQ \times u^2$$

$$= \frac{1}{2} \times \rho \times (u \times 2\pi r\, dr) \times u^2 = \frac{1}{2}\rho \times 2\pi r u^3\, dr = \pi\rho r u^3\, dr$$

$\therefore$ Total actual kinetic energy of flow per second

$$= \int_0^R \pi\rho r u^3\, dr = \int_0^R \pi\rho r \left[\frac{1}{4\mu}\left(-\frac{\partial p}{\partial x}\right)\left(R^2 - r^2\right)\right]^3 dr$$

$$= \pi\rho \times \left[\frac{1}{4\mu}\left(-\frac{\partial p}{\partial x}\right)\right]^3 \int_0^R [R^2 - r^2]^3\, r dr$$

$$= \pi\rho \times \frac{1}{64\mu^3}\left(-\frac{\partial p}{\partial x}\right)^3 \int_0^R (R^6 - r^6 - 3R^4r^2 + 3R^2r^4) r\, dr$$

$$= \frac{\pi\rho}{64\mu^3}\left(-\frac{\partial p}{\partial x}\right)^3 \int_0^R (R^6 r - r^7 - 3R^4r^3 + 3R^2r^5)\, dr$$

$$= \frac{\pi\rho}{64\mu^3}\left(-\frac{\partial p}{\partial x}\right)^3 \left[\frac{R^6r^2}{2} - \frac{r^8}{8} - \frac{3R^4r^4}{4} + \frac{3R^2r^6}{6}\right]_0^R$$

$$= \frac{\pi\rho}{64\mu^3}\left(-\frac{\partial p}{\partial x}\right)^3 \left[\frac{R^8}{2} - \frac{R^8}{8} - \frac{3R^8}{4} + \frac{3R^8}{6}\right]$$

$$= \frac{\pi\rho}{64\mu^3}\left(-\frac{\partial p}{\partial x}\right)^3 R^8 \left[\frac{12 - 3 - 18 + 12}{24}\right]$$

$$= \frac{\pi\rho}{64\mu^3}\left(-\frac{\partial p}{\partial x}\right)^3 \frac{R^8}{8} \qquad \ldots(4)$$

Kinetic energy of the flow based on average velocity

$$= \frac{1}{2} \times \text{mass} \times \bar{u}^2 = \frac{1}{2} \times \rho A\bar{u} \times \bar{u}^2 = \frac{1}{2} \times \rho A\bar{u}^3$$

Substituting the value of $A = \pi R^2$

and $\bar{u} = \frac{1}{8\mu}\left(-\frac{\partial p}{\partial x}\right) R^2$

$\therefore$ Kinetic energy of the flow/sec

$$= \frac{1}{2} \times \rho \times \pi R^2 \times \left[\frac{1}{8\mu} \left(-\frac{\partial p}{\partial x} \right) R^2 \right]^3$$

$$= \frac{1}{2} \times \rho \times \pi R^2 \times \frac{1}{64 \times 8\mu^3} \left(-\frac{\partial p}{\partial x} \right)^3 \times R^6$$

$$= \frac{\rho\pi}{128 \times 8\mu^3} \left(-\frac{\partial p}{\partial x} \right)^3 \times R^8 \qquad \ldots(5)$$

$\therefore$ $$\alpha = \frac{\text{K.E./sec based on actual velocity}}{\text{K.E./sec based on average velocity}} = \frac{\text{Equation (4)}}{\text{Equation (5)}}$$

$$= \frac{\frac{\pi\rho}{64\mu^3} \left(-\frac{\partial p}{\partial x} \right)^3 \times \frac{R^8}{8}}{\frac{\rho}{128 \times 8\mu^3} \left(-\frac{\partial p}{\partial x} \right)^3 \times R^8} = \frac{128 \times 8}{64 \times 8} = \mathbf{2.0.\ Ans.}$$

▶ 9.5 POWER ABSORBED IN VISCOUS FLOW

For the lubrication of the machine parts, an oil is used. Flow of oil in bearings is an example of viscous flow. If a highly viscous oil is used for lubrication of bearings, it will offer great resistance and thus a greater power loss will take place. But if a light oil is used, a required film between the rotating part and stationary metal surface will not be possible. Hence, the wear of the two surface will take place. Hence an oil of correct viscosity should be used for lubrication. The power required to overcome the viscous resistance in the following cases will be determined :

1. Viscous resistance of Journal Bearings,
2. Viscous resistance of Foot-step Bearings,
3. Viscous resistance of Collar Bearings.

9.5.1 Viscous Resistance of Journal Bearings. Consider a shaft of diameter D rotating in a journal bearing. The clearance between the shaft and journal bearing is filled with a viscous oil. The oil film in contact with the shaft rotates at the same speed as that of shaft while the oil film in contact with journal bearing is stationary. Thus the viscous resistance will be offered by the oil to the rotating shaft.

Let N = speed of shaft in r.p.m.
t = thickness of oil film
L = length of oil film

$\therefore$ Angular speed of the shaft, $\omega = \frac{2\pi N}{60}$

$\therefore$ Tangential speed of the shaft= $\omega \times R$ or $V = \frac{2\pi N}{60} \times \frac{D}{2} = \frac{\pi DN}{60}$

The shear stress in the oil is given by, $\tau = \mu \frac{du}{dy}$

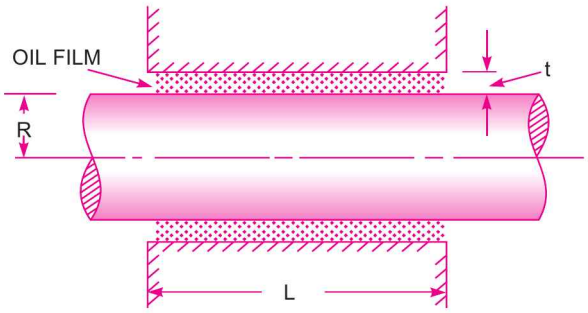


Fig. 9.10 *Journal bearing.*

As the thickness of oil film is very small, the velocity distribution in the oil film can be assumed as linear.

Hence $$\frac{du}{dy} = \frac{V - 0}{t} = \frac{V}{t} = \frac{\pi DN}{60 \times t}$$

$\therefore$ $$\tau = \mu \frac{\pi DN}{60 \times t}$$

$\therefore$ Shear force or viscous resistance = $\tau \times$ Area of surface of shaft

$$= \frac{\mu \pi DN}{60t} \times \pi DL = \frac{\mu \pi^2 D^2 NL}{60t}$$

$\therefore$ Torque required to overcome the viscous resistance,

$$T = \text{Viscous resistance} \times \frac{D}{2}$$

$$= \frac{\mu \pi^2 D^2 NL}{60t} \times \frac{D}{2} = \frac{\mu \pi^2 D^3 NL}{120t}$$

$\therefore$ Power absorbed in overcoming the viscous resistance

$$*P = \frac{2\pi NT}{60} = \frac{2\pi N}{60} \times \frac{\mu \pi^2 D^3 NL}{120t}$$

$$= \frac{\mu \pi^3 \mathbf{D^3 N^2 L}}{\mathbf{60 \times 60 \times t}} \textbf{ watts. Ans.} \qquad ...(9.18)$$

Problem 9.14 *A shaft having a diameter of 50 mm rotates centrally in a journal bearing having a diameter of 50.15 mm and length 100 mm. The angular space between the shaft and the bearing is filled with oil having viscosity of 0.9 poise. Determine the power absorbed in the bearing when the speed of rotation is 60 r.p.m.*

Solution. Given :

Dia. of shaft, $D = 50$ mm or .05 m

Dia. of bearing, $D_1 = 50.15$ mm or 0.05015 m

Length, $L = 100$ mm or 0.1 m

---

*Power, $P = T \times \omega = T \times \frac{2\pi N}{60} = \frac{2\pi NT}{60}$ watts $= \frac{2\pi NT}{60,000}$ kW.

$$\mu \text{ of oil} = 0.9 \text{ poise} = \frac{0.9}{10}\frac{\text{Ns}}{\text{m}^2}$$

$$N = 600 \text{ r.p.m.}$$

$$\text{Power} = ?$$

$\therefore$ Thickness of oil film, $t = \dfrac{D_1 - D}{2} = \dfrac{50.15 - 50}{2}$

$$= \frac{0.15}{2} = 0.075 \text{ mm} = 0.075 \times 10^{-3} \text{ m}$$

Tangential speed of shaft, $V = \dfrac{\pi DN}{60} = \dfrac{\pi \times 0.05 \times 600}{60} = 0.5 \times \pi \text{ m/s}$

Shear stress $\tau = \mu \dfrac{du}{dy} = \mu \dfrac{V}{t} = \dfrac{0.9}{10} \times \dfrac{0.5 \times \pi}{0.075 \times 10^{-3}} = 1883.52 \text{ N/m}^2$

$\therefore$ Shear force (F) $= \tau \times \text{Area} = 1883.52 \times \pi D \times L$

$$= 1883.52 \times \pi \times .05 \times 0.1 = 29.586 \text{ N}$$

Resistance torque $T = F \times \dfrac{D}{2} = 29.586 \times \dfrac{.05}{2} = 0.7387 \text{ Nm}$

Power $= \dfrac{2\pi NT}{60} = \dfrac{2\pi \times 600 \times 0.7387}{60} =$ **46.41 W. Ans.**

Problem 9.15 *A shaft of 100 mm, diameter rotates at 60 r.p.m. in a 200 mm long bearing. Taking that the two surfaces are uniformly separated by a distance of 0.5 mm and taking linear velocity distribution in the lubricating oil having dynamic viscosity of 4 centipoises, find the power absorbed in the bearing.*

Solution. Given :

Dia. of shaft, $D = 100 \text{ mm} = 0.1 \text{ m}$

Length of bearing, $L = 200 \text{ mm} = 0.2 \text{ m}$

$t = 0.5 \text{ mm} = .5 \times 10^{-3} \text{ m}$

$$\mu = 4 \text{ centipoise} = .04 \text{ poise} = \frac{0.04}{10}\frac{\text{Ns}}{\text{m}^2}$$

$$N = 60 \text{ r.p.m.}$$

Find power absorbed

Using equation (9.18), $P = \dfrac{\mu \pi^3 D^3 N^2 L}{60 \times 60 \times t}$

$$= \frac{.04}{10} \times \frac{\pi^3 \times (.1)^3 \times (60)^2 \times 0.2}{60 \times 60 \times 0.5 \times 10^{-3}} = \mathbf{4.961 \times 10^{-2}\ W.\ Ans.}$$

Problem 9.16 *A shaft of diameter 0.35 m rotates at 200 r.p.m. inside a sleeve 100 mm long. The dynamic viscosity of lubricating oil in the 2 mm gap between sleeve and shaft is 8 poises. Calculate the power lost in the bearing.*

Solution. Given :

Dia. of shaft, $D = 0.35 \text{ m}$

Speed of shaft, $N = 200$ r.p.m.
Length of sleeve, $L = 100$ mm $= 0.1$ m
Distance between sleeve and shaft, $t = 2$ mm $= 2 \times 10^{-3}$ m

Viscosity, $\mu = 8$ poise $= \frac{8}{10} \frac{\text{Ns}}{\text{m}^2}$

The power lost in the bearing is given by equation (9.18) as

$$P = \frac{\mu \pi^3 D^3 N^2 L}{60 \times 60 \times t} \text{ watts}$$

$$= \frac{8}{10} \times \frac{\pi^3 \times (.35)^3 \times (200)^2 \times 0.1}{60 \times 60 \times 2 \times 10^{-3}} = \mathbf{590.8\ W = 0.59\ kW.\ Ans.}$$

Problem 9.17 *A sleeve, in which a shaft of diameter 75 mm, is running at 1200 r.p.m., is having a radial clearance of 0.1 mm. Calculate the torque resistance if the length of sleeve is 100 mm and the space is filled with oil of dynamic viscosity 0.96 poise.*

Solution. Given :

Dia. of shaft, $D = 75$ mm $= 0.075$ m
$N = 1200$ r.p.m.
$t = 0.1$ mm $= 0.1 \times 10^{-3}$ m
Length of sleeve, $L = 100$ mm $= 0.1$ m

$$\mu = 0.96 \text{ poise} = \frac{0.96}{10} \frac{\text{Ns}}{\text{m}^2}$$

Tangential velocity of shaft, $V = \frac{\pi DN}{60} = \frac{\pi \times .075 \times 1200}{60} = 4.712$ m/s

Shear stress, $\tau = \mu \frac{V}{t} = \frac{.96}{10} \times \frac{4.712}{.1 \times 10^{-3}} = 4523.5$ N/m^2

Shear force, $F = \tau \times \pi DL$
$= 4523.5 \times \pi \times .075 \times .1 = 106.575$ N

$\therefore$ Torque resistance $= F \times \frac{D}{2}$

$$= 106.575 \times \frac{.075}{2} = \mathbf{3.996\ Nm.\ Ans.}$$

Problem 9.18 *A shaft of 100 mm diameter runs in a bearing of length 200 mm with a radial clearance of 0.025 mm at 30 r.p.m. Find the velocity of the oil, if the power required to overcome the viscous resistance is 183.94 watts.*

Solution. Given :

$D = 100$ mm $= 0.1$ m
$L = 200$ mm $= 0.2$ m
$t = .025$ mm $= 0.025 \times 10^{-3}$ m
$N = 30$ r.p.m. ; H.P. $= 0.25$

Find viscosity of oil, μ.

The h.p. is given by equation (9.18) as

$$P = \frac{\mu\pi^3 D^3 N^2 L}{60 \times 60 \times t} \quad \text{or} \quad 183.94 = \frac{\mu\pi^3 \times (.1)^3 \times (30)^2 \times 0.2}{60 \times 60 \times 0.025 \times 10^{-3}}$$

$$\therefore \quad \mu = \frac{183.94 \times 60 \times 60 \times .025 \times 10^{-3}}{\pi^3 \times .001 \times 900 \times 0.2} \frac{\text{Ns}}{\text{m}^2}$$

$$= 2.96 \frac{\text{Ns}}{\text{m}^2} = 2.96 \times 10 = \textbf{29.6 poise. Ans.}$$

9.5.2 Viscous Resistance of Foot-Step Bearing. Fig. 9.11 shows the foot-step bearing, in which a vertical shaft is rotating. An oil film between the bottom surface of the shaft and bearing is provided, to reduce the wear and tear. The viscous resistance is offered by the oil to the shaft. In this case the radius of the surface of the shaft in contact with oil is not constant as in the case of the journal bearing. Hence, viscous resistance in foot-step bearing is calculated by considering an elementary circular ring of radius r and thickness dr as shown in Fig. 9.11.

Let N = speed of the shaft

t = thickness of oil film

R = radius of the shaft

Area of the elementary ring $= 2\pi r dr$

Now shear stress is given by $t = \mu \frac{du}{dy} = \mu \frac{V}{t}$

where V is the tangential velocity of shaft at radius r and is equal to

$$\omega \times r = \frac{2\pi N}{60} \times r$$

$\therefore$ Shear force on the ring $= dF = \tau \times$ area of elementary ring

$$= \mu \times \frac{2\pi N}{60} \times \frac{r}{t} \times 2\pi r\, dr = \frac{\mu}{15} \frac{\pi^2 N r^2}{t} dr$$

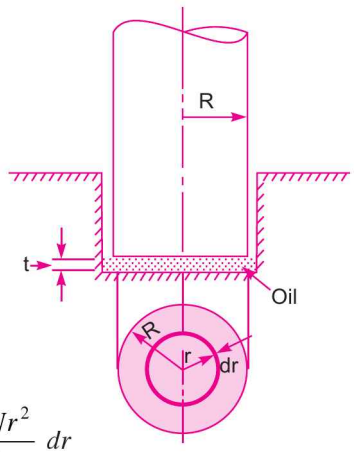


Fig. 9.11 *Foot-step bearing.*

$\therefore$ Torque required to overcome the viscous resistance,

$$dT = dF \times r$$

$$= \frac{\mu}{15t} \pi^2 N r^2\, dr \times r = \frac{\mu}{15t} \pi^2 N r^3\, dr \qquad ...(9.19)$$

$\therefore$ Total torque required to overcome the viscous resistance,

$$T = \int_0^R dT = \int_0^R \frac{\mu}{15t} \pi^2 N r^3 dr$$

$$= \frac{\mu}{15t} \pi^2 N \int_0^R r^3 dr = \frac{\mu}{15t} \pi^2 N \left[\frac{r^4}{4}\right]_0^R = \frac{\mu}{15t} \pi^2 N \frac{R^4}{4}$$

$$= \frac{\mu}{60t} \pi^2 N R^4 \qquad ...(9.19A)$$

$\therefore$ Power absorbed, $P = \frac{2\pi N T}{60}$ watts

$$= \frac{2\pi N}{60} \times \frac{\mu}{60t} \pi^2 N R^4 = \frac{\mu\pi^3 N^2 R^4}{60 \times 30t} \qquad ...(9.20)$$

Problem 9.19 *Find the torque required to rotate a vertical shaft of diameter 100 mm at 750 r.p.m. The lower end of the shaft rests in a foot-step bearing. The end of the shaft and surface of the bearing are both flat and are separated by an oil film of thickness 0.5 mm. The viscosity of the oil is given 1.5 poise.*

Solution. Given :

Dia. of shaft, $D = 100 \text{ mm} = 0.1 \text{ m}$

$\therefore$ $R = \frac{D}{2} = \frac{0.1}{2} = 0.05 \text{ m}$

$N = 750$ r.p.m.

Thickness of oil film, $t = 0.5 \text{ mm} = 0.0005 \text{ m}$

$$\mu = 1.5 \text{ poise} = \frac{1.5}{10} \frac{\text{Ns}}{\text{m}^2}$$

The torque required is given by equation (9.19) as

$$T = \frac{\mu}{60t} \pi^2 N R^4 \text{ Nm}$$

$$= \frac{1.5}{10} \times \frac{\pi^2 \times 750 \times (.05)^4}{60 \times .0005} = \mathbf{0.2305 \text{ Nm. Ans.}}$$

Problem 9.20 *Find the power required to rotate a circular disc of diameter 200 mm at 1000 r.p.m. The circular disc has a clearance of 0.4 mm from the bottom flat plate and the clearance contains oil of viscosity 1.05 poise.*

Solution. Given :

Dia. of disc, $D = 200 \text{ mm} = 0.2 \text{ m}$

$\therefore$ $R = \frac{D}{2} = \frac{0.2}{2} = 0.1 \text{ m}$

$N = 1000$ r.p.m.

Thickness of oil film, $t = 0.4 \text{ mm} = 0.0004 \text{ m}$

$$\mu = 1.05 \text{ poise} = \frac{1.05}{10} \text{ N s/m}^2$$

The power required to rotate the disc is given by equation (9.20) as

$$P = \frac{\mu \pi^3 N^2 R^4}{60 \times 30 \times t} \text{ watts}$$

$$= \frac{1.05}{10} \times \frac{\pi^3 \times 1000^2 \times (0.1)^4}{60 \times 30 \times .0004} = \mathbf{452.1 \text{ W. Ans.}}$$

9.5.3 Viscous Resistance of Collar Bearing. Fig. 9.12 shows the collar bearing, where the face of the collar is separated from bearing surface by an oil film of uniform thickness.

Let N = Speed of the shaft in r.p.m.

R_1 = Internal radius of the collar

R_2 = External radius of the collar

t = Thickness of oil film.

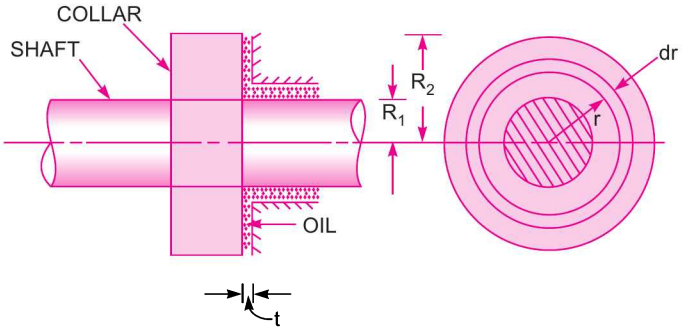


Fig. 9.12 *Collar bearing.*

Consider an elementary circular ring of radius 'r' and width dr of the bearing surface. Then the torque (dT) required to overcome the viscous resistance on the elementary circular ring is the same as given by equation (9.19A) or

$$dT = \frac{\mu}{15t}\ \pi^2 Nr^3\ dr$$

$\therefore$ Total torque, required to overcome the viscous resistance, on the whole collar is

$$T = \int_{R_1}^{R_2} dT = \int_{R_1}^{R_2} \frac{\mu}{15t}\ \pi^2 Nr^3\ dr = \frac{\mu}{15t}\ \pi^2 N \left[\frac{r^4}{4}\right]_{R_1}^{R_2}$$

$$= \frac{\mu}{15t \times 4}\ \pi^2 N\ [R_2^4 - R_1^4] = \frac{\mu}{60t}\ \pi^2 N\ [R_2^4 - R_1^4] \qquad ...(9.21)$$

$\therefore$ Power absorbed in overcoming viscous resistance

$$P = \frac{2\pi NT}{60} = \frac{2\pi N}{60} \times \frac{\mu}{60t}\ \pi^2 N\ [R_2^4 - R_1^4]$$

$$= \frac{\mu\pi^3 N^2}{60 \times 30t}\ [R_2^4 - R_1^4] \text{ watts.} \qquad ...(9.22)$$

Problem 9.21 *A collar bearing having external and internal diameters 150 mm and 100 mm respectively is used to take the thrust of a shaft. An oil film of thickness 0.25 mm is maintained between the collar surface and the bearing. Find the power lost in overcoming the viscous resistance when the shaft rotates at 300 r.p.m. Take μ = 0.91 poise.*

Solution. Given :

External Dia. of collar, $D_2 = 150 \text{ mm} = 0.15 \text{ m}$

$\therefore$ $R_2 = \frac{D_2}{2} = \frac{.15}{2} = 0.075 \text{ m}$

Internal Dia. of collar, $D_1 = 100 \text{ mm} = 0.1 \text{ m}$

$\therefore$ $R_1 = \frac{D_1}{2} = \frac{0.1}{2} = 0.05 \text{ m}$

Thickness of oil film, $t = 0.25 \text{ mm} = 0.00025 \text{ m}$

$N = 300$ r.p.m.

$$\mu = 0.91 \text{ poise} = \frac{0.91}{10}\ \frac{\text{Ns}}{\text{m}^2}$$

The power required is given by equation (9.22) or

$$P = \frac{\mu\pi^3 N^2}{60 \times 30 \times t} [R_2^4 - R_1^4]$$

$$= \frac{0.91}{10} \times \frac{\pi^3 \times 300^2 \times [.075^4 - .05^4]}{60 \times 30 \times .00025}$$

$$= 564314\ [.00003164 - .00000625]$$

$$= 564314 \times .00002539 = \mathbf{14.327\ W.\ Ans.}$$

Problem 9.22 *The external and internal diameters of a collar bearing are 200 mm and 150 mm respectively. Between the collar surface and the bearing, an oil film of thickness 0.25 mm and of viscosity 0.9 poise, is maintained. Find the torque and the power lost in overcoming the viscous resistance of the oil when the shaft is running at 250 r.p.m.*

Solution. Given :

$$D_2 = 200 \text{ mm} = 0.2 \text{ m}$$

$$\therefore \quad R_2 = \frac{D_2}{2} = \frac{0.2}{2} = 0.1 \text{ m}$$

$$D_1 = 150 \text{ mm} = 0.15 \text{ m}$$

$$\therefore \quad R_1 = \frac{D_1}{2} = \frac{0.15}{2} = .075 \text{ m}$$

$$t = 0.25 \text{ mm} = .00025 \text{ m}$$

$$\mu = 0.9 \text{ poise} = \frac{0.9}{10} \frac{\text{Ns}}{\text{m}^2}$$

Torque required is given by equation (9.21)

$$\therefore \quad T = \frac{\mu}{60t} \pi^2 N [R_2^4 - R_1^4] = \frac{0.9}{10} \times \frac{\pi^2 \times 250 [0.1^4 - .075^4]}{60 \times 0.00025} \text{ Nm}$$

$$= 14804.4\ [\ .0001 - .00003164] = \mathbf{1.0114\ Nm.\ Ans.}$$

$\therefore$ Power lost in viscous resistance

$$= \frac{2\pi NT}{60} = \frac{2\pi \times 250 \times 1.0114}{60} = \mathbf{26.48\ W.\ Ans.}$$

▶ 9.6 LOSS OF HEAD DUE TO FRICTION IN VISCOUS FLOW

The loss of pressure head, h_f in a pipe of diameter D, in which a viscous fluid of viscosity μ is flowing with a velocity $\bar{u}$ is given by Hagen Poiseuille formula *i.e.*, by equation (9.6) as

$$h_f = \frac{32\mu\bar{u}L}{\rho g D^2} \qquad ...(i)$$

where L = length of pipe

The loss of head due to friction* is given by

$$h_f = \frac{4 . f . L . V^2}{D \times 2g} = \frac{4 . f . L . \bar{u}^2}{D \times 2g} \qquad ...(ii)$$

{$\because$ velocity in pipe is always average velocity $\quad \therefore \quad V = \bar{u}$}

*For derivation, please refer to Art. 10.3.1.

where f = co-efficient of friction between the pipe and fluid.

Equating (*i*) and (*ii*), we get $\frac{32\mu\bar{u}L}{\rho g D^2} = \frac{4 . f . L . \bar{u}^2}{D \times 2g}$

$$f = \frac{32\mu\bar{u}L \times D \times 2g}{4 . L . \bar{u}^2 . \rho g . D^2} = \frac{16\mu}{\bar{u} . \rho . D} \qquad \{\because \quad \bar{u} = V\}$$

$$= 16 \times \frac{\mu}{\rho VD} = 16 \times \frac{1}{R_e}$$

where $\frac{\mu}{\rho VD} = \frac{1}{R_e}$ and R_e = Reynolds number = $\frac{\rho VD}{\mu}$

$$\therefore \qquad f = \frac{16}{R_e}. \qquad \text{...(9.23)}$$

Problem 9.23 *Water is flowing through a 200 mm diameter pipe with coefficient of friction f = 0.04. The shear stress at a point 40 mm from the pipe axis is 0.00981 N/cm^2. Calculate the shear stress at the pipe wall.*

Solution. Given :

Dia. of pipe, D = 200 mm = 0.20 m

Co-efficient of friction, f = 0.04

Shear stress at r = 40 mm, τ = 0.00981 N/cm^2

Let the shear stress at pipe wall = τ_0.

First find whether the flow is viscous or not. The flow will be viscous if Reynolds number R_e is less than 2000.

Using equation (9.23), we get $f = \frac{16}{R_e}$ or $.04 = \frac{16}{R_e}$

$$\therefore \qquad R_e = \frac{16}{.04} = 400$$

This means flow is viscous. The shear stress in case of viscous flow through a pipe is given by the equation (9.1) as

$$\tau = -\frac{\partial p}{\partial x}\frac{r}{2}$$

But $\frac{\partial p}{\partial x}$ is constant across a section. Across a section, there is no variation of x and there is no variation of p.

$$\therefore \qquad \tau \propto r$$

At the pipe wall, radius = 100 mm and shear stress is τ_0

$$\therefore \qquad \frac{\tau}{r} = \frac{\tau_0}{100} \quad \text{or} \quad \frac{0.00981}{40} = \frac{\tau_0}{100}$$

$$\therefore \qquad \tau_0 = \frac{100 \times 0.00981}{40} = \mathbf{0.0245\ N/cm^2.\ Ans.}$$

Problem 9.24 *A pipe of diameter 20 cm and length 10^4 m is laid at a slope of 1 in 200. An oil of sp. gr. 0.9 and viscosity 1.5 poise is pumped up at the rate of 20 litres per second. Find the head lost due to friction. Also calculate the power required to pump the oil.*

Solution. Given :

Dia. of pipe, $D = 20 \text{ cm} = 2.50 \text{ m}$

Length of pipe, $L = 10000 \text{ m}$

Slope of pipe, $i = 1 \text{ in } 200 = \dfrac{1}{200}$

Sp. gr. of oil, $S = 0.9$

$\therefore$ Density of oil, $\rho = 0.9 \times 1000 = 900 \text{ kg/m}^3$

Viscosity of oil, $\mu = 1.5 \text{ poise} = \dfrac{1.5}{10} \dfrac{\text{Ns}}{\text{m}^2}$

Discharge, $Q = 20 \text{ litre/s} = 0.02 \text{ m}^3/\text{s}$ $\{\because \quad 1000 \text{ litres} = 1 \text{ m}^3\}$

$\therefore$ Velocity of flow, $\bar{u} = \dfrac{Q}{\text{Area}} = \dfrac{0.020}{\dfrac{\pi}{4}D^2} = \dfrac{0.020}{\dfrac{\pi}{4}(.2)^2} = 0.6366 \text{ m/s}$

$\therefore$ R_e = Reynolds number

$$= \frac{\rho VD}{\mu} = \frac{900 \times 0.6366 \times .2}{\dfrac{1.5}{10}}$$

$$= \frac{900 \times .6366 \times .2 \times 10}{1.5} \qquad \{\because \ V = \bar{u} = 0.6366\}$$

$$= 763.89$$

As the Reynolds number is less than 2000, the flow is viscous. The co-efficient of friction for viscous flow is given by equation (9.23) as

$$f = \frac{16}{R_e} = \frac{16}{763.89} = 0.02094$$

$\therefore$ Head lost due to friction, $h_f = \dfrac{4 . f . L . \bar{u}^2}{D \times 2g}$

$$= \frac{4 \times .02094 \times 10000 \times (.6366)^2}{0.2 \times 2 \times 9.81} \text{ m} = \textbf{86.50 m. Ans.}$$

Due to slope of pipe 1 in 200, the height through which oil is to be raised by pump

$$= \text{Slope} \times \text{Length of pipe}$$

$$= i \times L = \frac{1}{200} \times 10000 = 50 \text{ m}$$

$\therefore$ Total head against which pump is to work,

$$H = h_f + i \times L = 86.50 + 50 = 136.50 \text{ m}$$

$\therefore$ Power required to pump the oil

$$= \frac{\rho g \,.\, Q \,.\, H}{1000} = \frac{900 \times 9.81 \times 0.20 \times 136.50}{1000} = \textbf{24.1 kW. Ans.}$$

▶ 9.7 MOVEMENT OF PISTON IN DASH-POT

Consider a piston moving in a vertical dash-pot containing oil as shown in Fig. 9.13.

Let D = Diameter of piston,
L = Length of piston,
W = Weight of piston,
μ = Viscosity of oil,
V = Velocity of piston,
$\bar{u}$ = Average velocity of oil in the clearance,
l = Clearance between the dash-pot and piston,
Δp = Difference of pressure intensities between the two ends of the piston.

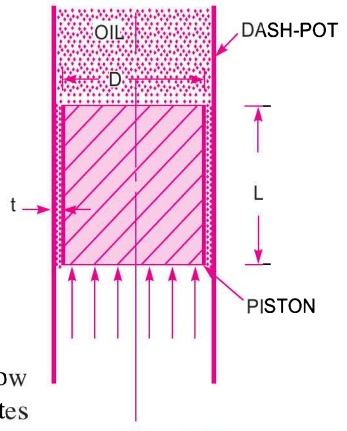


Fig. 9.13

The flow of oil through clearance is similar to the viscous flow between two parallel plates. The difference of pressure for parallel plates for length 'L' is given by

$$\Delta p = \frac{12\mu \bar{u} L}{t^2} \qquad ...(i)$$

Also the difference of pressure at the two ends of piston is given by,

$$\Delta p = \frac{\text{Weight of piston}}{\text{Area of piston}} = \frac{W}{\frac{\pi}{4} D^2} = \frac{4W}{\pi D^2} \qquad ...(ii)$$

Equating (i) and (ii), we get $\dfrac{12\mu \bar{u} L}{t^2} = \dfrac{4W}{\pi D^2}$

$$\therefore \qquad \bar{u} = \frac{4W}{\pi D^2} \times \frac{t^2}{12\mu L} = \frac{Wt^2}{3\pi \mu L D^2} \qquad ...(iii)$$

V is the velocity of piston or the velocity of oil in dash-pot in contact with piston. The rate of flow of oil in dash-pot

$$= \text{velocity} \times \text{area of dash-pot} = V \times \frac{\pi}{4} D^2$$

Rate of flow through clearance = velocity through clearance × area of clearance = $\bar{u} \times \pi D \times t$

Due to continuity equation, rate of flow through clearance must be equal to rate of flow through dash-pot.

$$\therefore \qquad \bar{u} \times \pi D \times t = V \times \frac{\pi}{4} D^2$$

$$\therefore \qquad \bar{u} = V \times \frac{\pi}{4} D^2 \times \frac{1}{\pi D \times t} = \frac{VD}{4t} \qquad ...(iv)$$

Equating the value of $\bar{u}$ from (iii) and (iv), we get

$$\frac{Wt^2}{3\pi \mu L D^2} = \frac{VD}{4t}$$

$$\mu = \frac{4t^3W}{3\pi LD^3V} = \frac{4Wt^3}{3\pi LD^3V}. \qquad ...(9.24)$$

Problem 9.25 *An oil dash-pot consists of a piston moving in a cylinder having oil. This arrangement is used to damp out the vibrations. The piston falls with uniform speed and covers 5 cm in 100 seconds. If an additional weight of 1.36 N is placed on the top of the piston, it falls through 5 cm in 86 seconds with uniform speed. The diameter of the piston is 7.5 cm and its length is 10 cm. The clearance between the piston and the cylinder is 0.12 cm which is uniform throughout. Find the viscosity of oil.*

Solution. Given :

Distance covered by piston due to self weight, = 5 cm

Time taken, = 100 sec

Additional weight, = 1.36 N

Time taken to cover 5 cm due to additional weight, = 86 sec

Dia. of piston, D = 7.5 cm = 0.075 m

Length of piston, L = 10 cm = 0.1 m

Clearance, t = 0.12 cm = 0.0012 m

Let the viscosity of oil = μ

W = Weight of piston,

V = Velocity of piston without additional weight,

V^* = Velocity of piston with additional weight.

Using equation (9.24), we have

$$\mu = \frac{4Wt^3}{3\pi D^3LV} = \frac{4[W + 1.36]t^3}{3\pi D^3LV^*}$$

or $$\frac{W}{V} = \frac{W + 1.36}{V^*} \qquad \left(\text{Cancelling } \frac{4Wt^3}{3\pi D^3L}\right)$$

or $$\frac{V}{V^*} = \frac{W}{W + 1.36} \qquad ...(i)$$

But V = Velocity of piston due to self weight of piston

$$= \frac{\text{Distance covered}}{\text{Time taken}} = \frac{5}{100} \text{ cm/s}$$

Similarly, $$V^* = \frac{\text{Distance covered due to self weight + additional weight}}{\text{Time taken}}$$

$$= \frac{5}{86} \text{ cm/s}$$

$\therefore$ $$\frac{V}{V^*} = \frac{5}{100} \times \frac{86}{5} = 0.86 \qquad ...(ii)$$

Equating (i) and (ii), we get $\frac{W}{W + 1.36} = 0.86$

or $W = 0.86\ W + .86 \times 1.36$

or $W - 0.86\ W = 0.14\ W = .86 \times 1.36$

$$\therefore \qquad W = \frac{0.86 \times 1.36}{0.14} = 8.354 \text{ N}$$

Using equation (9.24), we get $\mu = \dfrac{4Wt^3}{3\pi D^3 LV}$

$$= \frac{4 \times 8.354 \times (.0012)^3}{3\pi \times (0.075)^3 \times .10 \times \left(\frac{5}{100} \times \frac{1}{100}\right)} \left\{\because V = \frac{5}{100} \text{ cm/s} = \frac{5}{100} \times \frac{1}{100} \text{ m/s}\right\}$$

$$= 0.29 \text{ N s/m}^2 = 0.29 \times 10 \text{ poise} = \textbf{2.9 poise. Ans.}$$

▶ 9.8 METHODS OF DETERMINATION OF CO-EFFICIENT OF VISCOSITY

The following are the experimental methods of determining the co-efficient of viscosity of a liquid:

1. Capillary tube method,
2. Falling sphere resistance method,
3. By rotating cylinder method, and
4. Orifice type viscometer.

The apparatus used for determining the viscosity of a liquid is called viscometer.

9.8.1 Capillary Tube Method. In capillary tube method, the viscosity of a liquid is calculated by measuring the pressure difference for a given length of the capillary tube. The Hagen Poiseuille law is used for calculating viscosity.

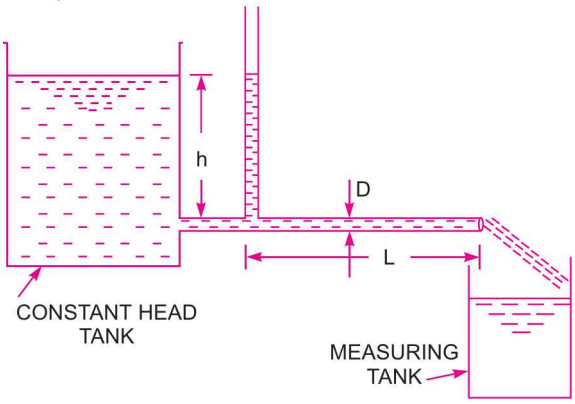


Fig. 9.14 *Capillary tube viscometer.*

Fig. 9.14 shows the capillary tube viscometer. The liquid whose viscosity is to be determined is filled in a constant head tank. The liquid is maintained at constant temperature and is allowed to pass through the capillary tube from the constant head tank. Then, the liquid is collected in a measuring tank for a given time. Then the rate of liquid collected in the tank per second is determined. The pressure head 'h' is measured at a point far away from the tank as shown in Fig. 9.14.

Then h = Difference of pressure head for length L.

The pressure at outlet is atmospheric.

Let D = Diameter of capillary tube,

L = Length of tube for which difference of pressure head is known,

ρ = Density of fluid,

and μ = Co-efficient of viscosity.

Using Hagen Poiseuille's Formula, $h = \dfrac{32\mu\bar{u}L}{\rho g D^2}$

But $\bar{u} = \dfrac{Q}{\text{Area}} = \dfrac{Q}{\frac{\pi}{4}D^2}$

where Q is rate of liquid flowing through tube.

$$h = \frac{32\mu \times \dfrac{Q}{\frac{\pi}{4}D^2} \times L}{\rho g D^2} = \frac{128\,\mu\, Q\,.\,L}{\pi\rho g D^4}$$

or $$\mu = \frac{\pi\rho g h \bar{D}^4}{128\, Q\,.\,L} \qquad ...(9.25)$$

Measurement of D should be done very accurately.

9.8.2 Falling Sphere Resistance Method.

Theory. This method is based on Stoke's law, according to which the drag force, F on a small sphere moving with a constant velocity, U through a viscous fluid of viscosity, μ for viscous conditions is given by

$$F = 3\pi\mu U d \qquad ...(i)$$

where d = diameter of sphere

U = velocity of sphere.

When the sphere attains a constant velocity U, the drag force is the difference between the weight of sphere and buoyant force acting on it.

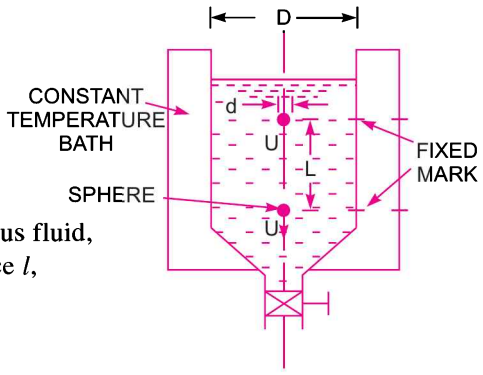


Fig. 9.15 *Falling sphere resistance method.*

Let L = distance travelled by sphere in viscous fluid,

t = time taken by sphere to cover distance l,

ρ_s = density of sphere,

ρ_f = density of fluid,

W = weight of sphere,

and F_B = buoyant force acting on sphere.

Then constant velocity of sphere, $U = \dfrac{L}{t}$

Weight of sphere, W = volume × density of sphere × g

$$= \frac{\pi}{6} d^3 \times \rho_s \times g \qquad \left\{\because \text{ volume of sphere} = \frac{\pi}{6}d^3\right\}$$

and buoyant force, F_B = weight of fluid displaced

= volume of liquid displaced × density of fluid × g

$$= \frac{\pi}{6} d^3 \times \rho_f \times g \quad \{\text{volume of liquid displaced = volume of sphere}\}$$

For equilibrium,

Drag force = Weight of sphere – buoyant force

or $F = W - F_B$

Substituting the values of F, W and F_B, we get

$$3\pi\mu Ud = \frac{\pi}{6} d^3 \times \rho_s \times g - \frac{\pi}{6} d^3 \times \rho_f \times g = \frac{\pi}{6} d^3 \times g\,[\rho_s - \rho_f]$$

or
$$\mu = \frac{\pi}{6} \frac{d^3 \times g[\rho_s - \rho_f]}{3\pi Ud} = \frac{gd^2}{18U}\,[\rho_s - \rho_f] \qquad ...(9.26)$$

where ρ_f = Density of liquid

Hence in equation (9.26), the values of d, U, ρ_s and ρ_f are known and hence the viscosity of liquid can be determined.

Method. Thus this method consists of a tall vertical transparent cylindrical tank, which is filled with the liquid whose viscosity is to be determined. This tank is surrounded by another transparent tank to keep the temperature of the liquid in the cylindrical tank to be constant.

A spherical ball of small diameter 'd' is placed on the surface of liquid. Provision is made to release this ball. After a short distance of travel, the ball attains a constant velocity. The time to travel a known vertical distance between two fixed marks on the cylindrical tank is noted to calculate the constant velocity U of the ball. Then with the known values of d, ρ_s, ρ_f the viscosity μ of the fluid is calculated by using equation (9.26).

9.8.3 Rotating Cylinder Method. This method consists of two concentric cylinders of radii R_1 and R_2 as shown in Fig. 9.16. The narrow space between the two cylinders is filled with the liquid whose viscosity is to be determined. The inner cylinder is held stationary by means of a torsional spring while outer cylinder is rotated at constant angular speed ω. The torque T acting on the inner cylinder is measured by the torsional spring. The torque on the inner cylinder must be equal and opposite to the torque applied on the outer cylinder.

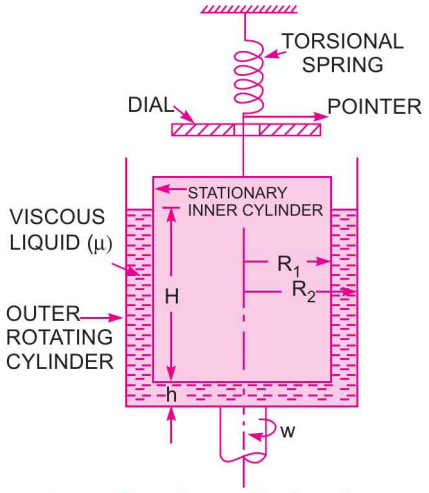


Fig. 9.16 *Rotating cylinder viscometer.*

The torque applied on the outer cylinder is due to viscous resistance provided by liquid in the annular space and at the bottom of the inner cylinder.

Let ω = angular speed of outer cylinder.

Tangential (peripheral) speed of outer cylinder

$= \omega \times R_2$

Tangential velocity of liquid layer in contact with outer cylinder will be equal to the tangential velocity of outer cylinder.

$\therefore$ Velocity of liquid layer with outer cylinder = $\omega \times R_2$

Velocity of liquid layer with inner cylinder = 0 {$\because$ Inner cylinder is stationary}

$\therefore$ Velocity gradient over the radial distance $(R_2 - R_1)$

$$= \frac{du}{dy} = \frac{\omega R_2 - 0}{R_2 - R_1} = \frac{\omega R_2}{R_2 - R_1}$$

$\therefore$ Shear stress (τ)
$$= \mu \frac{du}{dy} = \mu \frac{\omega R_2}{(R_2 - R_1)}$$

$\therefore$ Shear force (F) = shear stress × area of surface

$$= \tau \times 2\pi R_1 H$$

{$\because$ shear stress is acting on surface area = $2\pi R_1 \times H$}

$$= \mu \frac{\omega R_2}{(R_2 - R_1)} \times 2\pi R_1 H$$

The torque T_1 on the inner cylinder due to shearing action of the liquid in the annular space is

T_1 = shear force × radius

$$= \mu \frac{\omega R_2}{(R_2 - R_1)} \times 2\pi R_1 H \times R_1$$

$$= \frac{2\pi\mu\omega H R_1^2 R_2}{(R_2 - R_1)} \quad ...(i)$$

If the gap between the bottom of the two cylinders is 'h', then the torque applied on inner cylinder (T_2) is given by equation (9.19A) as

$$T_2 = \frac{\mu}{60t} \pi^2 N R^4$$

But here $R = R_1$, $t = h$ then $T_2 = \frac{\mu}{60h} \pi^2 N R_1^4$

$$\omega = \frac{2\pi N}{60} \text{ or } N = \frac{60\omega}{2\pi}$$

$\therefore$ $$T_2 = \frac{\mu}{60h} \times \pi^2 \times \frac{60\omega}{2\pi} \times R_1^4 = \frac{\pi\mu\omega}{2h} R_1^4 \quad ...(ii)$$

$\therefore$ Total torque T acting on the inner cylinder is

$$T = T_1 + T_2$$

$$= \frac{2\pi\mu\omega H R_1^2 R_2}{(R_2 - R_1)} + \frac{\pi\mu\omega}{2h} R_1^4 = 2\pi\mu R_1^2 \left[\frac{R_2 H}{R_2 - R_1} + \frac{R_1^2}{4h}\right] \times \omega$$

$\therefore$ $$\mu = \frac{2(R_2 - R_1)hT}{\pi R_1^2 \omega \left[4HhR_2 + R_1^2 (R_2 - R_1)\right]} \quad ...(9.27)$$

where T = torque measured by the strain of the torsional spring,

R_1, R_2 = radii of inner and outer cylinder,

h = clearance at the bottom of cylinders,

H = height of liquid in annular space,

μ = co-efficient of viscosity to be determined.

Hence, the value of μ can be calculated from equation (9.27).

9.8.4 Orifice Type Viscometer. In this method, the time taken by a certain quantity of the liquid whose viscosity is to be determined, to flow through a short capillary tube is noted down. The co-efficient of viscosity is then obtained by comparing with the co-efficient of viscosity of a liquid whose viscosity is known or by the use conversion factors.

Viscometers such as Saybolt, Redwood or Engler are usually used. The principle for all the three viscometer is same. In the United Kingdom, Redwood viscometer is used while in U.S.A., Saybolt viscometer is commonly used.

Fig. 9.17 shows that Saybolt viscometer, which consists of a tank at the bottom of which a short capillary tube is fitted. In this tank the liquid whose viscosity is to be determined is filled. This tank is surrounded by another tank, called constant temperature bath. The liquid is allowed to flow through capillary tube at a standard temperature. The time taken by 60 c.c. of the liquid to flow through the capillary tube is noted down. The initial height of liquid in the tank is previously adjusted to a standard height. From the time measurement, the kinematic viscosity of liquid is known from the relation,

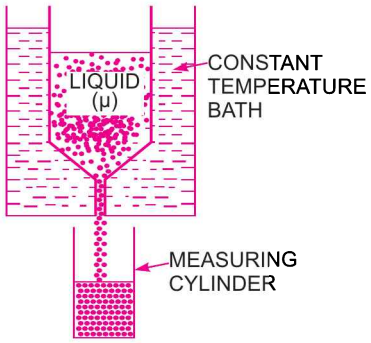


Fig. 9.17 *Saybolt viscometer.*

$$\nu = At - \frac{B}{t}$$

where $A = 0.24$, $B = 190$, t = time noted in seconds, ν = kinematic viscosity in stokes.

Problem 9.26 *The viscosity of an oil of sp. gr. 0.9 is measured by a capillary tube of diameter 50 mm. The difference of pressure head between two points 2 m apart is 0.5 m of water. The mass of oil collected in a measuring tank is 60 kg in 100 seconds. Find the viscosity of oil.*

Solution. Given :

Sp. gr. of oil $= 0.9$

Dia. of capillary tube, $D = 50$ mm $= 5$ cm $= 0.05$ m

Length of tube, $L = 2$ m

Difference of pressure head, $h = 0.5$ m

Mass of oil, $M = 60$ kg

Time, $t = 100$ s

Mass of oil per second $= \dfrac{60}{100} = 0.6$ kg/s

Density of oil, ρ = sp. gr. of oil $\times 1000 = 0.9 \times 1000 = 900$ kg/m^3

$\therefore$ Discharge, $Q = \dfrac{\text{Mass of oil / s}}{\text{Density}} = \dfrac{0.6}{900}$ m^3/s $= 0.000667$ m^3/s

Using equation (9.25), we get viscosity,

$$\mu = \frac{\pi \rho g h D^4}{128\, Q \,.\, L} \qquad [\text{here } h = h_f = 0.5]$$

$$= \frac{\pi \times 900 \times 9.81 \times 0.5 \times (.05)^4}{128 \times 0.000667 \times 2.0} = 0.5075 \text{ (SI Units) N s/m}^2$$

$$= 0.5075 \times 10 \text{ poise} = \textbf{5.075 poise. Ans.}$$

Problem 9.27 *A capillary tube of diameter 2 mm and length 100 mm is used for measuring viscosity of a liquid. The difference of pressure between the two ends of the tube is 0.6867 N/cm^2 and the viscosity of liquid is 0.25 poise. Find the rate of flow of liquid through the tube.*

Solution. Given :

Dia. of capillary tube, $D = 2$ m $= 2 \times 10^{-3}$ m

Length of tube, $L = 100$ mm $= 10$ cm $= 0.1$ m

Difference of pressure, $\Delta p = 0.6867$ N/cm$^2 = 0.6867 \times 10^4$ N/m^2

$\therefore$ Difference of pressure head, $h = \dfrac{\Delta p}{\rho g} = \dfrac{0.6867 \times 10^4}{\rho g}$

Viscosity, $\mu = 0.25$ poise

$$= \frac{0.25}{10} \text{ Ns/m}^2$$

Let the rate of flow of liquid = Q

Using equation (9.25), we get $\mu = \dfrac{\pi \rho g h D^4}{128 \,.\, Q \,.\, L} = \pi \rho g \times \dfrac{\dfrac{0.6867 \times 10^4}{\rho g} \times \left(2 \times 10^{-3}\right)^4}{128 \times Q \times 0.1}$

or $$\frac{0.25}{10} = \frac{\pi \times 0.6867 \times 10^4 \times \left(2 \times 10^{-3}\right)^4}{128 \times Q \times 0.1}$$

or $$Q = \frac{\pi \times 0.6867 \times 10^4 \times 2^4 \times 10^{-12} \times 10}{128 \times 0.1 \times 0.25} \text{ m}^3/\text{s}$$

$$= 107.86 \times 10^{-8} \text{ m}^3/\text{s} = 107.86 \times 10^{-8} \times 10^6 \text{ cm}^3/\text{s}$$

$$= 107.86 \times 10^{-2} \text{ cm}^3/\text{s} = \mathbf{1.078 \text{ cm}^3/\text{s. Ans.}}$$

Problem 9.28 *A sphere of diameter 2 mm falls 150 mm in 20 seconds in a viscous liquid. The density of the sphere is 7500 kg/m^3 and of liquid is 900 kg/m^3. Find the co-efficient of viscosity of the liquid.*

Solution. Given :

Dia. of sphere, $d = 2 \text{ mm} = 2 \times 10^{-3}$ m

Distance travelled by sphere $= 150 \text{ mm} = 0.15$ m

Time taken, $t = 20$ seconds

Velocity of sphere, $U = \dfrac{\text{Distance}}{\text{Time}} = \dfrac{0.15}{20} = .0075$ m/s

Density of sphere, $\rho_s = 7500 \text{ kg/m}^3$

Density of liquid, $\rho_f = 900 \text{ kg/m}^3$

Using relation (9.26), we get $\mu = \dfrac{gd^2}{18U} [\rho_s - \rho_f] = \dfrac{9.81 \times \left[2 \times 10^{-3}\right]^2}{18 \times 0.0075} [7500 - 900]$

$$= \frac{9.81 \times 4 \times 10^{-6} \times 6600}{18 \times 0.0075} = 1.917 \frac{\text{Ns}}{\text{m}^2}$$

$$= 1.917 \times 10 = \mathbf{19.17 \text{ poise. Ans.}}$$

Problem 9.29 *Find the viscosity of a liquid of sp. gr. 0.8, when a gas bubble of diameter 10 mm rises steadily through the liquid at a velocity of 1.2 cm/s. Neglect the weight of the bubble.*

Solution. Given :

Sp. gr. of liquid $= 0.8$

$\therefore$ Density of liquid, $\rho_f = 0.8 \times 1000 = 800 \text{ kg/m}^3$

Dia. of gas bubble, $D = 10 \text{ mm} = 1 \text{ cm} = 0.01$ m

Velocity of bubble, $U = 1.2 \text{ cm/s} = .012$ m/s

As weight of bubble is neglected and density of bubble

$$\rho_s = 0$$

Now using the relation, $\mu = \frac{gd^2}{18U}[\rho_s - \rho_f]$ which is for a falling sphere.

For a rising bubble, the relation will become as

$$\mu = \frac{gd^2}{18U}[\rho_f - \rho_s]$$

Substituting the values, we get $\mu = \frac{9.81 \times .01 \times .01}{18 \times .012}[800 - 0]\ \frac{\text{Ns}}{\text{m}^2} = 3.63\frac{\text{Ns}}{\text{m}^2}$

$= 3.63 \times 10 =$ **36.3 poise. Ans.**

Problem 9.30 *The viscosity of a liquid is determined by rotating cylinder method, in which case the inner cylinder of diameter 20 cm is stationary. The outer cylinder of diameter 20.5 cm, contains the liquid upto a height of 30 cm. The clearance at the bottom of the two cylinders is 0.5 cm. The outer cylinder is rotated at 400 r.p.m. The torque registered on the torsion meter attached to the inner cylinder is 5.886 Nm. Find the viscosity of fluid.*

Solution. Given :

Dia. of inner cylinder, $D_1 = 20$ cm

$\therefore$ Radius of inner cylinder, $R_1 = 10$ cm $= 0.1$ m

Dia. of outer cylinder, $D_2 = 20.5$ cm

$\therefore$ Radius of outer cylinder, $R_2 = \frac{20.5}{2} = 10.25$ cm $= .1025$ m

Height of liquid from bottom of outer cylinder = 30 cm

Clearance at the bottom of two cylinders, $h = 0.5$ cm $= .005$ m

$\therefore$ Height of inner cylinder immersed in liquid

$= 30 - h = 30 - 0.5 = 29.5$ m

or $H = 29.5$ cm $= .295$ m

Speed of outer cylinder, $N = 400$ r.p.m.

$\therefore$ $\omega = \frac{2\pi N}{60} = \frac{2 \times \pi \times 400}{60} = 41.88$

Torque measured, $T = 5.886$ Nm

Using equation (9.27), we get $\mu = \frac{2(R_2 - R_1) \times h \times T}{\pi R_1^2 \omega \left[4HhR_2 + R_1^2(R_2 - R_1)\right]}$

$$= \frac{2(.1025 - 0.1) \times .005 \times 5.886}{\pi \times (.1)^2 \times 41.88\left[4 \times .295 \times .005 \times .1025 + .1^2\,(.1025 - .1)\right]}$$

$$= \frac{2 \times .0025 \times .005 \times 5.886}{\pi \times .01 \times 41.88\,[.0006047 - .000025]}$$

$= 0.19286$ Ns/m^2 $= 0.19286 \times 10 =$ **1.9286 poise. Ans.**

Problem 9.31 *A sphere of diameter 1 mm falls through 335 m in 100 seconds in a viscous fluid. If the relative densities of the sphere and the liquid are 7.0 and 0.96 respectively, determine the dynamic viscosity of the liquid.*

Solution. Given :

Dia. of sphere, $d = 1 \text{ mm} = 0.001 \text{ m}$

Distance travelled by sphere $= 335 \text{ mm} = 0.335 \text{ m}$

Time taken, $t = 100 \text{ seconds}$

$\therefore$ Velocity of sphere, $U = \dfrac{\text{Distance}}{\text{Time}} = \dfrac{0.335}{100} = 0.00335 \text{ m/sec}$

Relative density of sphere $= 7$

$\therefore$ Density of sphere, $\rho_s = 7 \times 1000 = 7000 \text{ kg/m}^3$

Relative density of liquid $= 0.96$

$\therefore$ Density of liquid, $\rho_f = 0.96 \times 1000 = 960 \text{ kg/m}^3$

Using the relation (9.26), we get $\mu = \dfrac{gd^2}{18U}[\rho_s - \rho_f] = \dfrac{9.81 \times 0.001^2}{18 \times 0.00335}[7000 - 960]$

$$= \frac{0.00000981 \times 6040}{18 \times 0.00335} = 0.981 \text{ Ns/m}^2$$

$$= 0.981 \times 10 = \textbf{9.81 poise. Ans.}$$

Problem 9.32 *Determine the fall velocity of 0.06 mm sand particle (specific gravity = 2.65) in water at 20°C, take $\mu = 10^{-3}$ kg/ms.*

Solution. Given :

Dia. of sand particle, $d = 0.06 \text{ mm} = 0.06 \times 10^{-3} \text{ m}$

Specific gravity of sand $= 2.65$

$\therefore$ Density of sand, $\rho_s = 2.65 \times 1000 \text{ kg/m}^3$ ($\because$ ρ for water in S.I. unit $= 1000 \text{ kg/m}^3$)

$= 2650 \text{ kg/m}^3$

Viscosity of water, $\mu^* = 10^{-3} \text{ kg/ms} = 10^{-3} \text{ Ns/m}^2$ $\left[\because \dfrac{\text{Ns}}{\text{m}^2} = \left(\text{kg} \times \dfrac{\text{m}}{\text{s}^2}\right) \times \dfrac{\text{s}}{\text{m}^2} = \dfrac{\text{kg}}{\text{ms}}\right]$

Density of water, $\rho_f = 1000 \text{ kg/m}^3$

Sand particle is just like a sphere.

For equilibrium of sand particle,

Drag force = Weight of sand particle – buoyant force

or $F_D = W - F_B$ ...(*i*)

But $F_D = 3\pi\mu \times U \times d$, where U = Velocity of particle

$= 3\pi \times 10^{-3} \times U \times 0.06 \times 10^{-3} \text{ N}$

W = Weight of sand particle

$$= \frac{\pi}{6} \times d^3 \times \rho_s \times g = \frac{\pi}{6} \times (0.06 \times 10^{-3})^3 \times 2650 \times 9.81 \text{ N}$$

F_B = Buoyant force = Weight of water displaced

---

*Viscosity in S.I. unit $= \text{N s/m}^2$. But $1 \text{ N} = 1 \text{ kg} \times 1 \text{ m/s}^2$

Hence viscosity $= \left(\dfrac{1 \text{ kg} \times 1 \text{ m}}{\text{s}^2}\right) \times \dfrac{\text{s}}{\text{m}^2} = \text{kg/ms}$. Hence $\text{kg/ms} = \dfrac{\text{Ns}}{\text{m}^2}$.

$$= \frac{\pi}{6} \times d^3 \times \rho_f \times g = \frac{\pi}{6} \times (0.06 \times 10^{-3})^3 \times 1000 \times 9.81 \text{ N}$$

Substituting the above values in equation (*i*), we get

$$3\pi \times 10^{-3} \times U \times 0.06 \times 10^{-3} = \frac{\pi}{6} \times (0.06 \times 10^{-3})^3 \times 2650 \times 9.81 - \frac{\pi}{6} \times (0.06 \times 10^{-3})^3 \times 1000 \times 9.81$$

Cancelling $(\pi \times 0.06 \times 10^{-3})^2$ throughout, we get

$$3 \times U = \frac{1}{6} \times 0.06^2 \times 10^{-3} \times 2650 \times 9.81 - \frac{1}{6} \times 0.06^2 \times 10^{-3} \times 1000 \times 9.81$$

$$= \frac{1}{6} \times 0.06^2 \times 10^{-3} \times 9.81 \ (2650 - 1000)$$

$$= \frac{1}{6} \times 0.0036 \times 10^{-3} \times 9.81 \times 1650 = 0.009712$$

$\therefore$ $U = 0.009712/3 =$ **0.00323 m/sec. Ans.**

HIGHLIGHTS

1. A flow is said to be viscous if the Reynolds number is less than 2000, or the fluid flows in layers.

2. For the viscous flow through circular pipes,

(*i*) Shear stress ..... $\tau = -\frac{\partial p}{\partial x}\frac{r}{2}$ (*ii*) Velocity .....$u = -\frac{1}{4\mu}\frac{\partial p}{\partial x}[R^2 - r^2]$

(*iii*) Ratio of velocities $\frac{U_{max}}{\bar{u}} = 2.0$ (*iv*) Loss of pressure head, $h_f = \frac{32\mu\bar{u}L}{\rho g D^2}$

where $\frac{\partial p}{\partial x}$ = pressure gradient, r = radius at any point,

R = radius of the pipe, U_{max} = maximum velocity or velocity at $r = 0$,

$\bar{u}$ = average velocity = $\frac{Q}{\pi R^2}$, μ = co-efficient of viscosity,

D = diameter of the pipe.

3. For the viscous flow between two parallel plates,

$$u = -\frac{1}{2\mu}\frac{\partial p}{\partial x}(ty - y^2) \quad \text{... Velocity distribution}$$

$$\frac{U_{max}}{\bar{u}} = 1.5 \quad \text{... Ratio of maximum and average velocity}$$

$$h_f = \frac{12\mu\bar{u}L}{\rho g t^2} \quad \text{... Loss of pressure head}$$

$$\tau = -\frac{1}{2}\frac{\partial p}{\partial x}[t - 2y] \quad \text{... Shear stress distribution}$$

where t = thickness or distance between two plates,

y = distance in the vertical direction from the lower plate,

τ = shear stress at any point in flow.

4. The kinetic energy correction factor α is given as

$$\alpha = \frac{\text{K.E. per second based on actual velocity}}{\text{K.E. per second based on average velocity}}$$

$= 2.0$ ... for a circular pipe.

5. Momentum correction factor, β is given by

$$\beta = \frac{\text{Momentum per second based on actual velocity}}{\text{Momentum per second based on average velocity}}$$

$$= \frac{4}{3} \text{ ... for a circular pipe.}$$

6. For the viscous resistance of Journal Bearing.

$$V = \frac{\pi DN}{60}, \frac{du}{dy} = \frac{V}{t} = \frac{\pi DN}{60t}$$

$$\tau = \frac{\mu \pi dN}{60t}, \text{ Shear force} = \frac{\mu \pi^2 D^2 NL}{60t}$$

Torque, $$T = \frac{\mu \pi^2 D^3 NL}{120t} \text{ and power} = \frac{\mu \pi^3 D^3 N^2 L}{60 \times 60 \times t}$$

where L= length of bearing, N = speed of shaft

t = clearance between the shaft and bearing.

7. For the Foot-Step Bearing, the shear force, torque and h.p. absorbed are given as :

Shear force, $$F = \frac{\mu}{15} \frac{\pi^2 N}{t} \frac{R^3}{3}$$

Torque, $$T = \frac{\mu}{60t} \pi^2 NR^4$$

Power $$= \frac{\mu \pi^3 N^2 R^4}{60 \times 30 \times t}$$

where R= radius of the shaft, N = speed of the shaft.

8. For the collar bearing the torque and power absorbed are given as

$$T = \frac{\mu}{60t} \pi^2 N [R_2^4 - R_1^4], \qquad P = \frac{\mu \pi^3 N^2}{60 \times 30t} [R_2^4 - R_1^4]$$

where R_1 = internal radius of the collar, R_2 = external radius of the collar,

t = thickness of oil film, P = power in watts.

9. For the viscous flow the co-efficient of friction is given by, $f = \frac{16}{R_e}$

where R_e = the Reynolds number $= \frac{\rho VD}{\mu} = \frac{VD}{\nu}$.

10. The co-efficient of viscosity is determined by dash-pot arrangement as $\mu = \frac{4Wt^3}{3\pi LD^3 V}$

where W = weight of the piston, t = clearance between dash-pot and piston,

L = length of the piston, D = diameter of the piston,

V = velocity of the piston.

11. The co-efficient of viscosity of a liquid is also determined experimentally by the following method :

(*i*) Capillary tube method, $\mu = \dfrac{\pi \rho g h D^4}{128\, QL}$

(*ii*) Falling sphere method, $\mu = \dfrac{gd^2\left[\rho_s - \rho_f\right]}{18U}$

(*iii*) Rotating cylinder method, $\mu = \dfrac{2\left(R_2 - R_1\right)hT}{\pi R_1^2 \omega\left[4HhR_2 + R_1^2\left(R_2 - R_1\right)\right]}$

where w = specific weight of fluid, L = length of the tube,
D = diameter of the capillary tube, Q = rate of flow of fluid through capillary tube,
d = diameter of the sphere, ρ_s = density of sphere,
ρ_f = density of fluid, U = velocity of sphere,
R_2 = radius of outer rotating cylinder, R_1 = radius of inner stationary cylinder,
T = torque.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define the terms : Viscosity, kinematic viscosity, velocity gradient and pressure gradient.
2. What do you mean by 'Viscous Flow'?
3. Derive an expression for the velocity distribution for viscous flow through a circular pipe. Also sketch the velocity distribution and shear stress distribution across a section of the pipe.
4. Prove that the maximum velocity in a circular pipe for viscous flow is equal to two times the average velocity of the flow. (*Delhi University, December 2002*)
5. Find an expression for the loss of head of a viscous fluid flowing through a circular pipe.
6. What is Hagen Poiseuille's Formula ? Derive an expression for Hagen Poiseuille's Formula.
7. Prove that the velocity distribution for viscous flow between two parallel plates when both plates are fixed across a section is parabolic in nature. Also prove that maximum velocity is equal to one and a half times the average velocity.
8. Show that the difference of pressure head for a given length of the two parallel plates which are fixed and through which viscous fluid is flowing is given by

$$h_f = \frac{12\mu \bar{u} L}{\rho g t^2}$$

where μ = Viscosity of fluid, $\bar{u}$ = Average velocity,
t = Distance between the two parallel plates, L = Length of the plates.

9. Define the terms : Kinetic energy correction factor and momentum correction factor.
10. Prove that for viscous flow through a circular pipe the kinetic energy correction factor is equal to 2 while momentum correction factor = $\dfrac{4}{3}$.
11. A shaft is rotating in a journal bearing. The clearance between the shaft and the bearing is filled with a viscous oil. Find an expression for the power absorbed in overcoming viscous resistance.
12. Prove that power absorbed in overcoming viscous resistance in foot-step bearing is given by

$$P = \frac{\mu \pi^3 N^2 R^4}{60 \times 30 t}$$

where R = Radius of the shaft, N = Speed of the shaft,
t = Clearance between shaft and foot-step bearing, μ = Viscosity of fluid.

13. Show that the value of the co-efficient of friction for viscous flow through a circular pipe is given by,

$$f = \frac{16}{R_e}, \quad \text{where} \quad R_e = \text{Reynolds number.}$$

14. Prove that the co-efficient of viscosity by the dash-pot arrangement is given by,

$$\mu = \frac{4Wt^3}{3\pi LD^3V}$$

where W = Weight of the piston, t = Clearance between dash-pot and piston,
L = Length of piston, D = Diameter of piston,
V = Velocity of piston.

15. What are the different methods of determining the co-efficient of viscosity of a liquid ? Describe any two method in details.

16. Prove that the loss of pressure head for the viscous flow through a circular pipe is given by

$$h_f = \frac{32\mu\,\bar{u}L}{\rho g d^2}$$

where $\bar{u}$ = Average velocity, w = Specific weight.

17. For a laminar steady flow, prove that the pressure gradient in a direction of motion is equal to the shear gradient normal to the direction of motion.

18. Describe Reynolds experiments to demonstrate the two types of flow.

19. For the laminar flow through a circular pipe, prove that :
(*i*) the shear stress variation across the section of the pipe is linear and
(*ii*) the velocity variation is parabolic.

(B) NUMERICAL PROBLEMS

1. A crude oil of viscosity 0.9 poise and sp. gr. 0.8 is flowing through a horizontal circular pipe of diameter 80 mm and of length 15 m. Calculate the difference of pressure at the two ends of the pipe, if 50 kg of the oil is collected in a tank in 15 seconds. [**Ans.** 0.559 N/cm^2]

2. A viscous flow is taking place in a pipe of diameter 100 mm. The maximum velocity is 2 m/s. Find the mean velocity and the radius at which this occurs. Also calculate the velocity at 30 mm from the wall of the pipe. [**Ans.** 1 m/s, r = 35.35 mm, u = 1.68 m/s]

3. A fluid of viscosity 0.5 poise and specific gravity 1.20 is flowing through a circular pipe of diameter 100 mm. The maximum shear stress at the pipe wall is given as 147.15 N/m^2, find : (*a*) the pressure gradient, (*b*) the average velocity, and (*c*) the Reynolds number of the flow.
[**Ans.** (*a*) – 64746 N/m^2 per m, (*b*) 3.678 m/s, (*c*) 882.72]

4. Determine (*a*) the pressure gradient, (*b*) the shear stress at the two horizontal parallel plates and (*c*) the discharge per metre width for the laminar flow of oil with a maximum velocity of 1.5 m/s between two horizontal parallel fixed plates which are 80 mm apart. Take viscosity of oil as $\frac{1.962\ \text{Ns}}{\text{m}^2}$.
[**Ans.** (*a*) – 3678.7 N/m^2 per m, (*b*) 147.15 N/m^2, (*c*) .08 m^3/s]

5. Water is flowing between two large parallel plates which are 2.0 mm apart. Determine : (*a*) maximum velocity, (*b*) the pressure drop per unit length and (*c*) the shear stress at walls of the plate if the average velocity is 0.4 m/s. Take viscosity of water as 0.01 poise.
[**Ans.** (*a*) 0.6 m/s, (*b*) 1199.7 N/m^2 per m, (*c*) 1.199 N/m^3]

6. There is a horizontal crack 50 mm wide and 3 mm deep in a wall of thickness 150 mm. Water leaks through the crack. Find the rate of leakage of water through the crack if the difference of pressure between the two ends of the crack is 245.25 N/m^2. Take the viscosity of water as 0.01 poise. [**Ans.** 183.9 cm^3/s]

7. A shaft having a diameter of 10 cm rotates centrally in a journal bearing having a diameter of 10.02 cm and length 20 cm. The annular space between the shaft and the bearing is filled with oil having viscosity of 0.8 poise. Determine the power absorbed in the bearing when the speed of rotation is 500 r.p.m.
[**Ans.** 343.6 W]

8. A shaft 150 mm diameter runs in a bearing of length 300 mm, with a radial clearance of 0.04 mm at 40 r.p.m. Find the viscosity of the oil, if the power required to overcome the viscous resistance is 220.725 W.
[**Ans.** 6.32 poise]

9. Find the torque required to rotate a vertical shaft of diameter 8 cm at 800 r.p.m. The lower end of the shaft rests in a foot-step bearing. The end of the shaft and surface of the bearing are both flat and are separated by an oil film of thickness 0.075 cm. The viscosity of the oil is given as 1.2 poise. [**Ans.** 0.0538 Nm]

10. A collar bearing having external and internal diameters 20 cm and 10 cm respectively is used to take the thrust of a shaft. An oil film of thickness 0.03 cm is maintained between the collar surface and the bearing. Find the power lost in overcoming the viscous resistance when the shaft rotates at 250 r.p.m. Take $\mu = 0.9$ poise. [**Ans.** 30.165 W]

11. Water is flowing through a 150 mm diameter pipe with a co-efficient of friction $f = .05$. The shear stress at a point 40 mm from the pipe wall is 0.01962 N/cm^2. Calculate the shear stress at the pipe wall.
[**Ans.** 0.04198 N/cm^2]

12. An oil dash-pot consists of a piston moving in a cylinder having oil. The piston falls with uniform speed and covers 4.5 cm in 80 seconds. If an additional weight of 1.5 N is placed on the top of the piston, it falls through 4.5 cm in 70 seconds with uniform speed. The diameter of the piston is 10 cm and its length is 15 cm. The clearance between the piston and the cylinder is 0.15 cm, which is uniform throughout. Find the viscosity of oil. [**Ans.** 0.177 poise]

13. The viscosity of oil of sp. gr. 0.8 is measured by a capillary tube of diameter 40 mm. The difference of pressure head between two points 1.5 m apart is 0.3 m of water. The mass of oil collected in a measuring tank is 40 kg in 120 seconds. Find the viscosity of the oil. [**Ans.** 2.36 poise]

14. A capillary tube of diameter 4 mm and length 150 mm is used for measuring viscosity of a liquid. The difference of pressure between the two ends of the tube is 0.7848 N/cm^2 and the viscosity of the liquid is 0.2 poise. Find the rate of flow of liquid through the tube. [**Ans.** 16.43 cm^3/s]

15. A sphere of diameter 3 mm falls 100 mm in 1.5 seconds in a viscous liquid. The density of the sphere is 7000 kg/m^3 and of liquid is 800 kg/m^3. Find the co-efficient of viscosity of the liquid. [**Ans.** 45.61 poise]

16. The viscosity of a liquid is determined by rotating cylinder method, in which case the inner cylinder of diameter 25 cm is stationary. The outer cylinder of diameter 25.5 cm contains the liquid upto a height of 40 cm. The clearance at the bottom of the two cylinders is 0.6 cm. The outer cylinder is rotated at 300 r.p.m. The torque registered on the torsion metre attached to the inner cylinder is 4.905 Nm. Find the viscosity of liquid. [**Ans.** .77 poise]

17. Calculate : (*a*) the pressure gradient along the flow, (*b*) the average velocity, and (*c*) the discharge for an oil of viscosity 0.02 Ns/m^2 flowing between two stationary parallel plates 1 m wide maintained 10 mm apart. The velocity midway between the plates is 2.5 m/s.
[**Ans.** (a) – 4000 N/m^2 per m, (b) 1.667 m/s, (c) .01667 m^3/s]

18. Calculate :
(*i*) the pressure gradient along the flow,
(*ii*) the average velocity, and
(*iii*) the discharge for an oil of viscosity 0.03 N s/m^2 flowing between two stationary plates which are parallel and are at 10 mm apart. Width of plates is 2 m. The velocity midway between the plates is 2.0 m/s.

19. A cylinder of 100 mm diameter, 0.15 m length and weighing 10 N slides axially in a vertical pipe of 104 mm dia. If the space between cylinder surface and pipe wall is filled with liquid of viscosity μ and the cylinder slides downwards at a velocity of 0.45 m/s, determine μ.
[**Hint.** $D = 100$ mm $= 0.1$, $L = 0.15$ m, $W = 10$ N, $D_p = 1.4$ mm $= 0.104$ m,
$V = 0.45$ m/s. Hence $t = (0.104 - 0.1)/2 = 0.002$ m.

$\mu = \dfrac{4Wt^3}{3\pi D^3 LV} = \dfrac{4 \times 10 \times 0.002^3}{3\pi \times 0.1^3 \times 0.15 \times .45}$ **= 503 × 10^{-6} N s/m^2**]

20. A liquid is pumped through a 15 cm diameter and 300 m long pipe at the rate of 20 tonnes per hour. The density of liquid is 910 kg/m^3 and kinematic viscosity = 0.002 m^2/s. Determine the power required and show that the flow is viscous.

[**Hint.** D = 15 cm = 0.15 m, L = 300 m, W = 20 tonnes/hr

= 20 × 1000 kgf/60 × 60 sec = 5.555 kgf/sec = 5.555 × 9.81 N/s.

$$Q = \frac{W}{\rho g} = \frac{5.555 \times 9.81}{910 \times 9.81} = 0.0061 \text{ m}^3\text{/s}. \quad V = \frac{Q}{A} = \frac{0.0061}{\frac{\pi}{4}(.15^2)}$$

= 0.345 m/s, v = 0.002 m^2/s.

Now $$R_e = \frac{\rho VD}{\mu} = \frac{V \times D}{\nu} = \frac{0.345 \times 0.15}{0.002} = 25.87$$

which is less than 2000. Hence flow is viscous.

$$h_f = 32\,\mu LV/\rho g D^2, \text{ where } \nu = \frac{\mu}{\rho} \quad \therefore \mu = \nu \times \rho = 0.002 \times 910 = 1.82$$

Hence, $$h_f = \frac{32 \times 1.82 \times 300 \times 0.345}{\left(910 \times 9.81 \times 0.15^2\right)} = 30$$

$\therefore$ $P = \rho g.Q.h_f/1000 = 910 \times 9.81 \times 0.0061 \times 30/1000 =$ **1.633 kW.**]

21. An oil of specific gravity 0.9 and viscosity 10 poise is flowing through a pipe of diameter 110 mm. The velocity at the centre is 2 m/s, find : (*i*) pressure gradient in the direction of flow, (*ii*) shear stress at the pipe wall ; (*iii*) Reynolds number, and (*iv*) velocity at a distance of 30 mm from the wall.

[**Hint.** ρ = 900 kg/m^3 ; μ = 10 poise = 1 N s/m^2 ; D = 110 mm = 0.11 m,

U_{max} = 2 m/s ; $\bar{u}$ = 1 m/s ; $U_{max} = \dfrac{1}{4\mu}\left(\dfrac{-dp}{dx}\right) R^2$

(*i*) $\left(\dfrac{-dp}{dx}\right) = \dfrac{4\mu \times U_{max}}{R^2} = \dfrac{4 \times 1 \times 2}{0.055^2} = 2644.6$ N/m^3 ;

(*ii*) $\tau_0 = \left(\dfrac{-dp}{dx}\right) \times \dfrac{R}{2} = 2644.6 \times \dfrac{0.055}{2} = 72.72$ N/m^2 ;

(*iii*) $R_e = \dfrac{\rho \times \bar{u} \times D}{\mu} = \dfrac{900 \times 1 \times 0.11}{1} = 99$; and

(*iv*) $u = \dfrac{1}{4\mu}\left(\dfrac{-dp}{dx}\right)(R^2 - r^2) = \dfrac{1}{4 \times 1}(2644.6)(0.055^2 - 0.025^2) = 1.586$ m/s.]

22. Determine (*i*) the pressure gradient, (*ii*) the shear stress at the two horizontal plates, (*iii*) the discharge per metre width for laminar flow of oil with a maximum velocity of 2 m/s between two plates which are 150 mm apart. Given : μ = 2.5 N s/m^2. (*Delhi University, December 2002*)

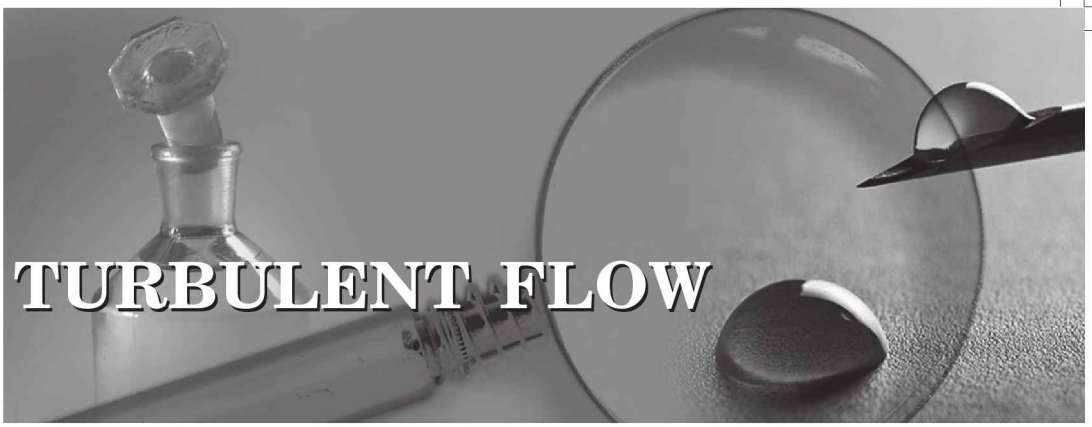
[**Hint.** U_{max} = 2 m/s, t = 150 mm = 0.15 m, μ = 2.5 N s/m^2

(*i*) $U_{max} = -\dfrac{1}{8\mu}\dfrac{dp}{dx}t^2 \quad \therefore \quad \dfrac{dp}{dx} = \dfrac{-8\mu\, U_{max}}{t^2} = \dfrac{-8 \times 2.5 \times 2}{0.15^2} =$ **– 1777.77 N/m^2.**

(*ii*) $\tau_0 = -\dfrac{1}{2}\dfrac{dp}{dx} \times t = -\dfrac{1}{2}(-1777.77) \times 0.15 =$ **133.33 N/m^2.**

(*iii*) Q = Mean velocity × Area $= \left(\dfrac{2}{3}U_{max}\right) \times (t \times 1) = \left(\dfrac{2}{3} \times 2\right) \times (0.15 \times 1) =$ **0.2 m^3/s.**]

10 CHAPTER



TURBULENT FLOW

▶ 10.1 INTRODUCTION

The laminar flow has been discussed in chapter 9. In laminar flow the fluid particles move along straight parallel path in layers or laminae, such that the paths of individual fluid particles do not cross those of neighbouring particles. Laminar flow is possible only at low velocities and when the fluid is highly viscous. But when the velocity is increased or fluid is less viscous, the fluid particles do not move in straight paths. The fluid particles move in random manner resulting in general mixing of the particles. This type of flow is called turbulent flow.

A laminar flow changes to turbulent flow when (*i*) velocity is increased or (*ii*) diameter of a pipe is increased or (*iii*) the viscosity of fluid is decreased. O. Reynold was first to demonstrate that the transition from laminar to turbulent depends not only on the mean velocity but on the quantity $\frac{\rho VD}{\mu}$. This quantity $\frac{\rho VD}{\mu}$ is a dimensionless quantity and is called Reynolds number (R_e). In case of circular pipe if $R_e < 2000$ the flow is said to be laminar and if $R_e > 4000$, the flow is said to be turbulent. If R_e lies between 2000 to 4000, the flow changes from laminar to turbulent.

▶ 10.2 REYNOLDS EXPERIMENT

The type of flow is determined from the Reynolds number *i.e.*, $\frac{\rho V \times d}{\mu}$. This was demonstrated by O. Reynold in 1883. His apparatus is shown in Fig. 10.1.

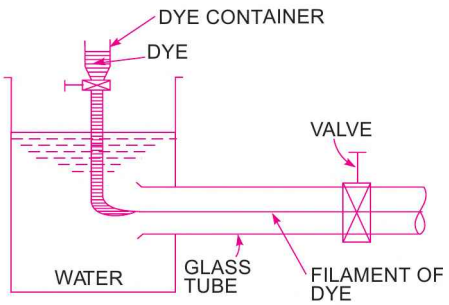


Fig. 10.1 *Reynold apparatus.*

The apparatus consists of :

(*i*) A tank containing water at constant head,

(*ii*) A small tank containing some dye,

(*iii*) A glass tube having a bell-mouthed entrance at one end and a regulating value at other ends.

The water from the tank was allowed to flow through the glass tube. The velocity of flow was varied by the regulating valve. A liquid dye having same specific weight as water was introduced into the glass tube as shown in Fig. 10.1.

The following observations were made by Reynold :

(*i*) When the velocity of flow was low, the dye filament in the glass tube was in the form of a straight line. This straight line of dye filament was parallel to the glass tube, which was the case of laminar flow as shown in Fig. 10.2 (*a*).

(*ii*) With the increase of velocity of flow, the dye-filament was no longer a straight-line but it became a wavy one as shown in Fig. 10.2 (*b*). This shows that flow is no longer laminar.

(*iii*) With further increase of velocity of flow, the wavy dye-filament broke-up and finally diffused in water as shown in Fig. 10.2 (*c*). This means that the fluid particles of the dye at this higher velocity are moving in random fashion, which shows the case of turbulent flow. Thus in case of turbulent flow the mixing of dye-filament and water is intense and flow is irregular, random and disorderly.

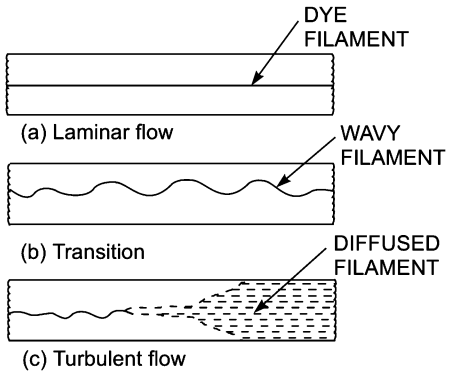


Fig. 10.2 *Different stages of filament.*

In case of laminar flow, the loss of pressure head was found to be proportional to the velocity but in case of turbulent flow, Reynold observed that loss of head is approximately proportional to the square of velocity. More exactly the loss of head, $h_f \propto V^n$, where n varies from 1.75 to 2.0

▶ 10.3 FRICTIONAL LOSS IN PIPE FLOW

When a liquid is flowing through a pipe, the velocity of the liquid layer adjacent to the pipe wall is zero. The velocity of liquid goes on increasing from the wall and thus velocity gradient and hence shear stresses are produced in the whole liquid due to viscosity. This viscous action causes loss of energy which is usually known as frictional loss.

On the basis of his experiments, William Froude gave the following laws of fluid fraction for turbulent flow.

The frictional resistance for turbulent flow is :

(*i*) proportional to V^n, where n varies from 1.5 to 2.0,

(*ii*) proportional to the density of fluid,

(*iii*) proportional to the area of surface in contact,

(*iv*) independent of pressure,

(*v*) dependent on the nature of the surface in contact.

10.3.1 Expression for Loss of Head Due to Friction in Pipes. Consider a uniform horizontal pipe, having steady flow as shown in Fig. 10.3. Let 1-1 and 2-2 are two sections of pipe.

Let p_1 = pressure intensity at section 1-1,

V_1 = velocity of flow at section 1-1,

L = length of the pipe between sections 1-1 and 2-2,

d = diameter of pipe,

f' = frictional resistance per unit wetted area per unit velocity,

h_f = loss of head due to friction,

and p_2, V_2 = are values of pressure intensity and velocity at section 2-2.

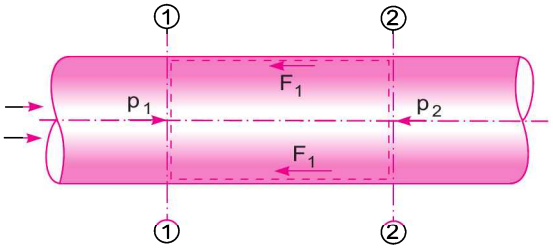


Fig. 10.3 *Uniform horizontal pipe.*

Applying Bernoulli's equations between sections 1-1 and 2-2,

Total head at 1-1 = Total head at 2-2 + loss of head due to friction between 1-1 and 2-2

or $$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + h_f$$

But $z_1 = z_2$ as pipe is horizontal

$V_1 = V_2$ as dia. of pipe is same at 1-1 and 2-2

$\therefore$ $$\frac{p_1}{\rho g} = \frac{p_2}{\rho g} + h_f \text{ or } h_f = \frac{p_1}{\rho g} - \frac{p_2}{\rho g} \quad ...(i)$$

But h_f is the head lost due to friction and hence intensity of pressure will be reduced in the direction of flow by frictional resistance.

Now frictional resistance = frictional resistance per unit wetted area per unit velocity × wetted area × velocity^2

or $$F_1 = f' \times \pi dL \times V^2 \quad [\because \text{wetted area} = \pi d \times L, \text{ velocity} = V = V_1 = V_2]$$

$$= f' \times P \times L \times V^2 \quad [\because \pi d = \text{Perimeter} = P] \ ...(ii)$$

The forces acting on the fluid between sections 1-1 and 2-2 are :

1. pressure force at section 1-1 = $p_1 \times A$

where A = Area of pipe

2. pressure force at section 2-2 = $p_2 \times A$
3. frictional force F_1 as shown in Fig. 10.3.

Resolving all forces in the horizontal direction, we have

$$p_1A - p_2A - F_1 = 0 \quad ...(10.1)$$

or $$(p_1 - p_2)A = F_1 = f' \times P \times L \times V^2 \quad [\because \text{From } (ii), F_1 = f'PLV^2]$$

or $$p_1 - p_2 = \frac{f' \times P \times L \times V^2}{A}$$

But from equation (i), $p_1 - p_2 = \rho g h_f$

Equating the value of $(p_1 - p_2)$, we get

$$\rho g h_f = \frac{f' \times P \times L \times V^2}{A}$$

or
$$h_f = \frac{f'}{\rho g} \times \frac{P}{A} \times L \times V^2 \qquad ...(iii)$$

In equation (*iii*), $\frac{P}{A} = \frac{\text{Wetted perimeter}}{\text{Area}} = \frac{\pi d}{\frac{\pi}{4} d^2} = \frac{4}{d}$

$\therefore$
$$h_f = \frac{f'}{\rho g} \times \frac{4}{d} \times L \times V^2 = \frac{f'}{\rho g} \times \frac{4LV^2}{d} \qquad ...(iv)$$

Putting $\frac{f'}{\rho} = \frac{f}{2}$, where f is known as co-efficient of friction.

Equation (*iv*), becomes as $h_f = \frac{4 \,.\, f}{2g} \cdot \frac{LV^2}{d} = \frac{4f \,.\, L \,.\, V^2}{d \times 2g}$ ...(10.2)

Equation (10.2) is known as Darcy-Weisbach equation. This equation is commonly used for finding loss of head due to friction in pipes.

Sometimes equation (10.2) is written as

$$h_f = \frac{f \,.\, L \,.\, V^2}{d \times 2g} \qquad ...(10.2A)$$

Then f is known as friction factor.

10.3.2 Expression for Co-efficient of Friction in Terms of Shear Stress. The equation (10.1) gives the forces acting on a fluid between sections 1-1 and 2-2 of Fig. 10.3 in horizontal direction as

$$p_1 A - p_2 A - F_1 = 0$$

or
$$(p_1 - p_2)A = F_1 = \text{force due to shear stress } \tau_0$$
$$= \text{shear stress} \times \text{surface area}$$
$$= \tau_0 \times \pi d \times L$$

or
$$(p_1 - p_2) \frac{\pi}{4} d^2 = \tau_0 \times \pi d \times L \qquad \left\{\because A = \frac{\pi}{4} d^2\right\}$$

Cancelling πd from both sides, we have

$$(p_1 - p_2) \frac{d}{4} = \tau_0 \times L$$

or
$$(p_1 - p_2) = \frac{4\tau_0 \times L}{d} \qquad ...(10.3)$$

Equation (10.2) can be written as $h_f = \frac{p_1 - p_2}{\rho g} = \frac{4f \,.\, L \,.\, V^2}{d \times 2g}$

or $$(p_1 - p_2) = \frac{4f \,.\, L \,.\, V^2}{d \times 2g} \times \rho g \qquad ...(10.4)$$

Equating the value of $(p_1 - p_2)$ in equations (10.3) and (10.4),

$$\frac{4\tau_0 \times L}{d} = \frac{4f \,.\, L \,.\, V^2}{d \times 2g} \times \rho g$$

or $$\tau_0 = \frac{fV^2 \times \rho g}{2g} = \frac{fV^2}{2g} \times \rho g$$

or $$\tau_0 = f \frac{\rho V^2}{2} \qquad ...(10.5)$$

$\therefore$ $$f = \frac{2\tau_0}{\rho V^2}. \qquad ...(10.6)$$

▶ 10.4 SHEAR STRESS IN TURBULENT FLOW

The shear stress in viscous flow is given by Newton's law of viscosity as

$$\tau_v = \mu \frac{du}{dy}, \qquad \text{where } \tau_v = \text{shear stress due to viscosity.}$$

Similar to the expression for viscous shear, J. Boussinesq expressed the turbulent shear in mathematical form as

$$\tau_t = \eta \frac{d\overline{u}}{dy} \qquad ...(10.7)$$

where τ_t = shear stress due to turbulence
η = eddy viscosity
$\overline{u}$ = average velocity at a distance y from boundary.

The ratio of η (eddy viscosity) and ρ (mass density) is known as kinematic eddy viscosity and is denoted by ε (epsilon). Mathematically it is written as

$$\varepsilon = \frac{\eta}{\rho} \qquad ...(10.8)$$

If the shear stress due to viscous flow is also considered, then the total shear stress becomes as

$$\tau = \tau_v + \tau_t = \mu \frac{du}{dy} + \eta \frac{d\,\overline{u}}{dy} \qquad ...(10.9)$$

The value of $\eta = 0$ for laminar flow. For other cases the value of η may be several thousand times the value of μ. To find shear stress in turbulent flow, equation (10.7) given by Boussinesq is used. But as the value of η (eddy viscosity) cannot be predicted, this equation is having limited use.

10.4.1 Reynolds Expression for Turbulent Shear Stress. Reynolds in 1886 developed an expression for turbulent shear stress between two layers of a fluid at a small distance apart, which is given as

$$\tau = \rho u' v' \qquad ...(10.10)$$

where u', v' = fluctuating component of velocity in the direction of x and y due to turbulence.

As u' and v' are varying and hence τ will also vary. Hence to find the shear stress, the time average on both the sides of the equation (10.10) is taken. Then equation (10.10) becomes as

$$\bar{\tau} = \overline{\rho u' v'} \qquad ...(10.11)$$

The turbulent shear stress given by equation (10.11) is known as Reynold stress.

10.4.2 Prandtl Mixing Length Theory for Turbulent Shear Stress. In equation (10.11), the turbulent shear stress can only be calculated if the value of $u' v'$ is known. But it is very difficult to measure $\overline{u' v'}$. To overcome this difficulty, *L.* Prandtl in 1925, presented a mixing length hypothesis which can be used to express turbulent shear stress in terms of measurable quantities.

According to Prandtl, the mixing length l, is that distance between two layers in the transverse direction such that the lumps of fluid particles from one layer could reach the other layer and the particles are mixed in the other layer in such a way that the momentum of the particles in the direction of x is same. He also assumed that the velocity fluctuation in the x-direction u' is related to the mixing length l as

$$u' = l \frac{du}{dy}$$

and v', the fluctuation component of velocity in y-direction is of the same order of magnitude as u' and hence

$$v' = l \frac{du}{dy}$$

Now $\overline{u' \times v'}$ becomes as $\overline{u' v'} = \left(l \frac{du}{dy} \right) \times \left(l \frac{du}{dy} \right) = l^2 \left(\frac{du}{dy} \right)^2$

Substituting the value of $\overline{u' v'}$ in equation (10.11), we get the expression for shear stress in turbulent flow due to Prandtl as

$$\bar{\tau} = \rho l^2 \left(\frac{du}{dy} \right)^2 \qquad ...(10.12)$$

Thus the total shear stress at any point in turbulent flow is the sum of shear stress due to viscous shear and turbulent shear and can be written as

$$\bar{\tau} = \mu \frac{du}{dy} + \rho l^2 \left(\frac{du}{dy} \right)^2 \qquad ...(10.13)$$

But the viscous shear stress is negligible except near the boundary. Equation (10.13) is used for most of turbulent fluid flow problems for determining shear stress in turbulent flow.

▶ 10.5 VELOCITY DISTRIBUTION IN TURBULENT FLOW IN PIPES

In case of turbulent flow, the total shear stress at any point is the sum of viscous shear stress and turbulent shear stress. Also the viscous shear stress is negligible except near the boundary. Hence it can be assumed that the shear stress in turbulent flow is given by equation (10.12). From this equation, the velocity distribution can be obtained if the relation between l, the mixing length and y is known. Prandtl assumed that the mixing length, l is a linear function of the distance y from the pipe wall *i.e.*, $l = ky$, where k is a constant, known as Karman constant and = 0.4.

Substituting the value of l in equation (10.12), we get

$$\bar{\tau} \text{ or } \tau = \rho \times (ky)^2 \times \left(\frac{du}{dy}\right)^2$$

or $$\tau = \rho k^2 y^2 \left(\frac{du}{dy}\right)^2 \text{ or } \left(\frac{du}{dy}\right)^2 = \tau/\rho k^2 y^2$$

or $$\frac{du}{dy} = \sqrt{\frac{\tau}{\rho k^2 y^2}} = \frac{1}{ky}\sqrt{\frac{\tau}{\rho}} \qquad ...(10.14)$$

For small values of y that is very close to the boundary of the pipe, Prandtl assumed shear stress τ to be constant and approximately equal to τ_0 which presents the turbulent shear stress at the pipe boundary. Substituting $\tau = \tau_0$ in equation (10.14), we get

$$\frac{du}{dy} = \frac{1}{ky}\sqrt{\frac{\tau_0}{\rho}} \qquad ...(10.15)$$

In equation (10.15), $\sqrt{\frac{\tau_0}{\rho}}$ has the dimensions $\sqrt{\frac{ML^{-1}T^{-2}}{ML^{-3}}} = \sqrt{\frac{L^2}{T^2}} = \frac{L}{T}$. But $\frac{L}{T}$ is velocity and hence $\sqrt{\frac{\tau_0}{\rho}}$ has the dimension of velocity, which is known as shear velocity and is denoted by u_*.

Thus $\sqrt{\frac{\tau_0}{\rho}} = u_*$, then equation (10.15) becomes $\frac{du}{dy} = \frac{1}{ky} u_*$.

For a given case of turbulent flow, u_* is constant. Hence integrating above equation, we get

$$u = \frac{u_*}{k} \log_e y + C \qquad ...(10.16)$$

where C = constant of integration.

Equation (10.16) shows that in turbulent flow, the velocity varies directly with the logarithm of the distance from the boundary or in other words the velocity distribution in turbulent flow is logarithmic in nature. To determine the constant of integration, C the boundary condition that at $y = R$ (radius of pipe), $u = u_{max}$ is substituted in equation (10.16).

Hence $$u_{max} = \frac{u_*}{k} \log_e R + C \quad \therefore \quad C = u_{max} - \frac{u_*}{k} \log_e R$$

Substituting the value of C in equation (10.16), we get

$$u = \frac{u_*}{k} \log_e y + u_{max} - \frac{u_*}{k} \log_e R = u_{max} + \frac{u_*}{k} (\log_e y - \log_e R)$$

$$= u_{max} + \frac{u_*}{0.4} \log_e (y/R) \qquad [\because k = 0.4 = \text{Karman constant}]$$

$$= u_{max} + 2.5\, u_* \log_e (y/R) \qquad ...(10.17)$$

Equation (10.17) is called 'Prandtl's universal velocity distribution equation for turbulent flow in pipes. This equation is applicable to smooth as well as rough pipe boundaries. Equation (10.17) is also written as

$$u_{max} - u = -2.5\, u_* \log_e (y/R) = 2.5\, u_* \log_e (R/y)$$

Dividing by u_*, we get

$$\frac{u_{max} - u}{u_*} = 2.5 \log_e (R/y) = 2.5 \times 2.3 \log_{10} (R/y) \quad [\because \log_e (R/y) = 2.3 \log_{10} (R/y)]$$

or
$$\frac{u_{max} - u}{u_*} = 5.75 \log_{10} (R/y) \qquad ...(10.18)$$

In equation (10.18), the difference between the maximum velocity u_{max}, and local velocity u at any point *i.e.*, $(u_{max} - u)$ is known as '**velocity defect**'.

10.5.1 Hydrodynamically Smooth and Rough Boundaries. Let k is the average height of the irregularities projecting from the surface of a boundary as shown in Fig. 10.4. If the value of k is large for a boundary then the boundary is called rough boundary and if the value of k is less, then boundary is known as smooth boundary, in general. This is the classification of rough and smooth boundary based on boundary characteristics. But for proper classification, the flow and fluid characteristics are also to be considered.

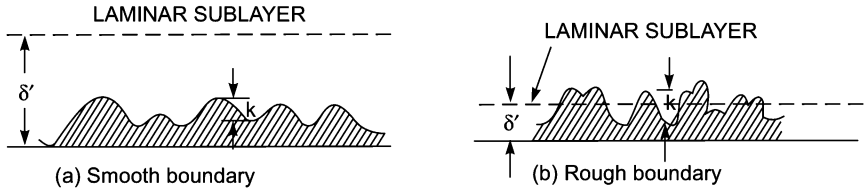


Fig. 10.4 *Smooth and rough boundaries.*

For turbulent flow analysis along a boundary, the flow is divided in two portions. The first portion consists of a thin layer of fluid in the immediate neighbourhood of the boundary, where viscous shear stress predominates while the shear stress due to turbulence is negligible. This portion is known as laminar sub-layer. The height upto which the effect of viscosity predominates in this zone is denoted by δ'. The second portion of flow, where shear stress due to turbulence are large as compared to viscous stress is known as turbulent zone.

If the average height k of the irregularities, projecting from the surface of a boundary is much less than δ', the thickness of laminar sub-layer as shown in Fig. 10.4 (*a*), the boundary is called smooth boundary. This is because, outside the laminar sub-layer the flow is turbulent and eddies of various size present in turbulent flow try to penetrate the laminar sub-layer and reach the surface of the boundary. But due to great thickness of laminar sub-layer the eddies are unable to reach the surface irregularities and hence the boundary behaves as a smooth boundary. This type of boundary is called hydrodynamically smooth boundary.

Now, if the Reynolds number of the flow is increased then the thickness of laminar sub-layer will decrease. If the thickness of laminar sub-layer becomes much smaller than the average height k of irregularities of the surface as shown in Fig. 10.4 (*b*), the boundary will act as rough boundary. This is because the irregularities of the surface are above the laminar sub-layer and the eddies present in turbulent zone will come in contact with the irregularities of the surface and lot of energy will be lost. Such a boundary is called hydrodynamically rough boundary.

From Nikuradse's experiment :

1. If $\frac{k}{\delta'}$ is less than 0.25 or $\frac{k}{\delta'} < 0.25$, the boundary is called smooth boundary.

2. If $\frac{k}{\delta'}$ is greater than 6.0, the boundary is rough,

3. If $0.25 < \left(\frac{k}{\delta'}\right) < 6.0$, the boundary is in transition.

In terms of roughness Reynolds number $\frac{u_* k}{\nu}$:

1. If $\frac{u_* k}{\nu} < 4$, boundary is considered smooth,

2. If $\frac{u_* k}{\nu}$ lies between 4 and 100, boundary is in transition stage, and

3. If $\frac{u_* k}{\nu} > 100$, the boundary is rough.

10.5.2 Velocity Distribution for Turbulent Flow in Smooth Pipes. The velocity distribution for turbulent flow in smooth or rough pipe is given by equation (10.16) as

$$u = \frac{u_*}{k} \log_e y + C$$

It may be seen that at $y = 0$, the velocity u at wall is $-\infty$. This means that velocity u is positive at some distance far away from the wall and $-\infty$ (minus infinity) at the wall. Hence at some finite distance from wall, the velocity will be equal to zero. Let this distance from pipe wall is y'. Now the constant C is determined from the boundary condition *i.e.,* at $y = y'$, $u = 0$. Hence above equation becomes as

$$0 = \frac{u_*}{k} \log_e y' + C \text{ or } C = -\frac{u_*}{k} \log_e y'$$

Substituting the value of C in the above equation, we get

$$u = \frac{u_*}{k} \log_e y - \frac{u_*}{k} \log_e y' = \frac{u_*}{k} \log_e (y/y')$$

Substituting the value of $k = 0.4$, we get

$$u = \frac{u_*}{0.4} \log_e (y/y') = 2.5\, u_* \log_e (y/y')$$

$$\frac{u}{u_*} = 2.5 \times 2.3 \log_{10} (y/y') \qquad [\because \log_e (y/y') = 2.3 \log_{10} (y/y')]$$

or

$$\frac{u}{u_*} = 5.75 \log_{10} (y/y') \qquad \ldots(10.19)$$

For the smooth boundary, there exists a laminar sub-layer as shown in Fig. 10.4 (*a*). The velocity distribution in the laminar sub-layer is parabolic in nature. Thus in the laminar sub-layer, logarithmic velocity distribution does not hold good. Thus it can be assumed that y' is proportional to δ', where δ' is the thickness of laminar sub-layer. From Nikuradse's experiment the value of y' is given as

$$y' = \frac{\delta'}{107}$$

where $\delta' = \frac{11.6\nu}{u_*}$, where ν = kinematic viscosity of fluid.

$$\therefore \qquad y' = \frac{11.6\nu}{u_*} \times \frac{1}{107} = \frac{0.108\nu}{u_*}$$

Substituting this value of y'in equation (10.19), we obtain

$$\frac{u}{u_*} = 5.75 \log_{10} \left(\frac{y}{\frac{.108\nu}{u_*}} \right)$$

$$= 5.75 \log_{10} \left(\frac{y\, u^*}{.108\, \nu} \right) = 5.75 \log_{10} \left(\frac{u_* y}{\nu} \times 9.259 \right)$$

$$= 5.75 \log_{10} \frac{u_* y}{\nu} + 5.75 \log_{10} 9.259 \qquad \left[\because \frac{1}{0.108} = 9.259 \right]$$

$$= 5.75 \log_{10} \frac{u_* y}{\nu} + 5.55 \qquad \text{...(10.20)}$$

10.5.3 Velocity Distribution for Turbulent Flow in Rough Pipes. In case of rough boundaries, the thickness of laminar sub-layer is very small as shown in Fig. 10.4 (*b*). The surface irregularities are above the laminar sub-layer and hence the laminar sub-layer is completely destroyed. Thus y' can be considered proportional to the height of protrusions k. Nikuradse's experiment shows the value of y' for pipes coated with uniform sand (rough pipes) as $y' = \frac{k}{30}$.

Substituting this value of y' in equation (10.19), we get

$$\frac{u}{u_*} = 5.75 \log_{10} \left(\frac{y}{k/30} \right) = 5.75 \left[\log_{10} (y/k) \times 30 \right]$$

$$= 5.75 \log_{10} (y/k) + 5.75 \log_{10} (30.0) = 5.75 \log_{10} (y/k) + 8.5 \quad \text{...(10.21)}$$

Problem 10.1 *A pipe-line carrying water has average height of irregularities projecting from the surface of the boundary of the pipe as 0.15 mm. What type of boundary is it ? The shear stress developed is 4.9 N/m^2. The kinematic viscosity of water is .01 stokes.*

Solution. Given :

Average height of irregularities, $k = 0.15 \text{ mm} = 0.15 \times 10^{-3} \text{ m}$

Shear stress developed, $\tau_0 = 4.9 \text{ N/m}^2$

Kinematic viscosity, $\nu = 0.01 \text{ stokes} = .01 \text{ cm}^2/\text{s} = .01 \times 10^{-4} \text{ m}^2/\text{s}$

Density of water, $\rho = 1000 \text{ kg/m}^3$

Shear velocity, $u_* = \sqrt{\tau_0 / \rho} = \sqrt{\frac{4.9}{1000}} = \sqrt{0.0049} = 0.07 \text{ m/s}$

Roughness Reynold number $= \frac{u_* k}{\nu} = \frac{0.07 \times 0.15 \times 10^{-3}}{.01 \times 10^{-4}} = 10.5.$

Since $\frac{u_* k}{\nu}$ lies between 4 and 100 and hence pipe surface behaves as in transition.

Problem 10.2 *A rough pipe is of diameter 8.0 cm. The velocity at a point 3.0 cm from wall is 30% more than the velocity at a point 1 cm from pipe wall. Determine the average height of the roughness.*

Solution. Given :

Dia. of rough pipe, $D = 8$ cm $= .08$ m

Let velocity of flow at 1 cm from pipe wall $= u$

Then velocity of flow at 3 cm from pipe wall $= 1.3\ u$

The velocity distribution for rough pipe is given by equation (10.21) as

$$\frac{u}{u_*} = 5.75 \log_{10} (y/k) + 8.5, \text{ where } k = \text{height of roughness.}$$

For a point, 1 cm from pipe wall, we have

$$\frac{u}{u_*} = 5.75 \log_{10} (1.0/k) + 8.5 \qquad ...(i)$$

For a point, 3 cm from pipe wall, velocity is $1.3\ u$ and hence

$$\frac{1.3u}{u_*} = 5.75 \log_{10} (3.0/k) + 8.5 \qquad ...(ii)$$

Dividing (ii) by (i), we get $1.3 = \frac{5.75 \log_{10}(3.0 / k) + 8.5}{5.75 \log_{10}(1 / k) + 8.5}$

or $1.3[5.75 \log_{10} (1/k) + 8.5] = 5.75 \log_{10} (3.0/k) + 8.5$

or $7.475 \log_{10} (1/k) + 11.05 = 5.75 \log_{10} (3.0/k) + 8.5$

or $7.475 \log_{10} (1/k) - 5.75 \log_{10} (3/k) = 8.5 - 11.05 = -2.55$

or $7.475 [\log_{10} 1.0 - \log_{10} k] - 5.75 [\log_{10} 3.0 - \log_{10} k] = -2.55$

or $7.475 [0 - \log_{10} k] - 5.75 [.4771 - \log_{10} k] = -2.55$

or $-7.475 \log_{10} k - 2.7433 + 5.75 \log_{10} k = -2.55$

or $-1.725 \log_{10} k = 2.7433 - 2.55 = 0.1933$

or $$\log_{10} k = \frac{0.1933}{-1.725} = -0.1120 = \bar{1}.888$$

$$k = \mathbf{.7726\ cm.\ Ans.}$$

Problem 10.3 *A smooth pipe of diameter 80 mm and 800 m long carries water at the rate of 0.480 m^3/minute. Calculate the loss of head, wall shearing stress, centre line velocity, velocity and shear stress at 30 mm from pipe wall. Also calculate the thickness of laminar sub-layer. Take kinematic viscosity of water as 0.015 stokes. Take the value of co-efficient of friction 'f' from the relation given as*

$$f = \frac{.0791}{(R_e)^{1/4}}, \text{ where } R_e = \text{Reynolds number.}$$

Solution. Given :

Dia. of smooth pipe, d = 80 mm = .08 m

Length of pipe, L = 800 m

Discharge, $Q = 0.048 \text{ m}^3/\text{minute} = \frac{0.48}{60} = .008 \text{ m}^3/\text{s}$

Kinematic viscosity, $\nu = .015 \text{ stokes} = .015 \times 10^{-4} \text{ m}^2/\text{s}$ [Stokes = cm^2/s]

Density of water, $\rho = 1000 \text{ kg/m}^3$

Mean velocity, $V = \frac{Q}{\text{Area}} = \frac{0.008}{\frac{\pi}{4}(.08)^2} = 1.591 \text{ m/s}$

$\therefore$ Reynolds number, $R_e = \frac{V \times d}{\nu} = \frac{1.591 \times 0.08}{.015 \times 10^{-4}} = 8.485 \times 10^4$

As the Reynolds number is more than 4000, the flow is turbulent.

Now the value of 'f' is given by $f = \frac{.0791}{R_e^{1/4}} = \frac{.0791}{\left(8.485 \times 10^4\right)^{1/4}} = .004636$

(*i*) Head lost is given by equation (10.2) as

$$h_f = \frac{4 \,.\, f \,.\, L \,.\, V^2}{d \times 2g} = \frac{4 \times .004636 \times 800 \times 1.591^2}{.08 \times 2 \times 9.81} = \textbf{23.42 m. Ans.}$$

(*ii*) Wall shearing stress, τ_0 is given by equation (10.5) as

$$\tau_0 = \frac{f\rho V^2}{2} = .004636 \times \frac{1000}{2} \times 1.591^2 = \textbf{5.866 N/m}^2\textbf{. Ans.}$$

(*iii*) Centre-line velocity, u_{max} for smooth pipe is given by equation (10.20) as

$$\frac{u}{u_*} = 5.75 \log_{10} \frac{u_* y}{\nu} + 5.55 \qquad ...(i)$$

where u_* is shear velocity and $= \sqrt{\frac{\tau_0}{\rho}} = \sqrt{\frac{5.866}{1000}} = 0.0765 \text{ m/s}$

The velocity will be maximum when $y = \frac{d}{2} = \frac{.08}{2} = .04$ m.

Hence at y = .04 m, $u = u_{max.}$ Substituting these values in (*i*), we get

$$\frac{u_{max}}{.0765} = 5.75 \log_{10} \frac{0.0765 \times .04}{.015 \times 10^{-4}} + 5.55$$

$$= 5.75 \log_{10} 2040 + 5.55$$

$$= 5.75 \times 3.309 + 5.55 = 19.03 + 5.55 = 24.58$$

$\therefore$ $u_{max} = .0765 \times 24.58 = \textbf{1.88 m/s. Ans.}$

(*iv*) The shear stress, τ at any point is given by

$$\tau = -\frac{\partial p}{\partial x} \frac{r}{2} \qquad ...(A)$$

where r = distance from centre of pipe
and hence shear stress at pipe wall where $r = R$ is

$$\tau_0 = -\frac{\partial p}{\partial x}\frac{R}{2} \qquad ...(B)$$

Dividing equation (A) by equation (B), we get

$$\frac{\tau}{\tau_0} = \frac{r}{R}$$

$\therefore$ Shear stress $$\tau = \frac{\tau_0\, r}{R}$$

A point 30 mm from pipe wall is having $r = 4 - 3 = 1$ cm $= .01$ m

$$\therefore \qquad \tau \text{ at } (r = .01 \text{ m}) = \frac{\tau_0 \times .01}{.04} = \frac{5.866}{4} = \mathbf{1.4665\ N/m^2.\ Ans.}$$

Velocity at a point 3 cm from pipe wall means $y = 3$ cm $= .03$ m

and is given by equation (10.20) as $\frac{u}{u_*} = 5.75 \log_{10} \frac{u_* y}{\nu} + 5.55$, where $u_* = .0765$, $y = .03$

$$\therefore \qquad \frac{u}{.0765} = 5.75 \log_{10} \frac{.0765 \times .03}{.015 \times 10^{-4}} + 5.55$$

$$= 5.75 \log_{10} 1530 + 5.55 = 23.86$$

$$\therefore \qquad u = 0.0765 \times 23.86 = \mathbf{1.825\ m/s.\ Ans.}$$

(v) Thickness of laminar sub-layer is given by

$$\delta' = \frac{11.6 \times \nu}{u_*} = \frac{11.6 \times .015 \times 10^{-4}}{.0765} = 2.274 \times 10^{-4} \text{ m}$$

$$= 2.274 \times 10^{-2} \text{ cm} = \mathbf{.02274\ cm.\ Ans.}$$

Problem 10.4 *Determine the wall shearing stress in a pipe of diameter 100 mm which carries water. The velocities at the pipe centre and 30 mm from the pipe centre are 2 m/s and 1.5 m/s respectively. The flow in pipe is given as turbulent.*

Solution. Given :

Dia. of pipe, $D = 100$ mm $= 0.10$ m

$\therefore$ Radius, $$R = \frac{0.10}{2} = 0.05 \text{ m}$$

Velocity at centre, $u_{max} = 2$ m/s

Velocity at 30 mm or 0.03 m from centre = 1.5 m/s

$\therefore$ Velocity (at $r = 0.03$ m), $u = 1.5$ m/s

Let the wall shearing stress $= \tau_0$

For turbulent flow, the velocity distribution in terms of centre line velocity (u_{max}) is given by equation (10.18) as

$$\frac{u_{max} - u}{u_*} = 5.75 \log_{10} \left(\frac{R}{y}\right)$$

where $u = 1.5$ m/s at $y = (R - r) = 0.05 - 0.03 = .02$ m

$$\therefore \quad \frac{2.0 - 1.5}{u_*} = 5.75 \log_{10} \frac{.05}{.02} = 2.288 \text{ or } \frac{0.5}{u_*} = 2.288$$

$$\therefore \quad u_* = \frac{0.5}{2.288} = 0.2185 \text{ m/s}$$

Using the relation $u_* = \sqrt{\tau_0 / \rho}$, where ρ *for* water = 1000 kg/m^3

$$\therefore \quad 0.2185 = \sqrt{\frac{\tau_0}{1000}} \text{ or } \frac{\tau_0}{1000} = 0.2185^2 = 0.0477$$

or $$\tau_0 = 0.0477 \times 1000 = \mathbf{47.676\ N/m^2.\ Ans.}$$

10.5.4 Velocity Distribution for Turbulent Flow in Terms of Average Velocity. The average velocity $\overline{U}$, through the pipe is obtained by first finding the total discharge Q and then dividing the total discharge by the area of the pipe.

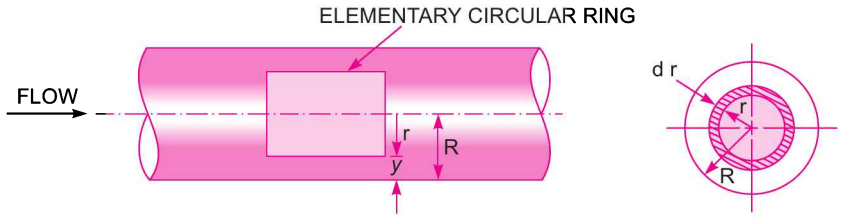


Fig. 10.5 *Average velocity for turbulent flow.*

Consider an elementary circular ring of radius 'r' and thickness dr as shown in Fig. 10.5. The distance of the ring from pipe wall is $y = (R - r)$, where R = radius of pipe.

Then the discharge, dQ, through the ring is given by

$$dQ = \text{area of ring} \times \text{velocity}$$
$$= 2\pi r dr \times u = u \times 2\pi r dr$$

Total discharge, $$Q = \int dQ = \int_0^R u \times 2\pi r dr \qquad ...(10.22)$$

(*a*) **For smooth pipes.** For smooth pipes, the velocity distribution is given by equation (10.20) as

$$\frac{u}{u_*} = 5.75 \log_{10} \frac{u_* y}{\nu} + 5.5$$

or $$u = \left[5.75 \log_{10} \frac{u_* y}{\nu} + 5.5\right] \times u_*$$

But $$y = (R - r)$$

$$\therefore \quad u = \left[5.75 \log_{10} \frac{u_* (R - r)}{\nu} + 5.5\right] \times u_*$$

Substituting the value of u in equation (10.22), we get

$$Q = \int_0^R \left[5.75 \log_{10} \frac{u_* (R - r)}{\nu} + 5.5\right] u_* \times 2\pi r dr$$

$\therefore$ Average velocity, $\overline{U} = \dfrac{Q}{\text{Area}} = \dfrac{Q}{\pi R^2}$

$$= \frac{1}{\pi R^2} \int_0^R \left[5.75 \log_{10} \frac{u_*(R-r)}{\nu} + 5.5 \right] u_* \, 2\pi r dr$$

Integration of the above equation and subsequent simplification gives the average velocity for turbulent flow in smooth pipes as

$$\frac{\overline{U}}{u_*} = 5.75 \log_{10} \frac{u_* R}{\nu} + 1.75 \qquad ...(10.23)$$

(*b*) **For rough pipes.** For rough pipes, the velocity at any point in turbulent flow is given by equation (10.21) as

$$\frac{u}{u_*} = 5.75 \log_{10} (y/k) + 8.5$$

But $y = (R - r)$

$\therefore$ $$\frac{u}{u_*} = 5.75 \log_{10} \left(\frac{R-r}{k} \right) + 8.5$$

or $$u = u_* \left[5.75 \log_{10} \left(\frac{R-r}{k} \right) + 8.5 \right]$$

Substituting the value of u in equation (10.22), we get

$$Q = \int_0^R u_* \left[5.75 \log_{10} \left(\frac{R-r}{k} \right) + 8.5 \right] 2\pi r dr$$

$\therefore$ Average velocity, $$\overline{U} = \frac{Q}{\pi R^2} = \frac{\int_0^R u_* \left[5.75 \log_{10} \left(\frac{R-r}{k} \right) + 8.5 \right] 2\pi r dr}{\pi R^2}$$

Integration of the above equation and subsequent simplification will give the following relation for average velocity, $\overline{U}$ for turbulent flow in rough pipe as

$$\frac{\overline{U}}{u_*} = 5.75 \log_{10} \frac{R}{k} + 4.75 \qquad ...(10.24)$$

(*c*) **Difference of the velocity at any point and average velocity for smooth and rough pipes.** The velocity at any point for turbulent flow for smooth pipes is given by equation (10.20) as

$$\frac{u}{u_*} = 5.75 \log_{10} \frac{u_*(R-r)}{\nu} + 5.5 \qquad [\because y = R - r]$$

and the average velocity is given by equation (10.23) as

$$\frac{\overline{U}}{u_*} = 5.75 \log_{10} \frac{u_* R}{\nu} + 1.75$$

$\therefore$ Difference of velocity u and $\overline{U}$ for smooth pipe is obtained as

$$\frac{u}{u_*} - \frac{\overline{U}}{u_*} = \left[5.75 \log_{10} \frac{u_*(R-r)}{\nu} + 5.5\right] - \left[5.75 \log_{10} \frac{u_* R}{\nu} + 1.75\right]$$

or
$$\frac{u - \overline{U}}{u_*} = 5.75 \left[\log_{10} \frac{u_*(R-r)}{\nu} - \log_{10} \frac{u_* R}{\nu}\right] + 5.5 - 1.75$$

$$= 5.75 \log_{10} \left[\frac{u_*(R-r)}{\nu} \div \frac{u_* R}{\nu}\right] + 3.75$$

$$= 5.75 \log_{10} \left(\frac{R-r}{\nu}\right) + 3.75$$

$$= 5.75 \log_{10} (y/R) + 3.75 \qquad \text{...(10.25)} \ [\because R - r = y]$$

Similarly the velocity, u at any point for rough pipe is given by equation (10.21) as

$$\frac{u}{u_*} = 5.75 \log_{10} (y/k) + 8.5$$

and average velocity is given by equation (10.24) as

$$\frac{\overline{U}}{u_*} = 5.75 \log_{10} (R/k) + 4.75$$

$\therefore$ Difference of velocity u and $\overline{U}$ for rough pipe is given by

$$\frac{u}{u_*} - \frac{\overline{U}}{u_*} = [5.75 \log_{10} (y/k) + 8.5] - [5.75 \log_{10} (R/k) + 4.75]$$

$$= 5.75 \log_{10} [(y/k) \div (R/k)] + 8.5 - 4.75$$

or
$$\frac{u - \overline{U}}{u_*} = 5.75 \log_{10} (y/R) + 3.75 \qquad \text{...(10.26)}$$

Equations (10.25) and (10.26) are the same. This shows that the difference of velocity at any point and the average velocity will be the same in case of smooth as well as rough pipes.

Problem 10.5 *Determine the distance from the pipe wall at which the local velocity is equal to the average velocity for turbulent flow in pipes.*

Solution. Given :

Local velocity at a point = average velocity

or
$$u = \overline{U}$$

For a smooth or rough pipe, the difference of velocity at any point and average velocity is given by equation (10.25) or equation (10.26) as

$$\frac{u - \overline{U}}{u_*} = 5.75 \log_{10} (y/R) + 3.75$$

Substituting the given condition *i.e.*, $u = \overline{U}$, we get

$$\frac{\overline{U} - \overline{U}}{u_*} = 0 = 5.75 \log_{10} (y/R) + 3.75 \quad \text{or} \quad 5.75 \log_{10} (y/R) = 3.75$$

or $$\log_{10}(y/R) = -\frac{3.75}{5.75} = -0.6521 = -\bar{1}.3479$$

$\therefore$ $$y/R = 0.22279 \simeq 0.2228 \text{ or } y = \textbf{.2228 R. Ans.}$$

Problem 10.6 *For turbulent flow in a pipe of diameter 300 mm, find the discharge when the centre-line velocity is 2.0 m/s and the velocity at a point 100 mm from the centre as measured by pitot-tube is 1.6 m/s.*

Solution. Given :

Dia. of pipe, $D = 300 \text{ mm} = 0.3 \text{ m}$

$\therefore$ Radius, $R = \frac{0.3}{2} = 0.15 \text{ m}$

Velocity at centre, $u_{max} = 2.0 \text{ m/s}$

Velocity (at $r = 100 \text{ mm} = 0.1 \text{ m}$), $u = 1.6 \text{ m/s}$

Now $y = R - r = 0.15 - 0.10 = 0.05 \text{ m}$

$\therefore$ Velocity (at $r = 0.1 \text{ m}$ or at $y = 0.05 \text{ m}$), $u = 1.6 \text{ m/s}$

The velocity in terms of centre-line velocity is given by equation (10.18) as

$$\frac{u_{max} - u}{u_*} = 5.75 \log_{10}(R/y)$$

Substituting the values, we get $\frac{2.0 - 1.6}{u_*} = 5.75 \log_{10} \frac{.15}{.05}$ $\left[\because \begin{matrix} y = .05 \text{ m} \\ R = 0.15 \text{ m} \end{matrix}\right]$

$$= 5.75 \log_{10} 3.0 = 2.7434$$

or $$\frac{0.4}{u_*} = 2.7434$$

$\therefore$ $$u_* = \frac{0.4}{2.7434} = 0.1458 \text{ m/s} \qquad ...(i)$$

Using equation (10.26) which gives relation between velocity at any point and average velocity, we have

$$\frac{u - \overline{U}}{u_*} = 5.75 \log_{10}(y/R) + 3.75$$

at $y = R$, velocity u becomes $= u_{max}$

$\therefore$ $$\frac{u_{max} - \overline{U}}{u_*} = 5.75 \log_{10}(R/R) + 3.75 = 5.75 \times 0 + 3.75 = 3.75$$

But $u_{max} = 2.0$ and u_* from $(i) = 0.1458$

$\therefore$ $$\frac{2.0 - \overline{U}}{0.1458} = 3.75$$

or $$\overline{U} = 2.0 - .1458 \times 3.75 = 2.0 - 0.5467 = 1.4533 \text{ m/s}$$

$\therefore$ Discharge, Q = Area $\times$ average velocity

$$= \frac{\pi}{4} D^2 \times \overline{U} = \frac{\pi}{4} (0.3)^2 \times 1.4533 = \mathbf{0.1027\ m^3/s.\ Ans.}$$

10.5.5 Velocity Distribution for Turbulent Flow in Smooth Pipes by Power Law. The velocity distribution for turbulent flow as given by equations (10.18), (10.20) and (10.21) are logarithmic in nature. These equations are not convenient to use. Nikuradse carried out experiments for different Reynolds number to determine the velocity distribution law in smooth pipes. He expressed the velocity distribution in exponential form as

$$\frac{u}{u_{max}} = (y/R)^{1/n} \qquad ...(10.27)$$

where exponent $\frac{1}{n}$ depends on Reynolds number

The value of $\left(\frac{1}{n}\right)$ decreases, with increasing Reynolds number.

For $R_e = 4 \times 10^3, \quad \frac{1}{n} = \frac{1}{6}$

$R_e = 1.1 \times 10^5, \quad \frac{1}{n} = \frac{1}{7}$

$R_e \geq 2 \times 10^6, \quad \frac{1}{n} = \frac{1}{10}$

Thus if $\frac{1}{n} = \frac{1}{7}$, the velocity distribution law becomes as

$$\frac{u}{u_{max}} = \left(\frac{y}{R}\right)^{1/7} \qquad ...(10.28)$$

Equation (10.28) is known as 1/7th power law of velocity distribution for smooth pipes.

▶ 10.6 RESISTANCE OF SMOOTH AND ROUGH PIPES

The loss of head, due to friction in pipes is given by equation (10.2) as

$$h_f = \frac{4 . f . L . V^2}{d \times 2g}$$

In this equation, the value of co-efficient of friction, f should be known accurately for predicting the loss of head due to friction in pipes. On the basis of dimensional analysis, it can be shown that the pressure loss in a straight pipe of diameter D, length L, roughness k, average velocity of flow $\overline{U}$, viscosity and density of fluid μ and ρ is

$$\Delta p = \frac{\rho \overline{U}^2}{2} \phi \left[R_e, \frac{k}{D}, \frac{L}{D} \right] \quad \text{or} \quad \frac{\Delta p}{\frac{\rho \overline{U}^2}{2}} = \phi \left[R_e, \frac{k}{D}, \frac{L}{D} \right]$$

Experimentally it was found that pressure drop is a function of $\frac{L}{D}$ to the first power and hence

$$\frac{\Delta p}{\frac{\rho \overline{U}^2}{2}} = \frac{L}{D} \phi \left[R_e, \frac{k}{D} \right] \quad \text{or} \quad \frac{\Delta p \times D}{L \frac{\rho \overline{U}^2}{2}} = \phi \left[R_e, \frac{k}{D} \right]$$

The term of the right hand side is called co-efficient of friction f. Thus $f = \phi \left[R_e, \frac{k}{D} \right]$

This equation shows that friction co-efficient is a function of Reynolds number and k/D ratio, where k is the average height of pipe wall roughness protrusions.

(*a*) **Variation of '*f* ' for Laminar Flow.** In viscous flow chapter, it is shown that co-efficient of friction '*f*' for laminar flow in pipes is given by

$$f = \frac{16}{R_e} \qquad ...(10.29)$$

Thus friction co-efficient is only a function of Reynolds number in case of laminar flow. It is independent of (k/D) ratio.

(*b*) **Variation of '*f*' for Turbulent Flow.** For turbulent flow, the co-efficient of friction is a function of R_e and k/D ratio. For relative roughness (k/D), in the turbulent flow the boundary may be smooth or rough and hence the value of '*f* 'will be different for these boundaries.

(*i*) **'*f*' for smooth pipes.** For turbulent flow in smooth pipes, co-efficient of friction is a function of Reynolds number only. The value of laminar sub-layer in case of smooth pipe is large as compared to the average height of surface roughness k. The value of '*f* ' for smooth pipe for Reynolds number varying from 4000 to 100000 is given by the relation

$$f = \frac{.0791}{(R_e)^{1/4}} \qquad ...(10.30)$$

The equation (10.30) is given by Blasius.

The value of '*f* ' for $R_e > 10^5$ is obtained from equation (10.23) which gives the velocity distribution for smooth pipe in terms of average velocity $(\overline{U})$ as

$$\frac{\overline{U}}{u_*} = 5.75 \log_{10} \frac{(u_* R)}{\nu} + 1.75 \qquad ...(10.31)$$

From equation (10.6), we have $f = \frac{2\tau_0}{\rho V^2}$, where V = average velocity

$$\therefore \quad f = \frac{2\tau_0}{\rho \overline{U}^2} = \frac{2}{\overline{U}^2} \left(\sqrt{\frac{\tau_0}{\rho}} \right)^2 = \frac{2}{\overline{U}^2} \times u_*^2 \qquad \left[\because \sqrt{\frac{\tau_0}{\rho}} = u_* \right]$$

$$\therefore \quad u_*^2 = \frac{f \overline{U}^2}{2}$$

or
$$u_* = \overline{U} \sqrt{\frac{f}{2}} \qquad ...(10.31A)$$

Substituting the value of u_* in equation (10.31), we get

$$\frac{\overline{U}}{\overline{U} \sqrt{f/2}} = 5.75 \log_{10} \left(\frac{\sqrt{f/2}}{\nu} \right) R + 1.75$$

or $$\frac{1}{\sqrt{f/2}} = 5.75 \log_{10}\left(\frac{\overline{U}R}{\nu}\sqrt{f/2}\right) + 1.75$$

Taking $R = D/2$ and simplifying, the above equation is written as

$$\frac{1}{\sqrt{4f}} = 2.03 \log_{10}\left(\frac{\overline{U}D}{\nu}\sqrt{4f}\right) - 0.91$$

But $\frac{\overline{U}D}{\nu} = R_e$ and hence above equation is written as

$$\frac{1}{\sqrt{4f}} = 2.03 \log_{10}(R_e\sqrt{4f}) - 0.91 \quad ...(10.32)$$

Equation (10.32) is valid upto $R_e = 4 \times 10^6$

Nikuradse's experimental result for turbulent flow in smooth pipe for 'f'is

$$\frac{1}{\sqrt{4f}} = 2.0 \log_{10}(R_e\sqrt{4f}) - 0.8 \quad ...(10.33)$$

This is applicable upto $R_e = 4 \times 10^7$. But the equation (10.33) is solved by hit and trial method. The value of 'f' (*i.e.*, co-efficient of friction) can alternately be obtained as

$$f = .0008 + \frac{.05525}{(R_e)^{0.237}} \quad ...(10.34)$$

The value of 'f'[*i.e.*, friction factor which is used in equation (10.2*A*)] is given by

$$f = 0.0032 + \frac{0.221}{(R_e)^{0.237}} \quad ...(10.34A)$$

(*ii*) **Value of 'f' for rough pipes.** For turbulent flow in rough pipes, the co-efficient of friction is a function of relative roughness (k/D) and it is independent of Reynolds number. This is because the value of laminar sub-layer for rough pipes is very small as compared to the height of surface roughness. The average velocity for rough pipes is given by (10.24) as

$$\frac{\overline{U}}{u_*} = 5.75 \log_{10}(R/k) + 4.75$$

But $$u_* = \overline{U}\sqrt{f/2}$$

Substituting the value of u_* in the above equation, we get

$$\frac{\overline{U}}{\overline{U}\sqrt{f/2}} = 5.75 \log_{10}(R/k) + 4.75$$

which is simplified to the form as $$\frac{1}{\sqrt{4f}} = 2.03 \log_{10}(R/k) + 1.68 \quad ...(10.35)$$

But Nikuradse's experimental result gave for rough pipe the following relation for 'f' as

$$\frac{1}{\sqrt{4f}} = 2 \log_{10}(R/k) + 1.74 \quad ...(10.36)$$

(*c*) **Value of 'f' for commercial pipes.** The value of 'f' for commercial pipes such as pipes made of metal, concrete and wood is obtained from Nikuradse's experimental data for smooth and rough

pipes. According to Colebrook, by subtracting $2 \log_{10} (R/k)$ from both sides of equations (10.33) and (10.36), the value of 'f' is obtained for commercial smooth and rough pipes as :

1. Smooth pipes

$$\frac{1}{\sqrt{4f}} - 2 \log_{10} (R/k) = 2 \log_{10} (R_e \sqrt{4f}) - 0.8 - 2 \log_{10} (R/k)$$

$$= 2 \log_{10} \left(\frac{R_e \sqrt{4f}}{R / k} \right) - 0.8 \qquad ...(10.37)$$

2. Rough pipes

$$\frac{1}{\sqrt{4f}} - 2 \log_{10} (R/k) = 2 \log_{10} (R/k) + 1.74 - 2 \log_{10} (R/k)$$

$$= 1.74. \qquad ...(10.38)$$

Problem 10.7 *For the problem 10.6, find the co-efficient of friction and the average height of roughness projections.*

Solution. From the solution of problem 10.6, we have

$$R = 0.15 \text{ m}$$
$$u_* = 0.1458 \text{ m/s}$$
$$\overline{U} = 1.4533 \text{ m/s}$$

For co-efficient of friction, we know that

$$u_* = \overline{U} \sqrt{f/2}$$

or $$0.1458 = 1.4533 \sqrt{f/2}$$

or $$\sqrt{f/2} = \frac{0.1458}{1.4533} = 0.1$$

$\therefore$ $$f = 2.0 \times (.1)^2 = \textbf{.02. Ans.}$$

Height of roughness projection is obtained from equation (10.36) as

$$\frac{1}{\sqrt{4f}} = 2 \log_{10} (R/k) + 1.74$$

Substituting the values of R and f, we get

$$\frac{1}{\sqrt{4 \times 0.2}} = 2 \log_{10} \left(\frac{0.15}{k} \right) + 1.74 \text{ or } 3.5355 = 2 \log_{10} \left(\frac{.15}{k} \right) + 1.74$$

or $$\log_{10} \left(\frac{.15}{k} \right) = \frac{3.5355 - 1.74}{2} = 0.8977 = \log_{10} 7.90$$

$\therefore$ $$\frac{0.15}{k} = 7.90$$

$\therefore$ $$k = \frac{0.15}{7.90} = 0.01898 \text{ m} = \textbf{18.98 mm. Ans.}$$

Problem 10.8 *Water is flowing through a rough pipe of diameter 500 mm and length 4000 m at the rate of 0.5 m^3/s. Find the power required to maintain this flow. Take the average height of roughness as k = 0.40 mm.*

Solution. Given :

Dia. of rough pipe, $D = 500 \text{ mm} = 0.50 \text{ m}$

$\therefore$ Radius, $R = \dfrac{D}{2} = 0.25 \text{ m}$

Length of pipe, $L = 4000 \text{ m}$

Discharge, $Q = 0.5 \text{ m}^3/\text{s}$

Average height of roughness, $k = 0.40 \text{ mm} = 0.4 \times 10^{-3} \text{ m}$

First find the value of co-efficient of friction. Then calculate the head lost due to friction and then power required.

For a rough pipe, the value of 'f' is given by the equation (10.36) as

$$\frac{1}{\sqrt{4f}} = 2 \log_{10} (R/k) + 1.74 = 2 \log_{10} \left(\frac{.25}{.4 \times 10^{-3}} \right) + 1.74$$

$$= 2 \log_{10} (625.0) + 1.74 = 5.591 + 1.74 = 7.331$$

or $$\sqrt{4f} = \frac{1}{7.331} = 0.1364 \text{ or } f = (0.1364)^2/4 = .00465$$

Also the average velocity, $\overline{U} = \dfrac{\text{Discharge}}{\text{Area}} = \dfrac{0.5}{\frac{\pi}{4} D^2} = \dfrac{0.5}{\frac{\pi}{4}(.5)^2} = 2.546$

$\therefore$ Head lost due to friction, $h_f = \dfrac{4 \,.\, f \,.\, L \,.\, V^2}{d \times 2g} = \dfrac{4 \times .00465 \times 4000 \times 2.546^2}{0.5 \times 2 \times 9.81}$

$= 49.16 \text{ m}$ [$\because V = \overline{U} = 2.546$, $d = D = 0.5$]

$\therefore$ Power required, $P = \dfrac{W \times h_f}{1000} = \dfrac{w.Q.h_f}{1000} = \dfrac{\rho \times g \times Q \times h_f}{1000} \text{ kW}$

$$= \frac{1000 \times 9.81 \times 0.5 \times 49.16}{1000} = \mathbf{241.13 \text{ kW. Ans.}}$$

Problem 10.9 *A smooth pipe of diameter 400 mm and length 800 m carries water at the rate of 0.04 m^3/s. Determine the head lost due to friction, wall shear stress, centre-line velocity and thickness of laminar sub-layer. Take the kinematic viscosity of water as 0.018 stokes.*

Solution. Given :

Dia. of pipe, $D = 400 \text{ mm} = 0.40 \text{ m}$

$\therefore$ Radius, $R = \dfrac{D}{2} = 0.20 \text{ m}$

Length of pipe, $L = 800 \text{ m}$

Discharge, $Q = 0.04 \text{ m}^3/\text{s}$

Kinematic viscosity, $\nu = 0.018 \text{ stokes} = 0.018 \text{ cm}^2/\text{s} = 0.018 \times 10^{-4} \text{ m}^2/\text{s}$

Average velocity, $\overline{U} = \dfrac{Q}{\text{Area}} = \dfrac{0.04}{\frac{\pi}{4}(0.4)^2} = 0.3183 \text{ m/s}$

$\therefore$ Reynolds number, $R_e = \dfrac{V \times D}{\nu} = \dfrac{\overline{U} \times D}{\nu} = \dfrac{0.3183 \times 0.4}{.018 \times 10^{-4}} = 7.073 \times 10^4$

The flow is turbulent.

The co-efficient of friction 'f' is obtained from equation (10.30) as

$$f = \frac{.0791}{(R_e)^{1/4}} = \frac{0.0791}{\left(7.073 \times 10^4\right)^{1/4}} = \frac{.0791}{16.30} = .00485$$

(*i*) Head lost due to friction, $h_f = \dfrac{4 . f . L . V^2}{D \times 2g} = \dfrac{4 . f . L . \overline{U}^2}{D \times 2g}$

$$= \frac{4 \times .00485 \times 800 \times (.3183)^2}{0.40 \times 2 \times 9.81} = \mathbf{0.20\ m.\ Ans.}$$

(*ii*) Wall shear stress (τ_0) is given by equation (10.5) as

$$\tau_0 = \frac{f . \rho . V^2}{2} = \frac{f . \rho . \overline{U}^2}{2} \qquad [\because V = \overline{U}\ ]$$

$$= 0.00485 \times 1000 \times \frac{(.3184)^2}{2.0}\ \text{N/m}^2 = \mathbf{0.245\ N/m^2.\ Ans.}$$

(*iii*) The centre-line velocity (u_{max}) for smooth pipe is given by equation (10.20) as in which $u = u_{max}$ at $y = R$

$$\therefore \qquad \frac{u_{max}}{u_*} = 5.75 \log_{10} \frac{u_* R}{\nu} + 5.55 \text{ [Put in equation (10.20), } u = u_{max} \text{ at } y = R]$$

where the shear velocity $u_* = \sqrt{\dfrac{\tau_0}{\rho}} = \sqrt{\dfrac{0.245}{1000}} = \sqrt{0.000245} = 0.0156$ m/s

Substituting the values of u_*, R and ν in the above equation, we get

$$\frac{u_{max}}{0.0156} = 5.75 \log_{10} \frac{0.0156 \times 0.20}{.018 \times 10^{-4}} + 5.55 = 24.173$$

or $$u_{max} = 24.173 \times .0156 = \mathbf{0.377\ m/s.\ Ans.}$$

(*iv*) The thickness of laminar sub-layer (δ') is given by

$$\delta' = \frac{11.6 \times \nu}{u_*} = \frac{11.6 \times .018 \times 10^{-4}}{.0156} = .001338\ \text{m} = \mathbf{1.338\ mm.\ Ans.}$$

Problem 10.10 *A rough pipe of diameter 400 mm and length 1000 m carries water at the rate of 0.4 m^3/s. The wall roughness is 0.012 mm. Determine the co-efficient of friction, wall shear stress, centre-line velocity and velocity at a distance of 150 mm from the pipe wall.*

Solution. Given :

Dia. of rough pipe, $D = 400$ mm $= 0.4$ m

$\therefore$ Radius, $R = \dfrac{D}{2} = \dfrac{0.4}{2} = 0.20$ m

Length of pipe, $L = 1000$ m

Discharge, $Q = 0.4\ \text{m}^3/\text{s}$

Wall roughness, $k = 0.012\ \text{mm} = 0.012 \times 10^{-3}\ \text{m}$

(*i*) The value of co-efficient of friction '*f*' for rough pipe is given by the equation (10.36) as

$$\frac{1.0}{\sqrt{4f}} = 2 \log_{10} (R/k) + 1.74$$

or
$$\frac{1.0}{\sqrt{4f}} = 2 \log_{10} \left(\frac{0.20}{.012 \times 10^{-3}} \right) + 1.74$$

$$= 2 \log_{10} (16666.67) + 1.74 = 10.183$$

$\therefore$
$$4f = \left(\frac{1}{10.183} \right)^2 = .00964$$

$\therefore$
$$f = \frac{.00964}{4.0} = \textbf{.00241. Ans.}$$

(*ii*) Centre-line velocity (u_{max}) for rough pipe is given by equation (10.21) in which u is made $= u_{max}$ at $y = R$ and hence

$$\frac{u_{max}}{u_*} = 5.75 \log_{10} (R/k) + 8.5 \qquad ...(i)$$

where shear velocity, $u_* = \sqrt{\frac{\tau_0}{\rho}}$

and τ_0 = wall shear stress $= \frac{f \cdot \rho \cdot V^2}{2}$

where $V = \frac{\text{Discharge}}{\text{Area}} = \frac{Q}{\frac{\pi}{4} D^2} = \frac{Q}{\frac{\pi}{4}(.4)^2} = \textbf{3.183 m/s. Ans.}$

(*iii*) $\therefore$
$$\tau_0 = \frac{f \cdot \rho \cdot V^2}{2} = .00241 \times 1000 \times \frac{3.183^2}{2.0} = \textbf{12.2 N/m}^2\textbf{. Ans.}$$

$\therefore$
$$u_* = \sqrt{\frac{\tau_0}{\rho}} = \sqrt{\frac{12.2}{1000}} = 0.11 \text{ m/s}$$

Substituting the value of u_*, R, k in equation (*i*), we get

$$\frac{u_{max}}{0.11} = 5.75 \log_{10} \left(\frac{0.2}{.012 \times 10^{-3}} \right) + 8.5 = 32.77$$

$\therefore$
$$u_{max} = 32.77 \times 0.11 = \textbf{3.60 m/s. Ans.}$$

(*iv*) Velocity (u) at a distance y = 150 mm = 0.15 m

The velocity (u) at any point for rough pipe is given by equation (10.21) as

$$\frac{u}{u*} = 5.75 \log_{10} (y/k) + 8.5$$

where u_* = 0.11 m/s and y = 0.15 m, $k = 0.012 \times 10^{-3}$ m

$$\because \quad \frac{u}{0.11} = 5.75 \log_{10}\left(\frac{0.15}{.012 \times 10^{-3}}\right) + 8.5 = 32.05$$

$$\therefore \quad u = 32.05 \times 0.11 = \mathbf{3.52\ m/s.\ Ans.}$$

Problem 10.11 *A smooth pipe line of 100 mm diameter carries 2.27 m^3 per minute of water at 20°C with kinematic viscosity of 0.0098 stokes. Calculate the friction factor, maximum velocity as well as shear stress at the boundary.*

Solution. Given :

Dia. of pipe, $D = 100$ mm $= 0.1$ m

$\therefore$ Radius of pipe, $R = 0.05$ m

Discharge, $Q = 2.27\ m^3/min = \frac{2.27}{60}\ m^3/s = 0.0378\ m^3/s$

Kinematic viscosity, $\nu = 0.0098$ stokes $= 0.0098\ cm^2/s = 0.0098 \times 10^{-4}\ m^2/s$

Now average velocity is given by $\overline{U} = \frac{Q}{\text{Area}} = \frac{0.0378}{\frac{\pi}{4}(0.1)^2} = \frac{0.0378 \times 4}{\pi \times 0.01} = 4.817$ m/s

$\therefore$ Reynolds number is given by, $R_e = \frac{\overline{U} \times D}{\nu} = \frac{4.817 \times 0.1}{0.0098 \times 10^{-4}} = 4.9154 \times 10^5$.

The flow is turbulent and R_e is more than 10^5. Hence for smooth pipe, the co-efficient of friction 'f' is obtained from equation (10.33) as

$$\frac{1}{\sqrt{4f}} = 2.0 \log_{10}(R_e\sqrt{4f}) - 0.8$$

or
$$\frac{1}{\sqrt{4f}} = 2.0 \log_{10}(4.9154 \times 10^5 \times \sqrt{4f}) - 0.8$$

$$= 2.0\ [\log_{10} 4.9154 \times 10^5 + \log_{10}\sqrt{4f}\ ] - 0.8$$

$$= 2.0\ [5.6915 + \log_{10}\sqrt{4f}\ ] - 0.8 = 2 \times 5.6915 + 2\log_{10}\sqrt{4f} - 0.8$$

$$= 11.3830 + \log_{10}\left(\sqrt{4f}\right)^2 - 0.8 = 11.383 + \log_{10}(4f) - 0.8$$

or
$$\frac{1}{\sqrt{4f}} - \log_{10}(4f) = 11.383 - 0.8 = 10.583 \qquad ...(i)$$

(*i*) *Friction factor*

Now, friction factor (f^*) = 4 × co-efficient of friction = $4f$

Substituting the value of '$4f$' in equation (*i*), we get

$$\frac{1}{\sqrt{f^*}} - \log_{10} f^* = 10.583 \qquad ...(ii)$$

The above equation is solved by hit and trial method.

Let $f^* = 0.1$, then L.H.S. of equation (*ii*), becomes as

$$\text{L.H.S.} = \frac{1}{\sqrt{0.1}} - \log_{10} 0.1 = 3.16 - (-1.0) = 4.16$$

Let $f^* = 0.01$, then L.H.S. of equation (*ii*), becomes as

$$\text{L.H.S.} = \frac{1}{\sqrt{0.01}} - \log_{10} 0.01 = 10 - (-2) = 12$$

But for exact solution, L.H.S. should be 10.583. Hence value of f^* lies between 0.1 and 0.01.

Let $f^* = 0.013$ then L.H.S. of equation (*ii*), becomes as

$$\text{L.H.S.} = \frac{1}{\sqrt{0.013}} - \log_{10} 0.013 = 8.77 - (-1.886) = 8.77 + 1.886 = 10.656$$

which is approximately equal to 10.583.

Hence the value of f^* is equal to 0.013.

$\therefore$ Friction factor, $f^* = \mathbf{0.013.\ Ans.}$

(*ii*) *Maximum velocity* (u_{max})

Now we know that $f^* = 4f$

$\therefore$ Co-efficient of friction, $f = \dfrac{f^*}{4} = \dfrac{0.013}{4} = 0.00325$

Now the shear velocity (u_*) in terms of co-efficient of friction and average velocity is given by equation (10.31*A*) as

$$u_* = \overline{U}\sqrt{\frac{f}{2}} = 4.817 \times \sqrt{\frac{0.00325}{2}} = 4.817 \times 0.0403 = 0.194$$

For smooth pipe, the velocity at any point is given by equation (10.20)

$$u = u_* \left[5.75 \log_{10} \frac{u_* \times y}{\nu} + 5.55\right]$$

The velocity will be maximum at the centre of the pipe,
where $y = R = 0.05$
i.e., radius of pipe. Hence the above equation becomes as

$$U_{max} = u_* \left[5.75 \log_{10} \frac{u_* \times R}{\nu} + 5.55\right]$$

$$= 0.194 \left[5.75 \log_{10} \frac{0.194 \times 0.05}{0.0098 \times 10^{-4}} + 5.55\right]$$

$$= 0.194\ [22.974 + 5.55] = \mathbf{5.528\ m/s.\ Ans.}$$

(*iii*) *Shear stress at the boundary* (τ_0)

We know that $u_* = \sqrt{\dfrac{\tau_0}{\rho}}$ or $u_*^2 = \dfrac{\tau_0}{\rho}$

$\therefore$ $\tau_0 = \rho\, u^2{}_* = 1000 \times 0.194^2 = \mathbf{37.63\ N/m^2.\ Ans.}$

Problem 10.12 *Hydrodynamically smooth pipe carries water at the rate of 300 l/s at 20°C (ρ = 1000 kg/m^3, $\nu = 10^{-6}$ m^2/s) with a head loss of 3 m in 100 m length of pipe. Determine the pipe diameter. Use* $f = 0.0032 + \dfrac{0.221}{(R_e)^{0.237}}$ *equation for f, where* $h_f = \dfrac{f \times L \times V^2}{D \times 2g}$ *and* $R_e = \dfrac{\rho VD}{\mu}$.

Solution. Given :

Discharge, $Q = 300$ l/s $= 0.3$ m^3/s

Density, $\rho = 1000$ kg/m^3

Kinematic viscosity, $\nu = 10^{-6}$ m^2/s

Head loss, $h_f = 3$ m

Length of pipe, $L = 100$ m

Value of friction factor, $$f = 0.0032 + \frac{0.221}{(R_e)^{0.237}}$$

Reynolds number, $$R_e = \frac{\rho VD}{\mu} = \frac{V \times D}{\nu} \qquad \left(\because \frac{\mu}{\rho} = \nu\right)$$

$$= \frac{V \times D}{10^{-6}} = V \times D \times 10^6$$

Find : Diameter of pipe.

Let D = Diameter of pipe

Head loss in terms of friction factor is given as

$$h_f = \frac{f \times L \times V^2}{D \times 2g}$$

or $$3 = \frac{f \times 100 \times V^2}{D \times 2 \times 9.81} \qquad (\because h_f = 3, L = 100 \text{ m})$$

or $$f = \frac{3 \times D \times 2 \times 9.81}{100\, V^2} \text{ or } f = \frac{0.5886\, D}{V^2} \qquad ...(i)$$

Now $$Q = A \times V$$

or $$0.3 = \frac{\pi}{4} D^2 \times V \text{ or } D^2 \times V = \frac{4 \times 0.3}{\pi} = 0.382$$

$\therefore$ $$V = \frac{0.382}{D^2} \qquad ...(ii)$$

Also $$f = 0.0032 + \frac{0.221}{(R_e)^{0.237}}$$

or $$\frac{0.5886\, D}{V^2} = 0.0032 + \frac{0.221}{\left(V \times D \times 10^6\right)^{0.237}}$$

$$\left(\because \text{From equation } (i), f = \frac{0.5886\, D}{V^2} \text{ and } R_e = V \times D \times 10^6\right)$$

or $$\frac{0.5886\, D}{\left(\frac{0.382}{D^2}\right)^2} = 0.0032 + \frac{0.221}{\left(\frac{0.382}{D^2} \times D \times 10^6\right)^{0.237}}$$

$$\left(\because \text{From equation}(ii), V = \frac{0.382}{D^2}\right)$$

or $$\frac{0.5886 \times D^5}{0.382^2} = 0.0032 + \frac{0.221}{\frac{\left(0.382 \times 10^6\right)^{0.237}}{D^{0.237}}}$$

or $$4.033\ D^5 = 0.0032 + 0.0105 \times D^{0.237}$$

or $$4.033\ D^5 - 0.0105\ D^{0.237} - 0.0032 = 0 \qquad ...(iii)$$

The above equation (*iii*) will be solved by hit and trial method.

(*i*) Assume $D = 1$ m, then L.H.S. of equation (*iii*), becomes as

$$\text{L.H.S.} = 4.033 \times 1^5 - 0.0105 \times 1^{0.237} - 0.0032$$
$$= 4.033 - 0.0105 - 0.0032 = 4.0193$$

By increasing the value of D more than 1 m, the L.H.S. will go on increasing. Hence decrease the value of D.

(*ii*) Assume $D = 0.3$ m, then L.H.S. of equation (*iii*),

becomes as $\text{L.H.S.} = 4.033 \times 0.3^5 - 0.0105 \times 0.3^{0.237} - 0.0032$

$$= 0.0098 - 0.00789 - 0.0032 = -0.00129$$

As this value is negative, the value of D will be slightly more than 0.3.

(*iii*) Assume $D = 0.306$ m, then L.H.S. of equation (*iii*), becomes as

$$\text{L.H.S.} = 4.033 \times 0.306^5 - 0.0105 \times 0.306^{0.237} - 0.0032$$
$$= 0.0108 - 0.00793 - 0.0032 = -0.00033$$

This value of L.H.S. is approximately equal to zero. Actually the value of D will be slightly more than 0.306 m say **0.308 m. Ans.**

Problem 10.13 *Water is flowing through a rough pipe of diameter 600 mm at the rate of 600 litres/second. The wall roughness is 3 mm. Find the power lost for 1 km length of pipe.*

Solution. Given :

Dia. of pipe, $D = 600$ mm $= 0.6$ m

$\therefore$ Radius of pipe, $R = \frac{0.6}{2} = 0.3$ m

Discharge, $Q = 600$ litre/s $= 0.6$ m^3/s

Wall roughness, $k = 3$ mm $= 3 \times 10^{-3}$ m $= 0.003$ m

Length of pipe, $L = 1$ km $= 1000$ m

For rough pipes, the co-efficient of friction in terms of wall roughness, k is given by equation (10.36) as

$$\frac{1}{\sqrt{4f}} = 2 \log_{10} (R/k) + 1.74 = 2 \log_{10} \left(\frac{0.3}{0.003}\right) + 1.74 = 5.74$$

or $$\sqrt{4f} = \frac{1}{5.76} = 0.1742 \text{ or } 4f = (0.1742)^2 = 0.03035$$

The head loss due to friction is given by, $h_f = \frac{4f \times L \times V^2}{D \times 2g}$

where $V = \frac{Q}{A} = \frac{0.6}{\frac{\pi}{4}(0.6^2)} = 2.122$ m/s

$$h_f = \frac{0.03035 \times 1000 \times 2.122^2}{0.6 \times 2 \times 9.81} = 11.6 \text{ m}$$

The power* lost is given by, $P = \frac{\rho g \times Q \times h_f}{1000} = \frac{1000 \times 9.81 \times 0.6 \times 11.6}{1000}$ kW = **68.27 kW. Ans.**

HIGHLIGHTS

1. If the Reynold number is less than 2000 in a pipe, the flow is laminar while if the Reynold number is more than 4000, the flow is turbulent in pipes.
2. Loss of pressure head in a laminar flow is proportional to the mean velocity of flow, while in case of turbulent flow it is approximately proportional to the square of velocity.
3. Expression for head loss due to friction in pipes is given by Darcy-Weisbach equation,

$$h_f = \frac{4 \times f \times L \times V^2}{d \times 2g}, \text{ where } f = \text{co-efficient of friction}$$

$$= \frac{f \times L \times V^2}{d \times 2g}, \text{ where } f = \text{friction factor}$$

4. Co-efficient of friction is expressed in terms of shear stress as $= \frac{2\tau_0}{\rho V^2}$

   where V = mean velocity of flow, ρ = mass density of fluid.
5. Shear stress in turbulent flow is sum of shear stress due to viscosity and shear stress due to turbulence, *i.e.*,

$$\tau = \tau_v + \tau_t, \quad \text{where} \quad \tau_v = \text{shear stress due to viscosity}, \ \tau_t = \text{shear stress due to turbulence}$$

$$= \mu \frac{d\bar{u}}{dy} + \eta \frac{d\bar{u}}{dy}$$

6. Turbulent shear stress by Reynolds is given as $\tau = \rho\, u'v'$
   where u' and v' = fluctuating component of velocity.
7. The expression for shear stress in turbulent flow due to Prandtl is $\bar{\tau} = \rho l^2 \left(\frac{du}{dy}\right)^2$, where l = mixing length.
8. The velocity distribution in the turbulent flow for pipes is given by the expression

$$u = u_{max} + 2.5\, u_* \log_e (y/R)$$

   where u_{max} = is the centre-line velocity,
   y = distance from the pipe wall,
   R = radius of the pipe,

   and u_* = shear velocity which is equal to $\sqrt{\frac{\tau_0}{\rho}}$.

* Power = $\rho g \times Q \times h_f$ watt = $\frac{\rho g \times Q \times h_f}{1000}$ kW.

9. Velocity defect is the difference between the maximum velocity (u_{max}) and local velocity (u) at any point and is given by $(u_{max} - u) = 5.75 \times u_* \log_{10} (R/y)$.

10. The boundary is known as hydrodynamically smooth if k, the average height of the irregularities projecting from the surface of the boundary is small compared to the thickness of the laminar sub-layer (δ') and boundary is rough if k is large in comparison with the thickness of the sub-layer.

or if $\frac{k}{\delta'} < 0.25$, the boundary is smooth ; if $\frac{k}{\delta'} > 6.0$, the boundary is rough

and if $\frac{k}{\delta'}$ lie between 0.25 to 6.0, the boundary is in transition.

11. Velocity distribution for turbulent flow is

$$\frac{u}{u_*} = 5.75 \log_{10} \frac{u_* y}{\nu} + 5.55 \text{ for smooth pipes}$$

$$= 5.75 \log_{10} (y/k) + 8.5 \text{ for rough pipes}$$

where u = velocity at any point in the turbulent flow,

u_* = shear velocity and = $\sqrt{\frac{\tau_0}{\rho}}$, ν = kinematic viscosity of fluid,

y = distance from pipe wall, and k = roughness factor.

12. Velocity distribution in terms of average velocity is

$$\frac{\overline{U}}{u_*} = 5.75 \log_{10} \frac{u_* R}{\nu} + 1.75 \text{ for smooth pipes,}$$

$$= 5.75 \log_{10} R/k + 4.75 \text{ for rough pipes.}$$

13. Difference of local velocity and average velocity for smooth and rough pipes is

$$\frac{u - \overline{U}}{u_*} = 5.75 \log_{10} (y/R) + 3.75.$$

14. The co-efficient of friction is given by

$$f = \frac{16}{R_e} \text{ ..... for laminar flow,}$$

$$= \frac{0.0791}{(R_e)^{1/4}} \text{ for turbulent flow in smooth pipes for } R_e \geq 4000 \text{ by} \leq 10^5$$

$$= .0008 + \frac{.05525}{(R_e)^{.257}} \text{ for } R_e \leq 10^5 \text{ but} \geq 4 \times 10^7$$

$$\frac{1}{\sqrt{4f}} = 2 \log_{10} (R/k) + 1.74 \text{ for rough pipes where } R_e = \text{Reynolds number.}$$

EXERCISE

(A) THEORETICAL PROBLEMS

1. What do you understand by turbulent flow ? What factor decides the type of flow in pipes ?

2. (*a*) Derive an expression for the loss of head due to friction in pipes.
(*b*) Derive Darcy-Weisbach equation. (*J.N.T.U., Hyderabad, S 2002*)

3. Explain the term co-efficient of friction. On what factors does this co-efficient depend ?

4. Obtain an expression for the co-efficient of friction in the terms of shear stress.
5. What do you mean by Prandtl mixing Length Theory ? Find an expression for shear stress due to Prandtl.
6. Derive an expression for Prandtl's universal velocity distribution for turbulent flow in pipes. Why this velocity distribution is called universal ?
7. What is a velocity defect ? Derive an expression for velocity defect in pipes.
8. How would you distinguish between hydrodynamically smooth and rough boundaries ?
9. Obtain an expression for the velocity distribution for turbulent flow in smooth pipes.
10. Show that velocity distribution for turbulent flow through rough pipe is given by

$$\frac{u}{u_*} = 5.75 \log_{10} (y/k) + 8.5$$

where u_* = shear velocity, y = distance from pipe wall, k = roughness factor.

11. Obtain an expression for velocity distribution in terms of average velocity for (*a*) smooth pipes and (*b*) rough pipes.
12. Prove that the difference of local velocity and average velocity for turbulent flow through rough or smooth pipes is given by

$$\frac{u - \overline{U}}{u_*} = 5.75 \log_{10} (y/R) + 3.75.$$

13. Obtain an expression for velocity distribution in turbulent flow for (*i*) smooth pipes and (*ii*) rough pipes. (*Delhi University, December, 2002*)

(B) NUMERICAL PROBLEMS

1. A pipe-line carrying water has average height of irregularities projecting from the surface of the boundary of the pipe as 0.20 mm. What type of the boundary is it ? The shear stress development is 7.848 N/m^2. Take value of kinematic viscosity for water as 0.01 stokes. [**Ans.** Boundary is in transition]
2. Determine the average height of the roughness for a rough pipe of diameter 10.0 cm when the velocity at a point 4 cm from wall is 40% more than the velocity at a point 1 cm from pipe wall. [**Ans.** 0.94 cm]
3. A smooth pipe of diameter 10 cm and 1000 m long carries water at the rate of 0.70 m^2/minute. Calculate the loss of head, wall shearing stress, centre line velocity, velocity and shear stress at 3 cm from pipe wall. Also calculate the thickness of the laminar sub-layer. Take kinematic viscosity of water as 0.015 stokes and value of co-efficient of friction 'f' as

$$f = \frac{.0791}{(R_e)^{1/4}}, \text{ where } R_e = \text{Reynolds number.}$$

[**Ans.** 20.05 m, 4.9 N/m^2 ; 1.774 m/s ; 1.65 m/s ; 19.62 N/m^2 ; 0.248 mm]

4. The velocities of water through a pipe of diameter 10 cm, are 4 m/s and 3.5 m/s at the centre of the pipe and 2 cm from the pipe centre respectively. Determine the wall shearing stress in the pipe for turbulent flow. [**Ans.** 15.66 kgf/m^2]
5. For turbulent flow in a pipe of diameter 200 mm, find the discharge when the centre-line velocity is 30 m/s and velocity at a point 80 mm from the centre as measured by pitot-tube is 2.0 m/s. [**Ans.** 64.9 litres/s]
6. For problem 5, find the co-efficient of friction and the average height of roughness projections. [**Ans.** 0.029, 25.2 mm]
7. Water is flowing through a rough pipe of diameter 40 cm and length 3000 m at the rate of 0.4 m^3/s. Find the power required to maintain this flow. Take the average height of roughness as K = 0.3 mm. [**Ans.** 278.5 kN]

8. A smooth pipe of diameter 300 mm and length 600 m carries water at rate of 0.04 m^2/s. Determine the head lost due to friction, wall shear stress, centre-line velocity and thickness of laminar sub-layer. Take the kinematic viscosity of water as 0.018 stokes. [**Ans.** 0.588 m, 0.72 N/cm^2, 0.665 m/s, 0.779 mm]

9. A rough pipe of diameter 300 mm and length 800 m carries water at the rate of 0.4 m^2/s. The wall roughness is 0.015 mm. Determine the co-efficient of friction, wall shear stress, centre line velocity and velocity at a distance of 100 mm from the pipe wall.

[**Ans.** $f = .00263$, $\tau_0 = 42.08$ N/cm^2, $u_{max} = 6.457$ m/s, $u = 6.249$ m/s]

10. Determine the distance from the centre of the pipe, at which the local velocity is equal to the average velocity for turbulent flow in pipes. [**Ans.** 0.7772 R]

11 CHAPTER

FLOW THROUGH PIPES

▶ 11.1 INTRODUCTION

In chapters 9 and 10, laminar flow and turbulent flow have been discussed. We have seen that when the Reynolds number is less than 2000 for pipe flow, the flow is known as laminar flow whereas when the Reynolds number is more than 4000, the flow is known as turbulent flow. In this chapter, the turbulent flow of fluids through pipes running full will be considered. If the pipes are partially full as in the case of sewer lines, the pressure inside the pipe is same and equal to atmospheric pressure. Then the flow of fluid in the pipe is not under pressure. This case will be taken in the chapter of flow of water through open channels. Here we will consider flow of fluids through pipes under pressure only.

▶ 11.2 LOSS OF ENERGY IN PIPES

When a fluid is flowing through a pipe, the fluid experiences some resistance due to which some of the energy of fluid is lost. This loss of energy is classified as :

Energy Losses

1. Major Energy Losses	2. Minor Energy Losses
This is due to friction and it is calculated by the following formulae :	This is due to
(*a*) Darcy-Weisbach Formula	(*a*) Sudden expansion of pipe
(*b*) Chezy's Formula	(*b*) Sudden contraction of pipe
	(*c*) Bend in pipe
	(*d*) Pipe fittings etc.
	(*e*) An obstruction in pipe.

▶ 11.3 LOSS OF ENERGY (OR HEAD) DUE TO FRICTION

(*a*) **Darcy-Weisbach Formula.** The loss of head (or energy) in pipes due to friction is calculated from Darcy-Weisbach equation which has been derived in chapter 10 and is given by

$$h_f = \frac{4 . f . L . V^2}{d \times 2g} \qquad ...(11.1)$$

where h_f = loss of head due to friction

f = co-efficient of friction which is a function of Reynolds number

$= \dfrac{16}{R_e}$ for $R_e < 2000$ (viscous flow)

$= \dfrac{0.079}{R_e^{1/4}}$ for R_e varying from 4000 to 10^6

L = length of pipe,

V = mean velocity of flow,

d = diameter of pipe.

(*b*) **Chezy's Formula for loss of head due to friction in pipes.** Refer to chapter 10 article 10.3.1 in which expression for loss of head due to friction in pipes is derived. Equation (*iii*) of article 10.3.1, is

$$h_f = \frac{f'}{\rho g} \times \frac{P}{A} \times L \times V^2 \qquad ...(11.2)$$

where h_f = loss of head due to friction, P = wetted perimeter of pipe,
A = area of cross-section of pipe, L = length of pipe,
and V = mean velocity of flow.

Now the ratio of $\dfrac{A}{P}\left(= \dfrac{\text{Area of flow}}{\text{Perimeter (wetted)}}\right)$ is called hydraulic mean depth or hydraulic radius and is denoted by m.

$\therefore$ Hydraulic mean depth, $m = \dfrac{A}{P} = \dfrac{\frac{\pi}{4}d^2}{\pi d} = \dfrac{d}{4}$

Substituting $\dfrac{A}{P} = m$ or $\dfrac{P}{A} = \dfrac{1}{m}$ in equation (11.2), we get

$$h_f = \frac{f'}{\rho g} \times L \times V^2 \times \frac{1}{m} \text{ or } V^2 = h_f \times \frac{\rho g}{f'} \times m \times \frac{1}{L} = \frac{\rho g}{f'} \times m \times \frac{h_f}{L}$$

$\therefore$
$$V = \sqrt{\frac{\rho g}{f'} \times m \times \frac{h_f}{L}} = \sqrt{\frac{\rho g}{f'}} \sqrt{m \frac{h_f}{L}} \qquad ...(11.3)$$

Let $\sqrt{\dfrac{\rho g}{f'}} = C$, where C is a constant known as Chezy's constant and $\dfrac{h_f}{L} = i$, where i is loss of head per unit length of pipe.

Substituting the values of $\sqrt{\dfrac{\rho g}{f'}}$ and $\sqrt{\dfrac{h_f}{L}}$ in equation (11.3), we get

$$V = C\sqrt{mi} \qquad ...(11.4)$$

Equation (11.4) is known as Chezy's formula. Thus the loss of head due to friction in pipe from Chezy's formula can be obtained if the velocity of flow through pipe and also the value of C is known. The value of m for pipe is always equal to $d/4$.

Problem 11.1 *Find the head lost due to friction in a pipe of diameter 300 mm and length 50 m, through which water is flowing at a velocity of 3 m/s using (i) Darcy formula, (ii) Chezy's formula for which C = 60.*

Take ν for water = 0.01 stoke.

Solution. Given :

Dia. of pipe, $d = 300 \text{ mm} = 0.30 \text{ m}$

Length of pipe, $L = 50 \text{ m}$

Velocity of flow, $V = 3 \text{ m/s}$

Chezy's constant, $C = 60$

Kinematic viscosity, $\nu = 0.01 \text{ stoke} = 0.01 \text{ cm}^2/\text{s}$

$= 0.01 \times 10^{-4} \text{ m}^2/\text{s}.$

(*i*) **Darcy Formula** is given by equation (11.1) as

$$h_f = \frac{4 \,.\, f \,.\, L \,.\, V^2}{d \times 2g}$$

where 'f' = co-efficient of friction is a function of Reynolds number, R_e

But R_e is given by $R_e = \dfrac{V \times d}{\nu} = \dfrac{3.0 \times 0.30}{.01 \times 10^{-4}} = 9 \times 10^5$

$\therefore$ Value of $f = \dfrac{0.079}{R_e^{1/4}} = \dfrac{0.079}{\left(9 \times 10^5\right)^{1/4}} = .00256$

$\therefore$ Head lost, $h_f = \dfrac{4 \times .00256 \times 50 \times 3^2}{0.3 \times 2.0 \times 9.81} =$ **.7828 m. Ans.**

(*ii*) **Chezy's Formula.** Using equation (11.4)

$$V = C\sqrt{mi}$$

where $C = 60$, $m = \dfrac{d}{4} = \dfrac{0.30}{4} = 0.075$ m

$\therefore$ $3 = 60\sqrt{.075 \times i}$ or $i = \left(\dfrac{3}{60}\right)^2 \times \dfrac{1}{.075} = 0.0333$

But $i = \dfrac{h_f}{L} = \dfrac{h_f}{50}$

Equating the two values of i, we have $\dfrac{h_f}{50} = .0333$

$\therefore$ $h_f = 50 \times .0333 =$ **1.665 m. Ans.**

Problem 11.2 *Find the diameter of a pipe of length 2000 m when the rate of flow of water through the pipe is 200 litres/s and the head lost due to friction is 4 m. Take the value of C = 50 in Chezy's formulae.*

Solution. Given :

Length of pipe, $L = 2000$ m

Discharge, $Q = 200$ litre/s $= 0.2$ m^3/s

Head lost due to friction, $h_f = 4$ m

Value of Chezy's constant, $C = 50$

Let the diameter of pipe $= d$

Velocity of flow, $$V = \frac{\text{Discharge}}{\text{Area}} = \frac{Q}{\frac{\pi}{4}d^2} = \frac{0.2}{\frac{\pi}{4}d^2} = \frac{0.2 \times 4}{\pi d^2}$$

Hydraulic mean depth, $$m = \frac{d}{4}$$

Loss of head per unit length, $$i = \frac{h_f}{L} = \frac{4}{2000} = .002$$

Chezy's formula is given by equation (11.4) as $V = C\sqrt{mi}$

Substituting the values of V, m, i and C, we get

$$\frac{0.2 \times 4}{\pi d^2} = 50\sqrt{\frac{d}{4} \times .002} \text{ or } \sqrt{\frac{d}{4} \times .002} = \frac{0.2 \times 4}{\pi d^2 \times 50} = \frac{.00509}{d^2}$$

Squaring both sides, $$\frac{d}{4} \times .002 = \frac{.00509^2}{d^4} = \frac{.0000259}{d^4} \text{ or } d^5 = \frac{4 \times .0000259}{.002} = 0.0518$$

$\therefore$ $$d = \sqrt[5]{0.0518} = (.0518)^{1/5} = 0.553 \text{ m} = \mathbf{553 \text{ mm. Ans.}}$$

Problem 11.3 *A crude oil of kinematic viscosity 0.4 stoke is flowing through a pipe of diameter 300 mm at the rate of 300 litres per sec. Find the head lost due to friction for a length of 50 m of the pipe.*

Solution. Given :

Kinematic viscosity, $\nu = 0.4$ stoke $= 0.4$ cm^2/s $= .4 \times 10^{-4}$ m^2/s

Dia. of pipe, $d = 300$ mm $= 0.30$ m

Discharge, $Q = 300$ litres/s $= 0.3$ m^3/s

Length of pipe, $L = 50$ m

Velocity of flow, $$V = \frac{Q}{\text{Area}} = \frac{0.3}{\frac{\pi}{4}(0.3)^2} = 4.24 \text{ m/s}$$

$\therefore$ Reynolds number, $$R_e = \frac{V \times d}{\nu} = \frac{4.24 \times 0.30}{0.4 \times 10^{-4}} = 3.18 \times 10^4$$

As R_e lies between 4000 and 100000, the value of f is given by

$$f = \frac{.079}{(R_e)^{1/4}} = \frac{.079}{\left(3.18 \times 10^4\right)^{1/4}} = .00591$$

$\therefore$ Head lost due to friction, $h_f = \dfrac{4 \,.\, f \,.\, L \,.\, V^2}{d \times 2g} = \dfrac{4 \times .00591 \times 50 \times 4.24^2}{0.3 \times 2 \times 9.81}$ = **3.61 m. Ans.**

Problem 11.4 *An oil of sp. gr. 0.7 is flowing through a pipe of diameter 300 mm at the rate of 500 litres/s. Find the head lost due to friction and power required to maintain the flow for a length of 1000 m. Take* ν *= .29 stokes.*

Solution. Given :

Sp. gr. of oil, $S = 0.7$

Dia. of pipe, $d = 300$ mm $= 0.3$ m

Discharge, $Q = 500$ litres/s $= 0.5$ m^3/s

Length of pipe, $L = 1000$ m

Velocity, $V = \dfrac{Q}{\text{Area}} = \dfrac{0.5}{\frac{\pi}{4} d^2} = \dfrac{0.5 \times 4}{\pi \times 0.3^2} = 7.073$ m/s

$\therefore$ Reynolds number, $R_e = \dfrac{V \times d}{\nu} = \dfrac{7.073 \times 0.3}{0.29 \times 10^{-4}} = 7.316 \times (10)^4$

$\therefore$ Co-efficient of friction, $f = \dfrac{.079}{R_e^{1/4}} = \dfrac{0.79}{\left(7.316 \times 10^4\right)^{1/4}} = .0048$

$\therefore$ Head lost due to friction, $h_f = \dfrac{4 \times f \times L \times V^2}{d \times 2g} = \dfrac{4 \times .0048 \times 1000 \times 7.073^2}{0.3 \times 2 \times 9.81} = 163.18$ m

Power required $= \dfrac{\rho g \,.\, Q \,.\, h_f}{1000}$ kW

where ρ = density of oil $= 0.7 \times 1000 = 700$ kg/m^3

$\therefore$ Power required $= \dfrac{700 \times 9.81 \times 0.5 \times 163.18}{1000}$ = **560.28 kW. Ans.**

Problem 11.5 *Calculate the discharge through a pipe of diameter 200 mm when the difference of pressure head between the two ends of a pipe 500 m apart is 4 m of water. Take the value of 'f' = 0.009 in the formula* $h_f = \dfrac{4 \,.\, f \,.\, L \,.\, V^2}{d \times 2g}$.

Solution. Given :

Dia. of pipe, $d = 200$ mm $= 0.20$ m

Length of pipe, $L = 500$ m

Difference of pressure head, $h_f = 4$ m of water

$f = .009$

Using equation (11.1), we have $h_f = \dfrac{4 \times f \times L \times V^2}{d \times 2g}$

or $4.0 = \dfrac{4 \times .009 \times 500 \times V^2}{0.2 \times 2 \times 9.81}$ or $V^2 = \dfrac{4.0 \times 0.2 \times 2 \times 9.81}{4.0 \times .009 \times 500} = 0.872$

$\therefore \quad V = \sqrt{0.872} = 0.9338 \simeq 0.934$ m/s

$\therefore$ Discharge, Q = velocity × area

$$= 0.934 \times \frac{\pi}{4} d^2 = 0.934 \times \frac{\pi}{4} (0.2)^2$$

$$= 0.0293 \text{ m}^3/\text{s} = \textbf{29.3 litres/s. Ans.}$$

Problem 11.6 *Water is flowing through a pipe of diameter 200 mm with a velocity of 3 m/s. Find the head lost due to friction for a length of 5 m if the co-efficient of friction is given by* $f = 0.02 + \frac{.09}{R_e^{0.3}}$, *where* R_e *is Reynolds number. The kinematic viscosity of water = .01 stoke.*

Solution. Given :

Dia. of pipe, $d = 200$ mm $= 0.20$ m

Velocity, $V = 3$ m/s

Length, $L = 5$ m

Kinematic viscosity, $\nu = 0.01$ stoke $= .01 \times 10^{-4}$ m^2/s

$\therefore$ Reynolds number, $R_e = \dfrac{V \times d}{\nu} = \dfrac{3 \times 0.20}{.01 \times 10^{-4}} = 6 \times 10^5$

Value of $f = .02 + \dfrac{0.9}{R_e^{0.3}} = .02 + \dfrac{.09}{\left(6 \times 10^5\right)^{.3}} = .02 + \dfrac{0.09}{54.13}$

$$= .02 + .00166 = 0.02166$$

$\therefore$ Head lost due to friction, $h_f = \dfrac{4 \times f \times L \times V^2}{d \times 2g} = \dfrac{4.0 \times .02166 \times 5.0 \times 3^2}{0.20 \times 2.0 \times 9.81}$

$$= \textbf{0.993 m of water. Ans.}$$

Problem 11.7 *An oil of sp. gr. 0.9 and viscosity 0.06 poise is flowing through a pipe of diameter 200 mm at the rate of 60 litres/s. Find the head lost due to friction for a 500 m length of pipe. Find the power required to maintain this flow.*

Solution. Given :

Sp. gr. of oil $= 0.9$

Viscosity, $\mu = 0.06$ poise $= \dfrac{0.06}{10}$ Ns/m^2

Dia. of pipe, $d = 200$ mm $= 0.2$ m

Discharge, $Q = 60$ litres/s $= 0.06$ m^3/s

Length, $L = 500$ m

Density $\rho = 0.9 \times 1000 = 900$ kg/m^3

$\therefore$ Reynolds number, $R_e = \dfrac{\rho V d}{\mu} = 900 \times \dfrac{V \times 0.2}{\frac{0.06}{10}}$

where $V = \dfrac{Q}{\text{Area}} = \dfrac{0.06}{\frac{\pi}{4} d^2} = \dfrac{0.06}{\frac{\pi}{4}(.2)^2} = 1.909 \text{ m/s} \simeq 1.91 \text{ m/s}$

$$\therefore \qquad R_e = 900 \times \frac{1.91 \times 0.2 \times 10}{0.06} = 57300$$

As R_e lies between 4000 and 10^5, the value of co-efficient of friction, f is given by

$$f = \frac{0.079}{R_e^{0.25}} = \frac{0.079}{(57300)^{0.25}} = .0051$$

$$\therefore \text{ Head lost due to friction, } h_f = \frac{4 \times f \times L \times V^2}{d \times 2g} = \frac{4 \times .0051 \times 500 \times 1.91^2}{0.2 \times 2 \times 9.81}$$

$$= \textbf{9.48 m of water. Ans.}$$

$$\therefore \text{ Power required} = \frac{\rho g \cdot Q \cdot h_f}{1000} = \frac{900 \times 9.81 \times 0.06 \times 9.48}{1000} = \textbf{5.02 kW. Ans.}$$

▶ 11.4 MINOR ENERGY (HEAD) LOSSES

The loss of head or energy due to friction in a pipe is known as major loss while the loss of energy due to change of velocity of the following fluid in magnitude or direction is called minor loss of energy. The minor loss of energy (or head) includes the following cases :

1. Loss of head due to sudden enlargement,
2. Loss of head due to sudden contraction,
3. Loss of head at the entrance of a pipe,
4. Loss of head at the exit of a pipe,
5. Loss of head due to an obstruction in a pipe,
6. Loss of head due to bend in the pipe,
7. Loss of head in various pipe fittings.

In case of long pipe the above losses are small as compared with the loss of head due to friction and hence they are called minor losses and even may be neglected without serious error. But in case of a short pipe, these losses are comparable with the loss of head due to friction.

11.4.1 Loss of Head Due to Sudden Enlargement. Consider a liquid flowing through a pipe which has sudden enlargement as shown in Fig. 11.1. Consider two sections (1)-(1) and (2)-(2) before and after the enlargement.

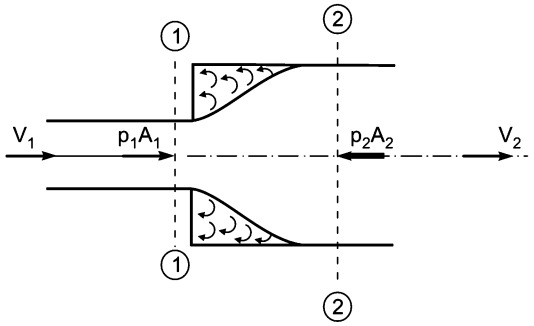


Fig. 11.1 *Sudden enlargement.*

Let p_1 = pressure intensity at section 1-1,
V_1 = velocity of flow at section 1-1,
A_1 = area of pipe at section 1-1,

p_2, V_2 and A_2 = corresponding values at section 2-2.

Due to sudden change of diameter of the pipe from D_1 to D_2, the liquid flowing from the smaller pipe is not able to follow the abrupt change of the boundary. Thus the flow separates from the boundary and turbulent eddies are formed as shown in Fig. 11.1. The loss of head (or energy) takes place due to the formation of these eddies.

Let p' = pressure intensity of the liquid eddies on the area $(A_2 - A_1)$

h_e = loss of head due to sudden enlargement

Applying Bernoulli's equation at sections 1-1 and 2-2,

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + \text{loss of head due to sudden enlargement}$$

But $z_1 = z_2$ as pipe is horizontal

$$\therefore \quad \frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + h_e$$

or
$$h_e = \left(\frac{p_1}{\rho g} - \frac{p_2}{\rho g}\right) + \left(\frac{V_1^2}{2g} - \frac{V_2^2}{2g}\right) \qquad ...(i)$$

Consider the control volume of liquid between sections 1-1 and 2-2. Then the force acting on the liquid in the control volume in the direction of flow is given by

$$F_x = p_1A_1 + p'(A_2 - A_1) - p_2A_2$$

But experimentally it is found that $p' = p_1$

$$\therefore \quad F_x = p_1A_1 + p_1(A_2 - A_1) - p_2A_2 = p_1A_2 - p_2A_2$$
$$= (p_1 - p_2)A_2 \qquad ...(ii)$$

Momentum of liquid/sec at section 1-1 = mass × velocity

$$= \rho A_1 V_1 \times V_1 = \rho A_1 V_1^2$$

Momentum of liquid/sec at section 2-2 = $\rho A_2V_2 \times V_2 = \rho A_2V_2^2$

$\therefore$ Change of momentum/sec = $\rho A_2V_2^2 - \rho A_1V_1^2$

But from continuity equation, we have

$$A_1V_1 = A_2V_2 \text{ or } A_1 = \frac{A_2V_2}{V_1}$$

$$\therefore \text{ Change of momentum/sec} = \rho A_2V_2^2 - \rho \times \frac{A_2V_2}{V_1} \times V_1^2 = \rho A_2V_2^2 - \rho A_2V_1V_2$$

$$= \rho A_2[V_2^2 - V_1V_2] \qquad ...(iii)$$

Now net force acting on the control volume in the direction of flow must be equal to the rate of change of momentum or change of momentum per second. Hence equating (*ii*) and (*iii*)

$$(p_1 - p_2)A_2 = \rho A_2[V_2^2 - V_1V_2]$$

or
$$\frac{p_1 - p_2}{\rho} = V_2^2 - V_1V_2$$

Dividing by g on both sides, we have $\frac{p_1 - p_2}{\rho g} = \frac{V_2^2 - V_1V_2}{g}$ or $\frac{p_1}{\rho g} - \frac{p_2}{\rho g} = \frac{V_2^2 - V_1V_2}{g}$

Substituting the value of $\left(\frac{p_1}{\rho g} - \frac{p_2}{\rho g}\right)$ in equation (*i*), we get

$$h_e = \frac{V_2^2 - V_1 V_2}{g} + \frac{V_1^2}{2g} - \frac{V_2^2}{2g} = \frac{2V_2^2 - 2V_1V_2 + V_1^2 - V_2^2}{2g}$$

$$= \frac{V_2^2 + V_1^2 - 2V_1V_2}{2g} = \left(\frac{V_1 - V_2}{2g}\right)^2$$

$$\therefore \qquad h_e = \frac{(V_1 - V_2)^2}{2g}. \qquad \text{...(11.5)}$$

11.4.2 Loss of Head due to Sudden Contraction. Consider a liquid flowing in a pipe which has a sudden contraction in area as shown in Fig. 11.2. Consider two sections 1-1 and 2-2 before and after contraction. As the liquid flows from large pipe to smaller pipe, the area of flow goes on decreasing and becomes minimum at a section *C-C* as shown in Fig. 11.2. This section *C-C* is called Vena-contracta. After section *C-C,* a sudden enlargement of the area takes place. The loss of head due to sudden contraction is actually due to sudden enlargement from Vena-contracta to smaller pipe.

Let A_c = Area of flow at section *C-C*

V_c = Velocity of flow at section *C-C*

A_2 = Area of flow at section 2-2

V_2 = Velocity of flow at section 2-2

h_c = Loss of head due to sudden contraction.

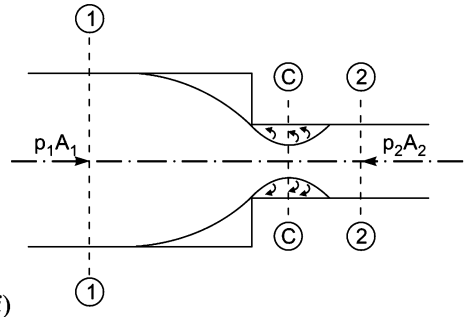


Fig. 11.2 *Sudden contraction.*

Now h_c = actual loss of head due to enlargement from section *C-C* to section 2-2 and is given by equation (11.5) as

$$= \frac{(V_c - V_2)^2}{2g} = \frac{V_2^2}{2g}\left[\frac{V_c}{V_2} - 1\right]^2 \qquad \text{...(}i\text{)}$$

From continuity equation, we have

$$A_c V_c = A_2 V_2 \quad \text{or} \quad \frac{V_c}{V_2} = \frac{A_2}{A_c} = \frac{1}{(A_c / A_2)} = \frac{1}{C_c} \qquad \left[\because C_c = \frac{A_c}{A_2}\right]$$

Substituting the value of $\frac{V_c}{V_2}$ in (*i*), we get

$$h_c = \frac{V_2^2}{2g}\left[\frac{1}{C_c} - 1\right]^2 \qquad \text{...(11.6)}$$

$$= \frac{kV_2^2}{2g}, \text{ where } k = \left[\frac{1}{C_c} - 1\right]^2$$

If the value of C_c is assumed to be equal to 0.62, then

$$k = \left[\frac{1}{0.62} - 1\right]^2 = 0.375$$

Then h_c becomes as $$h_c = \frac{kV_2^2}{2g} = 0.375\ \frac{V_2^2}{2g}$$

If the value of C_c is not given then the head loss due to contraction is taken as

$$= 0.5\ \frac{V_2^2}{2g} \text{ or } h_c = 0.5\ \frac{V_2^2}{2g}. \qquad ...(11.7)$$

Problem 11.8 *Find the loss of head when a pipe of diameter 200 mm is suddenly enlarged to a diameter of 400 mm. The rate of flow of water through the pipe is 250 litres/s.*

Solution. Given :

Dia. of smaller pipe, D_1= 200 mm = 0.20 m

$\therefore$ Area, $$A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} (.2)^2 = 0.03141 \text{ m}^2$$

Dia. of large pipe, $$D_2 = 400 \text{ mm} = 0.4 \text{ m}$$

$\therefore$ Area, $$A_2 = \frac{\pi}{4} \times (0.4)^2 = 0.12564 \text{ m}^2$$

Discharge, $$Q = 250 \text{ litres/s} = 0.25 \text{ m}^3/\text{s}$$

Velocity, $$V_1 = \frac{Q}{A_1} = \frac{0.25}{.03141} = 7.96 \text{ m/s}$$

Velocity, $$V_2 = \frac{Q}{A_2} = \frac{0.25}{.12564} = 1.99 \text{ m/s}$$

Loss of head due to enlargement is given by equation (11.5) as

$$h_e = \frac{(V_1 - V_2)^2}{2g} = \frac{(7.96 - 1.99)^2}{2g} = \textbf{1.816 m of water. Ans.}$$

Problem 11.9 *At a sudden enlargement of a water main from 240 mm to 480 mm diameter, the hydraulic gradient rises by 10 mm. Estimate the rate of flow.* (J.N.T.U., S 2002)

Solution. Given :

Dia. of smaller pipe, $$D_1 = 240 \text{ mm} = 0.24 \text{ m}$$

$\therefore$ Area, $$A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4} (.24)^2$$

Dia. of large pipe, $$D_2 = 480 \text{ mm} = 0.48 \text{ m}$$

$\therefore$ Area, $$A_2 = \frac{\pi}{4} (0.48)^2$$

Rise of hydraulic gradient*, *i.e.*, $\left(z_2 + \frac{p_2}{\rho g}\right) - \left(\frac{p_1}{\rho g} + z_1\right) = 10 \text{ mm} = \frac{10}{1000} = \frac{1}{100} \text{ m}$

Let the rate of flow = Q

Applying Bernoulli's equation to both sections, *i.e.*, smaller pipe section, and large pipe section.

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + \text{Head loss due to enlargement} \qquad ...(i)$$

* Please refer Art. 11.5.1.

But head loss due to enlargement,

$$h_e = \frac{(V_1 - V_2)^2}{2g} \qquad ...(ii)$$

From continuity equation, we have $A_1V_1 = A_2V_2$

$$\therefore \quad V_1 = \frac{A_2V_2}{A_1} = \frac{\frac{\pi}{4}D_2^2 \times V_2}{\frac{\pi}{4}D_1^2} = \left(\frac{D_2}{D_1}\right)^2 \times V_2 = \left(\frac{.48}{.24}\right)^2 \times V_2 = 2^2 \times V_2 = 4V_2$$

Substituting this value in (*ii*), we get

$$h_e = \frac{(4V_2 - V_2)^2}{2g} = \frac{(3V_2)^2}{2g} = \frac{9V_2^2}{2g}$$

Now substituting the value of h_e and V_1 in equation (*i*),

$$\frac{p_1}{\rho g} + \frac{(4V_2)^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + \frac{9V_2^2}{2g}$$

or $$\frac{16V_2^2}{2g} - \frac{V_2^2}{2g} - \frac{9V_2^2}{2g} = \left(\frac{p_2}{\rho g} + z_2\right) - \left(\frac{p_1}{\rho g} + z_1\right)$$

But hydraulic gradient rise $= \left(\frac{p_2}{\rho g} + z_2\right) - \left(\frac{p_1}{\rho g} + z_1\right) = \frac{1}{100}$

$$\therefore \quad \frac{16V_2^2}{2g} - \frac{V_2^2}{2g} - \frac{9V_2^2}{2g} = \frac{1}{100} \text{ or } \frac{6V_2^2}{2g} = \frac{1}{100}$$

$$\therefore \quad V_2 = \sqrt{\frac{2 \times 9.81}{6 \times 100}} = 0.1808 \simeq 0.181 \text{ m/s}$$

$\therefore$ Discharge, $Q = A_2 \times V_2$

$$= \frac{\pi}{4} D_2^2 \times V_2 = \frac{\pi}{4} (.48)^2 \times .181 = 0.03275 \text{ m}^3/\text{s}$$

$$= \mathbf{32.75 \text{ litres/s. Ans.}}$$

Problem 11.10 *The rate of flow of water through a horizontal pipe is 0.25 m^3/s. The diameter of the pipe which is 200 mm is suddenly enlarged to 400 mm. The pressure intensity in the smaller pipe is 11.772 N/cm^2. Determine :*

(*i*) *loss of head due to sudden enlargement,* (*ii*) *pressure intensity in the large pipe,*
(*iii*) *power lost due to enlargement.*

Solution. Given :

Discharge, $Q = 0.25 \text{ m}^3/\text{s}$

Dia. of smaller pipe, $D_1 = 200 \text{ mm} = 0.20 \text{ m}$

$\therefore$ Area, $A_1 = \frac{\pi}{4}(.2)^2 = 0.03141 \text{ m}^2$

Dia. of large pipe, $D_2 = 400 \text{ mm} = 0.40 \text{ m}$

$\therefore$ Area, $A_2 = \frac{\pi}{4}(0.4)^2 = 0.12566 \text{ m}^2$

Pressure in smaller pipe, $p_1 = 11.772 \text{ N/cm}^2 = 11.772 \times 10^4 \text{ N/m}^2$

Now velocity, $V_1 = \frac{Q}{A_1} = \frac{0.25}{.03141} = 7.96 \text{ m/s}$

Velocity, $V_2 = \frac{Q}{A_2} = \frac{0.25}{.12566} = 1.99 \text{ m/s}$

(*i*) Loss of head due to sudden enlargement,

$$h_e = \frac{(V_1 - V_2)^2}{2g} = \frac{(7.96 - 1.99)^2}{2 \times 9.81} = \mathbf{1.816 \text{ m. Ans.}}$$

(*ii*) Let the pressure intensity in large pipe = p_2.

Then applying Bernoulli's equation before and after the sudden enlargement,

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + h_e$$

But $z_1 = z_2$ (Given horizontal pipe)

$\therefore$ $$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + h_e \text{ or } \frac{p_2}{\rho g} = \frac{p_1}{\rho g} + \frac{V_1^2}{2g} - \frac{V_2^2}{2g} - h_e$$

$$= \frac{11.772 \times 10^4}{1000 \times 9.81} + \frac{7.96^2}{2 \times 9.81} - \frac{1.99^2}{2 \times 9.81} - 1.816$$

$$= 12.0 + 3.229 - 0.2018 - 1.8160$$

$$= 15.229 - 2.0178 = 13.21 \text{ m of water}$$

$\therefore$ $$p_2 = 13.21 \times \rho g = 13.21 \times 1000 \times 9.81 \text{ N/m}^2$$

$$= 13.21 \times 1000 \times 9.81 \times 10^{-4} \text{ N/cm}^2 = \mathbf{12.96 \text{ N/cm}^2. \text{ Ans.}}$$

(*iii*) Power lost due to sudden enlargement,

$$P = \frac{\rho g \cdot Q \cdot h_e}{1000} = \frac{1000 \times 9.81 \times 0.25 \times 1.816}{1000} = \mathbf{4.453 \text{ kW. Ans.}}$$

Problem 11.11 *A horizontal pipe of diameter 500 mm is suddenly contracted to a diameter of 250 mm. The pressure intensities in the large and smaller pipe is given as 13.734 N/cm^2 and 11.772 N/cm^2 respectively. Find the loss of head due to contraction if $C_c = 0.62$. Also determine the rate of flow of water.*

Solution. Given :

Dia. of large pipe, $D_1 = 500 \text{ mm} = 0.5 \text{ m}$

Area, $A_1 = \frac{\pi}{4}(0.5)^2 = 0.1963 \text{ m}^2$

Dia. of smaller pipe, $D_2 = 250 \text{ mm} = 0.25 \text{ m}$

$\therefore$ Area, $A_2 = \frac{\pi}{4}(.25)^2 = 0.04908 \text{ m}^2$

Pressure in large pipe, $p_1 = 13.734 \text{ N/cm}^2 = 13.734 \times 10^4 \text{ N/m}^2$

Pressure in smaller pipe, $p_2 = 11.772 \text{ N/cm}^2 = 11.772 \times 10^4 \text{ N/m}^2$

$C_c = 0.62$

Head lost due to contraction $= \frac{V_2^2}{2g}\left[\frac{1}{C_c} - 1.0\right]^2 = \frac{V_2^2}{2g}\left[\frac{1}{0.62} - 1.0\right]^2 = 0.375\,\frac{V_2^2}{2g}$

From continuity equation, we have $A_1V_1 = A_2V_2$

or $$V_1 = \frac{A_2V_2}{A_1} = \frac{\frac{\pi}{4}D_2^2 \times V_2}{\frac{\pi}{4}D_1^2} = \left(\frac{D_2}{D_1}\right)^2 \times V_2 = \left(\frac{0.25}{0.50}\right)^2 V_2 = \frac{V_2}{4}$$

Applying Bernoulli's equation before and after contraction,

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + h_c$$

But $z_1 = z_2$ (pipe is horizontal)

$\therefore$ $$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + h_c$$

But $h_c = 0.375\,\frac{V_2^2}{2g}$ and $V_1 = \frac{V_2}{4}$

Substituting these values in the above equation, we get

$$\frac{13.734 \times 10^4}{9.81 \times 1000} + \frac{(V_2/4)^2}{2g} = \frac{11.772 \times 10^4}{1000 \times 9.81} + \frac{V_2^2}{2g} + 0.375\,\frac{V_2^2}{2g}$$

or $$14.0 + \frac{V_2^2}{16 \times 2g} = 12.0 + 1.375\,\frac{V_2^2}{2g}$$

or $$14 - 12 = 1.375\,\frac{V_2^2}{2g} - \frac{1}{16}\,\frac{V_2^2}{2g} = 1.3125\,\frac{V_2^2}{2g}$$

or $$2.0 = 1.3125 \times \frac{V_2^2}{2g} \text{ or } V_2 = \sqrt{\frac{2.0 \times 2 \times 9.81}{1.3125}} = 5.467 \text{ m/s.}$$

(*i*) Loss of head due to contraction, $h_c = 0.375\,\frac{V_2^2}{2g} = \frac{0.375 \times (5.467)^2}{2 \times 9.81}$ = **0.571 m. Ans.**

(*ii*) Rate of flow of water, $Q = A_2V_2 = 0.04908 \times 5.467 = 0.2683 \text{ m}^3/\text{s}$ = **268.3 lit/s. Ans.**

Problem 11.12 *If in the problem 11.11, the rate of flow of water is 300 litres/s, other data remaining the same, find the value of co-efficient of contraction, C_c.*

Solution. Given :

$$D_1 = 0.5 \text{ m}, D_2 = 0.25 \text{ m}, p_1 = 13.734 \times 10^4 \text{ N/m}^2,$$

$$p_2 = 11.772 \times 10^4 \text{ N/m}^2, Q = 300 \text{ lit/s} = 0.3 \text{ m}^3/\text{s}$$

Also from Problem 11.11, $V_1 = \frac{V_2}{4}$, where $V_1 = \frac{Q}{A_1} = \frac{0.30}{\frac{\pi}{4}(0.5)^2} = 1.528$ m/s

$\therefore$ $V_2 = 4 \times V_1 = 4 \times 1.528 = 6.112$ m/s

From Bernoulli's equation, we have

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + h_c. \qquad [\because z_1 = z_2]$$

or $$\frac{13.734 \times 10^4}{9.81 \times 1000} + \frac{(1.528)^2}{2 \times 9.81} = \frac{11.772 \times (10)^4}{9.81 \times 1000} + \frac{(6.112)^2}{2 \times 9.81} + h_c$$

or $$14.0 + 0.119 = 12.0 + 1.904 + h_c$$

$$14.119 = 13.904 + h_c$$

$\therefore$ $$h_c = 14.119 - 13.904 = 0.215$$

But from equation (11.6), $h_c = \frac{V_2^2}{2g}\left[\frac{1}{C_c} - 1\right]^2$

Hence equating the two values of h_c, we get

$$\frac{V_2^2}{2g}\left[\frac{1}{C_c} - 1\right]^2 = 0.215$$

$$V_2 = 6.112, \therefore \frac{6.112^2}{2 \times 9.81}\left[\frac{1}{C_c} - 1\right]^2 = 0.215$$

or $$\left[\frac{1}{C_c} - 1\right]^2 = \frac{0.215 \times 2.0 \times 9.81}{6.112 \times 6.112} = 0.1129$$

or $$\frac{1.0}{C_c} - 1.0 = \sqrt{0.1129} = 0.336 \text{ or } \frac{1.0}{C_c} = 1.0 + 0.336 = 1.336$$

$\therefore$ $$C_c = \frac{1.0}{1.336} = \mathbf{0.748. \text{ Ans.}}$$

Problem 11.13 *A 150 mm diameter pipe reduces in diameter abruptly to 100 mm diameter. If the pipe carries water at 30 litres per second, calculate the pressure loss across the contraction. Take the co-efficient of contraction as 0.6.*

Solution. Given :

Dia. of large pipe, $D_1 = 150 \text{ mm} = 0.15 \text{ m}$

Area of large pipe, $A_1 = \frac{\pi}{4} (.15)^2 = 0.01767 \text{ m}^2$

Dia. of smaller pipe, $D_2 = 100 \text{ mm} = 1.10 \text{ m}$

Area of smaller pipe, $A_2 = \frac{\pi}{4} (.10)^2 = 0.007854 \text{ m}^2$

Discharge, $Q = 30 \text{ litres/s} = .03 \text{ m}^3/\text{s}$

Co-efficient of contraction, $C_c = 0.6$

From continuity equation, we have $A_1V_1 = A_2V_2 = Q$

$$\therefore \quad V_1 = \frac{Q}{A_1} = \frac{0.03}{.01767} = 1.697 \text{ m/s}$$

and

$$V_2 = \frac{Q}{A_2} = \frac{.03}{.007854} = 3.82 \text{ m/s}$$

Applying Bernoulli's equation before and after contraction,

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + Z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + Z_2 + h_c \qquad ...(i)$$

But $Z_1 = Z_2$

and h_c, the head loss due to contraction is given by equation (11.6) as

$$h_c = \frac{V_2^2}{2g}\left[\frac{1}{C_c} - 1\right]^2 = \frac{3.82^2}{2 \times 9.81}\left[\frac{1}{0.6} - 1\right]^2 = 0.33$$

Substituting these values in equation (i), we get

$$\frac{p_1}{\rho g} + \frac{1.697^2}{2 \times 9.81} = \frac{p_2}{\rho g} + \frac{3.82^2}{2 \times 9.81} + 0.33$$

or

$$\frac{p_1}{\rho g} + 0.1467 = \frac{p_2}{\rho g} + .7438 + .33$$

$$\therefore \quad \frac{p_1}{\rho g} - \frac{p_2}{\rho g} = .7438 + .33 - .1467 = 0.9271 \text{ m of water}$$

$$\therefore \quad (p_1 - p_2) = \rho g \times 0.9271 = 1000 \times 9.81 \times 0.9271 \text{ N/m}^2$$

$$= 0.909 \times 10^4 \text{ N/m}^2 = 0.909 \text{ N/cm}^2$$

$\therefore$ Pressure loss across contraction

$= p_1 - p_2 =$ **0.909 N/cm^2. Ans.**

Problem 11.14 *In Fig. 11.3 below, when a sudden contraction is introduced in a horizontal pipe line from 50 cm to 25 cm, the pressure changes from 10,500 kg/m^2 (103005 N/m^2) to 6900 kg/m^2 (67689 N/m^2). Calculate the rate of flow. Assume co-efficient of contraction of jet to be 0.65.*

Following this if there is a sudden enlargement from 25 cm to 50 cm and if the pressure at the 25 cm section is 6900 kg/m^2 (67689 N/m^2) what is the pressure at the 50 cm enlarged section ?

Solution. Given :

Dia. of large pipe, $D_1 = 50 \text{ cm} = 0.5 \text{ m}$

Area, $A_1 = \frac{\pi}{4} (.5)^2 = 0.1963 \text{ m}^2$

Dia. of smaller pipe, $D_2 = 25 \text{ cm} = 0.25 \text{ m}$

$\therefore$ Area, $A_2 = \frac{\pi}{4} (.25)^2 = 0.04908 \text{ m}^2$

Pressure in large pipe, $p_1 = 10500 \text{ kg/m}^2$ or 103005 N/m^2

Pressure in smaller pipe, $p_2 = 6900 \text{ kg/m}^2$ or 67689 N/m^2

Co-efficient of contraction, $C_c = 0.65$

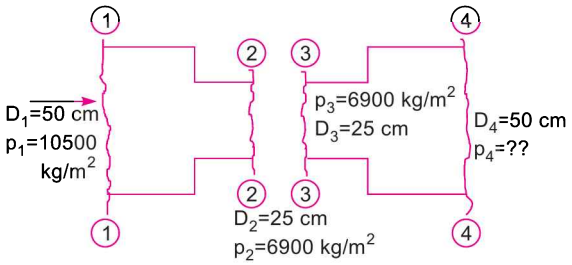


Fig. 11.3

Head lost due to contraction is given by equation (11.6),

$$h_c = \frac{V_2^2}{2g}\left(\frac{1}{C_c} - 1.0\right)^2 = \frac{V_2^2}{2g}\left(\frac{1}{0.65} - 1\right)^2 = 0.2899 \frac{V_2^2}{2g} \qquad ...(i)$$

From continuity equation, we have

$$A_1V_1 = A_2V_2 \text{ or } V_1 = \frac{A_2V_2}{A_1} = \frac{\frac{\pi}{4}D_2^2 \times V_2}{\frac{\pi}{4}D_1^2}$$

$$= \left(\frac{D_2}{D_1}\right)^2 \times V_2 = \left(\frac{0.50}{0.25}\right)^2 \times V_2 = \frac{V_2}{4} \qquad ...(ii)$$

Applying Bernoulli's equation at sections 1-1 and 2-2,

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + Z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + Z_2 + h_c$$

But $Z_1 = Z_2$ (as pipe is horizontal)

$$\therefore \quad \frac{p_1}{\rho g} + \frac{V_1^2}{2g} = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + h_c$$

Substituting the values of p_1, p_2, h_c and V_1, we get

$$\frac{103005}{1000 \times 9.81} + \frac{(V_2/4)^2}{2g} = \frac{67689}{1000 \times 9.81} + \frac{V_2^2}{2g} + .2899\ \frac{V_2^2}{2g}$$

or $$10.5 + \frac{V_2^2}{16 \times 2g} = 6.9 + 1.2899\ \frac{V_2^2}{2g}$$

or $$10.5 - 6.9 = 1.2899\ \frac{V_2^2}{2g} - \frac{1}{16} \times \frac{V_2^2}{2g} = 1.2274\ \frac{V_2^2}{2g}$$

or $$3.6 = 1.2274 \times \frac{V_2^2}{2g}$$

$$\therefore \quad V_2 = \sqrt{\frac{3.6 \times 2 \times 9.81}{1.2274}} = 7.586 \text{ m/s}$$

(*i*) Rate of flow of water, $Q = A_2 V_2 = 0.04908 \times 7.586$

$$= \mathbf{0.3723\ m^3/s \text{ or } 372.3 \text{ lit/s. Ans.}}$$

(*ii*) Applying Bernoulli's equation to sections 3-3 and 4-4,

$$\frac{p_3}{\rho g} + \frac{V_3^2}{2g} + Z_3 = \frac{p_4}{\rho g} + \frac{V_4^2}{2g} + Z_4 + \text{head loss due to sudden enlargement } (h_e)$$

But $$p_3 = 6900 \text{ kg/m}^2, \text{ or } 67689 \text{ N/m}^2$$

$$V_3 = V_2 = 7.586 \text{ m/s}$$

$$V_4 = V_1 = \frac{V_2}{4} = \frac{7.586}{4} = 1.8965$$

$$Z_3 = Z_4$$

And head loss due to sudden enlargement is given by equation (11.5) as

$$h_e = \frac{(V_3 - V_4)^2}{2g} = \frac{(7.586 - 1.8965)^2}{2 \times 9.81} = 1.65 \text{ m}$$

Substituting these values in Bernoulli's equation, we get

$$\frac{67689}{1000 \times 9.81} + \frac{7.586^2}{2 \times 9.81} = \frac{p_4}{1000 \times 9.81} + \frac{1.8965^2}{2 \times 9.81} + 1.65$$

or $$6.9 + 2.933 = \frac{p_4}{1000 \times 9.81} + 0.183 + 1.65$$

$$\therefore \quad \frac{p_4}{1000 \times 9.81} = 6.9 + 2.933 - 0.183 - 1.65 = 9.833 - 1.833 = 8.00$$

$$\therefore \quad p_4 = 8 \times 1000 \times 9.81 = \mathbf{78480\ N/m^2.\ Ans.}$$

11.4.3 Loss of Head at the Entrance of a Pipe. This is the loss of energy which occurs when a liquid enters a pipe which is connected to a large tank or reservoir. This loss is similar to the loss of head due to sudden contraction. This loss depends on the form of entrance. For a sharp edge entrance, this loss is slightly more than a rounded or bell mouthed entrance. In practice the value of loss of head at the entrance (or inlet) of a pipe with sharp cornered entrance is taken = $0.5 \frac{V^2}{2g}$, where V = velocity of liquid in pipe.

This loss is denoted by h_i

$$\therefore \quad h_i = 0.5 \frac{V^2}{2g} \qquad \text{...(11.8)}$$

11.4.4 Loss of Head at the Exit of Pipe. This is the loss of head (or energy) due to the velocity of liquid at outlet of the pipe which is dissipated either in the form of a free jet (if outlet of the pipe is free) or it is lost in the tank or reservoir (if the outlet of the pipe is connected to the tank or reservoir). This loss is equal to $\frac{V^2}{2g}$, where V is the velocity of liquid at the outlet of pipe. This loss is denoted h_o.

$$\therefore \quad h_o = \frac{V^2}{2g} \qquad \text{...(11.9)}$$

where V = velocity at outlet of pipe.

11.4.5 Loss of Head Due to an Obstruction in a Pipe. Whenever there is an obstruction in a pipe, the loss of energy takes place due to reduction of the area of the cross-section of the pipe at the place where obstruction is present. There is a sudden enlargement of the area of flow beyond the obstruction due to which loss of head takes place as shown in Fig. 11.3 (a)

Consider a pipe of area of cross-section A having an obstruction as shown in Fig. 11.3 (a).

Let a = Maximum area of obstruction
A = Area of pipe
V = Velocity of liquid in pipe

Then $(A - a)$ = Area of flow of liquid at section 1-1.

As the liquid flows and passes through section 1-1, a vena-contracta is formed beyond section 1-1, after which the stream of liquid widens again and velocity of flow at section 2-2 becomes uniform and equal to velocity, V in the pipe. This situation is similar to the flow of liquid through sudden enlargement.

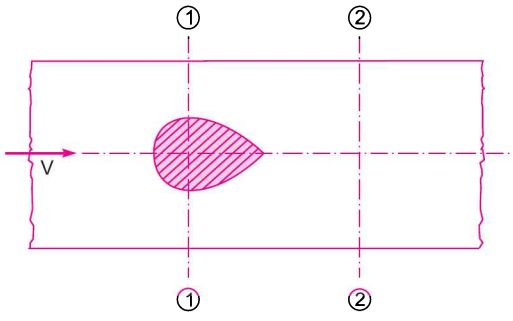


Fig. 11.3 (a) *An obstruction in a pipe.*

Let V_c = Velocity of liquid at vena-contracta.

Then loss of head due to obstruction = loss of head due to enlargement from vena-contracta to section 2-2.

$$= \frac{(V_c - V)^2}{2g} \qquad ...(i)$$

From continuity, we have $a_c \times V_c = A \times V$ ...(*ii*)

where a_c = area of cross-section at vena-contracta

If C_c = co-efficient of contraction

Then $$C_c = \frac{\text{area at vena - contracta}}{(A - a)} = \frac{a_c}{(A - a)}$$

$\therefore$ $$a_c = C_c \times (A - a)$$

Substituting this value in (*ii*), we get

$$C_c \times (A - a) \times V_c = A \times V \quad \therefore \quad V_c = \frac{A \times V}{C_c (A - a)}$$

Substituting this value of V_c in equation (*i*), we get

$$\text{Head loss due to obstruction} = \frac{(V_c - V)^2}{2g} = \frac{\left(\frac{A \times V}{C_c (A - a)} - V\right)^2}{2g} = \frac{V^2}{2g}\left(\frac{A}{C_c (A - a)} - 1\right)^2 \quad ...(11.10)$$

11.4.6 Loss of Head due to Bend in Pipe. When there is any bend in a pipe, the velocity of flow changes, due to which the separation of the flow from the boundary and also formation of eddies takes place. Thus the energy is lost. Loss of head in pipe due to bend is expressed as

$$h_b = \frac{kV^2}{2g}$$

where h_b = loss of head due to bend, V = velocity of flow, k = co-efficient of bend

The value of k depends on

(*i*) Angle of bend, (*ii*) Radius of curvature of bend, (*iii*) Diameter of pipe.

11.4.7 Loss of Head in Various Pipe Fittings. The loss of head in the various pipe fittings such as valves, couplings etc., is expressed as

$$= \frac{kV^2}{2g} \qquad ...(11.11)$$

where V = velocity of flow, k = co-efficient of pipe fitting.

Problem 11.15 *Water is flowing through a horizontal pipe of diameter 200 mm at a velocity of 3 m/s. A circular solid plate of diameter 150 mm is placed in the pipe to obstruct the flow. Find the loss of head due to obstruction in the pipe if $C_c = 0.62$.*

Solution. Given :

Dia. of pipe, $D = 200 \text{ mm} = 0.20 \text{ m}$

Velocity, $V = 3.0 \text{ m/s}$

Area of pipe, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} (0.2)^2 = 0.03141 \text{ m}^2$

Dia. of obstruction, $d = 150 \text{ mm} = 0.15 \text{ m}$

$\therefore$ Area of obstruction, $a = \frac{\pi}{4} (.15)^2 = 0.01767 \text{ m}^2$

$$C_c = 0.62$$

The head lost due to obstruction is given by equation (11.10) as

$$= \frac{V^2}{2g}\left(\frac{A}{C_c\,(A-a)} - 1.0\right)^2$$

$$= \frac{3\times 3}{2\times 9.81}\left[\frac{.03141}{0.62\,[.03141 - .01767]} - 1.0\right]^2$$

$$= \frac{9}{2\times 9.81}\,[3.687 - 1.0]^2 = \mathbf{3.311\ m.\ Ans.}$$

Problem 11.16 *Determine the rate of flow of water through a pipe of diameter 20 cm and length 50 m when one end of the pipe is connected to a tank and other end of the pipe is open to the atmosphere. The pipe is horizontal and the height of water in the tank is 4 m above the centre of the pipe. Consider all minor losses and take f = .009 in the formula* $h_f = \dfrac{4\,.\,f\,.\,L\,.\,V^2}{d\times 2g}$.

Solution. Dia. of pipe, $d = 20$ cm $= 0.20$ m

Length of pipe, $L = 50$ m

Height of water, $H = 4$ m

Co-efficient of friction, $f = .009$

Let the velocity of water in pipe $= V$ m/s.

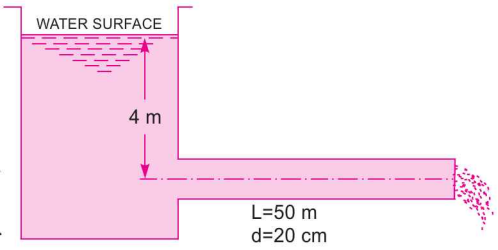


Fig. 11.4

Applying Bernoulli's equation at the top of the water surface in the tank and at the outlet of pipe, we have [Taking point 1 on the top and point 2 at the outlet of pipe].

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + \text{all losses}$$

Considering datum line passing through the centre of pipe

$$0 + 0 + 4.0 = 0 + \frac{V_2^2}{2g} + 0 + (h_i + h_f)$$

or $$4.0 = \frac{V_2^2}{2g} + h_i + h_f$$

But the velocity in pipe $= V$, $\therefore V = V_2$

$\therefore$ $$4.0 = \frac{V^2}{2g} + h_i + h_f \qquad ...(i)$$

From equation (11.8), $h_i = 0.5\,\dfrac{V^2}{2g}$ and h_f from equation (11.1) is given as

$$h_f = \frac{4 \cdot f \cdot L \cdot V^2}{d \times 2g}$$

Substituting these values, we have

$$4.0 = \frac{V^2}{2g} + \frac{0.5\,V^2}{2g} + \frac{4 \cdot f \cdot L \cdot V^2}{d \times 2g}$$

$$= \frac{V^2}{2g}\left[1.0 + 0.5 + \frac{4 \times .009 \times 50}{0.2}\right] = \frac{V^2}{2g}\ [1.0 + 0.5 + 9.0]$$

$$= 10.5 \times \frac{V^2}{2g}$$

$\therefore$ $$V = \sqrt{\frac{4 \times 2 \times 9.81}{10.5}} = 2.734 \text{ m/sec}$$

$\therefore$ Rate of flow, $$Q = A \times V = \frac{\pi}{4} \times (0.2)^2 \times 2.734 = 0.08589 \text{ m}^3/\text{s}$$

$$= \mathbf{85.89\ litres/s.\ Ans.}$$

Problem 11.17 *A horizontal pipe line 40 m long is connected to a water tank at one end and discharges freely into the atmosphere at the other end. For the first 25 m of its length from the tank, the pipe is 150 mm diameter and its diameter is suddenly enlarged to 300 mm. The height of water level in the tank is 8 m above the centre of the pipe. Considering all losses of head which occur, determine the rate of flow. Take f = .01 for both sections of the pipe.*

Solution. Given :

Total length of pipe, $L = 40$ m

Length of 1st pipe, $L_1 = 25$ m

Dia. of 1st pipe, $d_1 = 150$ mm $= 0.15$ m

Length of 2nd pipe, $L_2 = 40 - 25 = 15$ m

Dia. of 2nd pipe, $d_2 = 300$ mm $= 0.3$ m

Height of water, $H = 8$ m

Co-efficient of friction, $f = 0.01$

Applying Bernoulli's theorem to the free surface of water in the tank and outlet of pipe as shown in Fig. 11.5 and taking reference line passing through the centre of pipe.

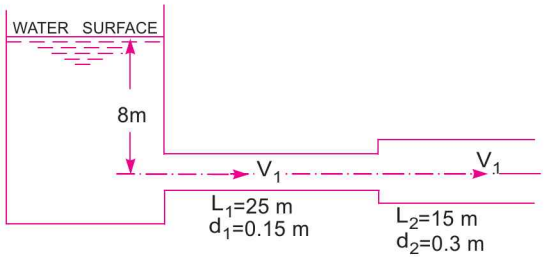


Fig. 11.5

$$0 + 0 + 8 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + 0 + \text{all losses}$$

or $$8.0 = 0 + \frac{V_2^2}{2g} + h_i + h_{f_1} + h_e + h_{f_2} \quad ...(i)$$

where h_i = loss of head at entrance = $0.5 \frac{V_1^2}{2g}$

h_{f_1} = head lost due to friction in pipe 1 = $\frac{4 \times f \times L_1 \times V_1^2}{d_1 \times 2g}$

h_e = loss head due to sudden enlargement = $\frac{(V_1 - V_2)^2}{2g}$

h_{f_2} = Head lost due to friction in pipe 2 = $\frac{4 \times f \times L_2 \times V_2^2}{d_2 \times 2g}$

But from continuity equation, we have

$$A_1V_1 = A_2V_2$$

$$\therefore \quad V_1 = \frac{A_2V_2}{A_1} = \frac{\frac{\pi}{4}d_2^2 \times V_2}{\frac{\pi}{4}d_1^2} = \left(\frac{d_2}{d_1}\right)^2 \times V_2 = \left(\frac{0.3}{.15}\right)^2 \times V_2 = 4V_2$$

Substituting the value of V_1 in different head losses, we have

$$h_i = \frac{0.5\, V_1^2}{2g} = \frac{0.5 \times (4V_2)^2}{2g} = \frac{8V_2^2}{2g}$$

$$h_{f_1} = \frac{4 \times 0.01 \times 25 \times (4V_2)^2}{0.15 \times 2 \times g}$$

$$= \frac{4 \times .01 \times 25 \times 16}{0.15} \times \frac{V_2^2}{2g} = 106.67 \frac{V_2^2}{2g}$$

$$h_e = \frac{(V_1 - V_2)^2}{2g} = \frac{(4V_2 - V_2)^2}{2g} = \frac{9V_2^2}{2g}$$

$$h_{f_2} = \frac{4 \times .01 \times 15 \times V_2^2}{0.3 \times 2g} = \frac{4 \times .01 \times 15}{0.3} \times \frac{V_2^2}{2g} = 2.0 \times \frac{V_2^2}{2g}$$

Substituting the values of these losses in equation (i), we get

$$8.0 = \frac{V_2^2}{2g} + \frac{8V_2^2}{2g} + 106.67 \frac{V_2^2}{2g} + \frac{9V_2^2}{2g} + 2 \times \frac{V_2^2}{2g}$$

$$= \frac{V_2^2}{2g} [1 + 8 + 106.67 + 9 + 2] = 126.67 \frac{V_2^2}{2g}$$

$$\therefore \quad V_2 = \sqrt{\frac{8.0 \times 2 \times g}{126.67}} = \sqrt{\frac{8.0 \times 2 \times 9.81}{126.67}} = \sqrt{1.2391} = 1.113 \text{ m/s}$$

$\therefore$ Rate of flow, $Q = A_2 \times V_2 = \frac{\pi}{4} (0.3)^2 \times 1.113 = 0.07867 \text{ m}^3/\text{s} =$ **78.67 litres/s. Ans.**

Problem 11.18 *Determine the difference in the elevations between the water surfaces in the two tanks which are connected by a horizontal pipe of diameter 300 mm and length 400 m. The rate of flow of water through the pipe is 300 litres/s. Consider all losses and take the value of f = .008.*

Solution. Given :

Dia. of pipe, $d = 300 \text{ mm} = 0.30 \text{ m}$

Length, $L = 400 \text{ m}$

Discharge, $Q = 300 \text{ lit/s} = 0.3 \text{ m}^3/\text{s}$

Co-efficient of friction, $f = 0.008$

Velocity, $$V = \frac{Q}{\text{Area}} = \frac{0.3}{\frac{\pi}{4} \times (.3)^2} = 4.244 \text{ m/s}$$

Let the two tanks are connected by a pipe as shown in Fig. 11.6.

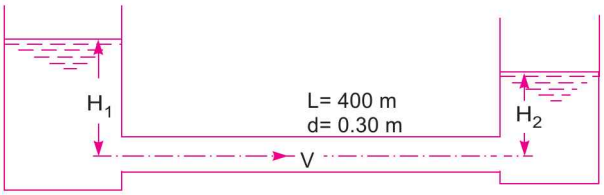


Fig. 11.6

Let H_1 = height of water in 1st tank above the centre of pipe

H_2 = height of water in 2nd tank above the centre of pipe

Then difference in elevations between water surfaces = $H_1 - H_2$

Applying Bernoulli's equation to the free surface of water in the two tanks, we have

$$H_1 = H_2 + \text{losses}$$

$$= H_2 + h_i + H_{f_1} + h_o \qquad ...(i)$$

where h_i = Loss of head at entrance $= 0.5 \dfrac{V^2}{2g} = \dfrac{0.5 \times 4.244^2}{2 \times 9.81} = 0.459 \text{ m}$

h_{f_1} = Loss of head due to friction $= \dfrac{4 \times f \times L \times V^2}{d \times 2g} = \dfrac{4 \times .008 \times 400 \times 4.244^2}{0.3 \times 2 \times 9.81} = 39.16 \text{ m}$

h_o = Loss of head at outlet $= \dfrac{V^2}{2g} = \dfrac{4.244^2}{2 \times 9.81} = 0.918 \text{ m}$

Substituting these values in (*i*), we get

$$H_1 = H_2 + 0.459 + 39.16 + 0.918 = H_2 + 40.537$$

$\therefore$ $H_1 - H_2$ = Difference in elevations

$$= \mathbf{40.537 \text{ m. Ans.}}$$

Problem 11.19 *The friction factor for turbulent flow through rough pipes can be determined by Karman-Prandtl equation,* $\dfrac{1}{\sqrt{f}} = 2 \log_{10} (R_0/k) + 1.74$

where f = friction factor, R_0 = pipe radius, k = average roughness.

Two reservoirs with a surface level difference of 20 metres are to be connected by 1 metre diameter pipe 6 km long. What will be the discharge when a cast iron pipe of roughness k = 0.3 mm is used ? What will be the percentage increase in the discharge if the cast iron pipe is replaced by a steel pipe of roughness k = 0.1 mm ? Neglect all local losses.

Solution. Given :

Difference in levels, $h = 20$ m

Dia. of pipe, $d = 1.0$ m $\quad \therefore$ Radius, $R_0 = 0.5$ m $= 500$ mm

Length of pipe, $L = 6$ km $= 6 \times 1000 = 6000$ m

Roughness of cast iron pipe, $k = 0.3$ mm

Roughness of steel pipe, $k = 0.1$ mm

1st Case. ***Cast Iron Pipe.*** First find the value of friction factor using

$$\frac{1}{\sqrt{f}} = 2 \log_{10} (R_0/k) + 1.74 \qquad ...(i)$$

$$= 2\log_{10} (500/0.3) + 1.74 = 8.1837$$

$$\therefore \qquad f = \left(\frac{1}{8.1837}\right)^2 = 0.0149$$

Local losses are to be neglected. This means only head loss due to friction is to be considered. Head loss due to friction is

$$20 = \frac{f \times L \times V^2}{d \times 2g}$$

[Here f is the friction factor and not co-efficient of friction

$\because$ Friction factor = 4 × co-efficient of friction]

$$20 = \frac{.0149 \times 6000 \times V^2}{1.0 \times 2 \times 9.81} = 4.556\ V^2$$

$$\therefore \qquad V = \sqrt{\frac{20}{4.556}} = 2.095 \text{ m/s}$$

$\therefore$ Discharge, $Q_1 = V \times \text{Area} = 2.095 \times \frac{\pi}{4} \times d^2 = 2.095 \times \frac{\pi}{4} \times 1^2 = 1.645 \text{ m}^3/\text{s}$

2nd Case. ***Steel Pipe.*** $k = 0.1$ mm, $R_0 = 500$ mm

Substituting these values in equation (i), we get

$$\frac{1}{\sqrt{f}} = 2 \log_{10} (500/0.1) + 1.74 = 9.1379$$

$$\therefore \qquad f = \left(\frac{1}{9.1379}\right)^2 = 0.0119$$

Head loss due to friction, $20 = \frac{f \times L \times V^2}{d \times 2g}$ or $20 = \frac{.0119 \times 6000 \times V^2}{1.0 \times 2 \times 9.81} = 3.639\ V^2$

$\therefore$ $$V = \sqrt{\frac{20}{3.639}} = 2.344 \text{ m/s}$$

$\therefore$ Discharge, $$Q_2 = V \times \text{Area} = 2.344 \times \frac{\pi}{4} \times 1^2 = 1.841 \text{ m}^3/\text{s}$$

percentage increase in the discharge $= \frac{Q_2 - Q_1}{Q_1} \times 100 = \frac{(1.841 - 1.645)}{1.645} \times 100 =$ **11.91%. Ans.**

Problem 11.20 *Design the diameter of a steel pipe to carry water having kinematic viscosity $\nu = 10^{-6}$ m^2/s with a mean velocity of 1 m/s. The head loss is to be limited to 5 m per 100 m length of pipe. Consider the equivalent sand roughness height of pipe $k_s = 45 \times 10^{-4}$ cm. Assume that the Darcy Weisbach friction factor over the whole range of turbulent flow can be expressed as*

$$f = 0.0055 \left[1 + \left(20 \times 10^3 \frac{k_s}{D} + \frac{10^6}{R_e} \right)^{1/3} \right]$$

where D = Diameter of pipe and R_e = Reynolds number.

Solution. Given :

Kinematic viscosity, $\nu = 10^{-6}$ m^2/s

Mean velocity, $V = 1$ m/s

Head loss, $h_f = 5$ m in a length $L = 100$ m

Value of $k_s = 45 \times 10^{-4}$ cm $= 45 \times 10^{-6}$ m

Value of $$f = 0.0055 \left[1 + \left(20 \times 10^3 \frac{k_s}{D} + \frac{10^6}{R_e} \right)^{1/3} \right] \quad \ldots(i)$$

Using Darcy Weisbach equation, $h_f = \frac{4 \times f \times L \times V^2}{D \times 2g}$

or $$f = \frac{h_f \times D \times 2g}{4 \times L \times V^2} = \frac{5 \times D \times 2 \times 9.81}{4 \times 100 \times 1^2} = 0.2452\, D$$

Now the Reynolds number is given by,

$$R_e = \frac{\rho VD}{\mu} = \frac{V \times D}{\nu} \qquad \left(\because \nu = \frac{\mu}{\rho} \right)$$

$$= \frac{1 \times D}{10^{-6}} = 10^6\, D$$

Substituting the values of f, R_e and k_s in equation (i), we get

$$0.2452\, D = 0.0055 \left[1 + \left(20 \times 10^3 \times \frac{45 \times 10^{-6}}{D} + \frac{10^6}{10^6\, D} \right)^{1/3} \right]$$

or $$\frac{0.2452}{0.0055} D = \left[1 + \left(\frac{0.9}{D} + \frac{1}{D} \right)^{1/3} \right]$$

or $$44.58\,D = \left[1 + \left(\frac{1.9}{D}\right)^{1/3}\right] \text{ or } 44.58\,D - 1 = \left(\frac{1.9}{D}\right)^{1/3}$$

or $$(44.58\,D - 1)^3 = \frac{1.9}{D} \text{ or } D\,(44.58\,D - 1)^3 = 1.9 \quad \ldots(ii)$$

Equation (*ii*) is solved by hit and trial method.

(*i*) Let $D = 0.1$ m, then L.H.S. of equation (*ii*) becomes as

L.H.S. $= 0.1\,(44.58 \times 0.1 - 1)^3 = 0.1 \times 3.458^3 = 4.135$

This is more than the R.H.S.

(*ii*) Let $D = 0.08$ m, then L.H.S. of equation (*ii*) becomes as

L.H.S. $= 0.08\,(44.58 \times 0.08 - 1)^3 = 0.08\,(2.5664)^3 = 1.352$

This is less than the R.H.S.

Hence value of D lies between 0.1 and 0.08

(*iii*) Let $D = 0.085$, then L.H.S. of equation (*ii*) becomes as

L.H.S. $= 0.085\,(44.58 \times 0.085 - 1)^3 = 1.844$

This value is slightly less than R.H.S. Hence increase the value of D slightly.

(*iv*) Let $D = 0.0854$ m, then L.H.S. of equation (*ii*) becomes as

L.H.S. $= 0.0854\,(44.58 \times 0.0854 - 1)^3 = 1.889$

This value is nearly equal to R.H.S.

$\therefore$ Correct value of D = **0.0854 m. Ans.**

Problem 11.21 *A pipe line AB of diameter 300 mm and of length 400 m carries water at the rate of 50 litres/s. The flow takes place from A to B where point B is 30 metres above A. Find the pressure at A if the pressure at B is 19.62 N/cm^2. Take $f = .008$.*

Solution. Given :

Dia. of pipe, $d = 300$ mm $= 0.30$ m

Length of pipe, $L = 400$ m

Discharge, $Q = 50$ litres/s $= 0.05$ m^3/s

$\therefore$ Velocity, $$V = \frac{Q}{\text{Area}} = \frac{0.05}{\frac{\pi}{4}d^2} = \frac{0.05}{\frac{\pi}{4} \times (.3)^2} = 0.7074 \text{ m/s}$$

Pressure at B, $p_B = 19.62$ N/cm^2 $= 19.62 \times 10^4$ N/m^2

$f = .008$

Applying Bernoulli's equations at points A and B and taking datum line passing through A, we have

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_B}{\rho g} + \frac{V_B^2}{2g} + z_B + h_f$$

But $V_A = V_B$ [$\because$ Dia. is same]

$z_A = 0,\ z_B = 30$

and $$h_f = \frac{4 \times f \times L \times V^2}{d \times 2g}$$

$$\therefore \quad \frac{p_A}{\rho g} + 0 = \frac{19.62 \times 10^4}{1000 \times 9.81} + 30 + \frac{4 \times .008 \times 400 \times .7074^2}{0.3 \times 2 \times 9.81}$$

$$= 20 + 30 + 1.088 = 51.088$$

$$\therefore \quad p_A = 51.088 \times 1000 \times 9.81 \text{ N/m}^2$$

$$= \frac{51.088 \times 1000 \times 9.81}{10^4}$$

$$= \mathbf{50.12 \text{ N/cm}^2. \text{ Ans.}}$$

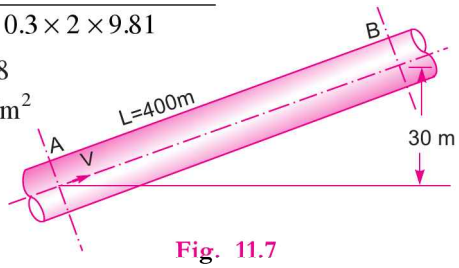


Fig. 11.7

▶ 11.5 HYDRAULIC GRADIENT AND TOTAL ENERGY LINE

The concept of hydraulic gradient line and total energy line is very useful in the study of flow of fluids through pipes. They are defined as :

11.5.1 Hydraulic Gradient Line. It is defined as the line which gives the sum of pressure head $\left(\frac{p}{w}\right)$ and datum head (z) of a flowing fluid in a pipe with respect to some reference line or it is the line which is obtained by joining the top of all vertical ordinates, showing the pressure head (p/w) of a flowing fluid in a pipe from the centre of the pipe. It is briefly written as H.G.L. (Hydraulic Gradient Line).

11.5.2 Total Energy Line. It is defined as the line which gives the sum of pressure head, datum head and kinetic head of a flowing fluid in a pipe with respect to some reference line. It is also defined as the line which is obtained by joining the tops of all vertical ordinates showing the sum of pressure head and kinetic head from the centre of the pipe. It is briefly written as T.E.L. (Total Energy Line).

Problem 11.22 *For the problem 11.16, draw the Hydraulic Gradient Line (H.G.L.) and Total Energy Line (T.E.L.).*

Solution. Given :

$$L = 50 \text{ m}, \; d = 200 \text{ mm} = 0.2 \text{ m}$$

$$H = 4 \text{ m}, \; f = .009$$

Velocity, V through pipe is calculated in problem 11.16 and its value is $V = 2.734$ m/s

Now, h_i = Head lost at entrance of pipe

$$= 0.5 \frac{V^2}{2g} + \frac{0.5 \times 2.734^2}{2 \times 9.81} = 0.19 \text{ m}$$

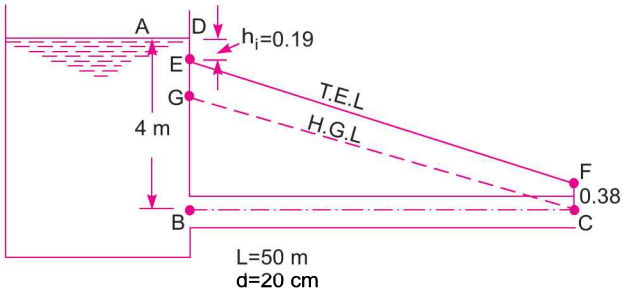


Fig. 11.8

and h_f = Head loss due to friction

$$= \frac{4 \times f \times L \times V^2}{d \times 2g} = \frac{4 \times 0.009 \times 50 \times (2.734)^2}{0.2 \times 2 \times 9.81} = 3.428 \text{ m.}$$

(*a*) **Total Energy Line (T.E.L.).** Consider three points, *A*, *B* and *C* on the free surface of water in the tank, at the inlet of the pipe and at the outlet of the pipe respectively as shown in Fig. 11.8. Let us find total energy at these points, taking the centre of pipe as reference line.

1. Total energy at $A = \frac{p}{\rho g} + \frac{V^2}{2g} + z = 0 + 0 + 4.0 = 4$ m
2. Total energy at B = Total energy at $A - h_i = 4.0 - 0.19 = 3.81$ m
3. Total energy at $C = \frac{p_c}{\rho g} + \frac{V_c^2}{2g} + z_c = 0 + \frac{V^2}{2g} + 0 = \frac{2.734^2}{2 \times 9.81} = 0.38$ m.

Hence total energy line will coincide with free surface of water in the tank. At the inlet of the pipe, it will decrease by h_i (= 0.19 m) from free surface and at outlet of pipe total energy is 0.38 m. Hence in Fig. 11.8,

(*i*) Point *D* represents total energy at *A*
(*ii*) Point *E*, where $DE = h_i$, represents total energy at inlet of the pipe
(*iii*) Point *F*, where $CF = 0.38$ represents total energy at outlet of pipe.
Join *D* to *E* and *E* to *F*. Then *DEF* represents the total energy line.

(*b*) **Hydraulic Gradient Line (H.G.L.).** H.G.L. gives the sum of $(p/w + z)$ with reference to the datum-line. Hence hydraulic gradient line is obtained by subtracting $\frac{V^2}{2g}$ from total energy line. At outlet of the pipe, total energy = $\frac{V^2}{2g}$. By subtracting $\frac{V^2}{2g}$ from total energy at this point, we shall get point *C*, which lies on the centre line of pipe. From *C*, draw a line *CG* parallel to *EF*. Then *CG* represents the hydraulic gradient line.

Problem 11.23 *For the problem 11.17, draw the hydraulic gradient and total energy line.*

Solution. Refer to problem 11.17.

Given: $L_1 = 25$ m, $d_1 = 0.15$ m
$L_2 = 15$ m, $d_2 = 0.3$ m, $f = .01$, $H = 8$ m

The velocity V_2 as calculated in problem 11.17 is

$V_2 = 1.113$ m/s
$V_1 = 4V_2 = 4 \times 1.113 = 4.452$ m/s

The various head losses are $h_i = 0.5 \times \frac{V_1^2}{2g} = \frac{0.5 \times 4.452^2}{2 \times 9.81} = 0.50$ m

$$h_{f_1} = \frac{4f \times L_1 \times V_1^2}{d_1 \times 2g} = \frac{4 \times .01 \times 25 \times (4.452)^2}{0.15 \times 2 \times 9.81} = 6.73 \text{ m}$$

$$h_e = \frac{(V_1 - V_2)^2}{2g} = \frac{(4.452 - 1.11)^2}{2 \times 9.81} = 0.568 \text{ m}$$

$$h_{f_2} = \frac{4 \times f \times L_2 \times V_2^2}{d_2 \times 2g} = \frac{4 \times .01 \times 15 \times (1.113)^2}{0.3 \times 2 \times 9.81} = 0.126 \text{ m}$$

$$h_o = \frac{V_2^2}{2g} = \frac{1.113^2}{2 \times 9.81} = 0.063 \text{ m}$$

Also $$V_1^2/2g = \frac{4.452^2}{2 \times 9.81} = 1.0 \text{ m}.$$

Total Energy Line

(*i*) Point *A* lies on free surface of water.

(*ii*) Take $AB = h_i = 0.5$ m.

(*iii*) From *B*, draw a horizontal line. Take *BL* equal to the length of pipe, *i.e.*, L_1. From *L* draw a vertical line downward.

(*iv*) Cut the line $LC = h_{f_1} = 6.73$ m.

(*v*) Join the point *B* to *C*. From *C*, take a line *CD* vertically downward equal to $h_e = 0.568$ m.

(*vi*) From *D*, draw *DM* horizontal and from point *F* which is lying on the centre of the pipe, draw a vertical line in the upward direction, meeting at *M*. From *M*, take a distance $ME = h_{f_2} = 0.126$. Join *DE*.

Then line *ABCDE* represents the total energy line.

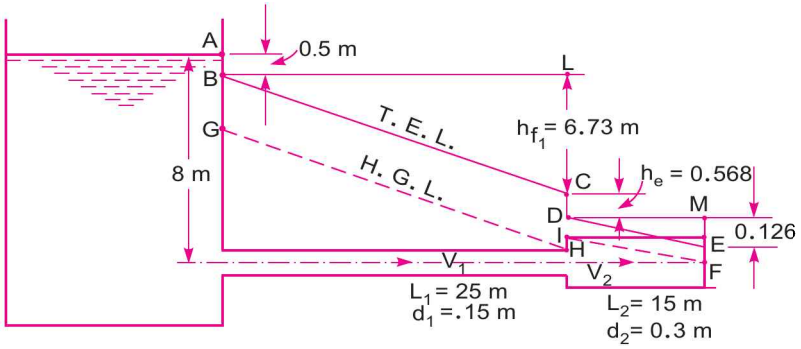


Fig. 11.9

Hydraulic Gradient Line (H.G.L.)

(*i*) From *B*, take $BG = \frac{V_1^2}{2g} = 1.0$ m.

(*ii*) Draw the line *GH* parallel to the line *BC*.

(*iii*) From *F*, draw a line *FI* parallel to the line *ED*.

(*iv*) Join the point *H* and *I*.

Then the line *GHIF* represents the hydraulic gradient line (H.G.L.).

Problem 11.24 *For Problem 11.18, draw the hydraulic gradient and total energy line.*

Solution. Refer to Problem 11.18,

Given : $d = 300$ mm $= 0.3$ m

$L = 400$ m, $Q = 300$ litres/s $= 0.3$ m^3/s

$f = .008$

Let $H_1 = 50$ m. But $H_1 - H_2 = 40.537$ m (Calculated in Problem 11.18)

$\therefore$ $H_2 = 50 - 40.537 = 9.463$ m.

The calculated losses are :

(*i*) $h_i = 0.459$ m (*ii*) $h_{f_1} = 39.16$ m

(*iii*) $h_o = 0.918$ m

(*a*) **T.E.L.**

(*i*) Point *A* is on the free surface of water in 1st tank. From *A*, take $AB = h_i = 0.459$ m.

(*ii*) Draw a horizontal line *BF*. Take *BF* equal to the length of pipe. From *F*, draw a vertical line in the downward direction. Cut $FC = h_{f_1} = 39.16$ m.

(*iii*) Join *BC*. From *C* take $CD = h_0 = 0.918$ m. The point *D* should coincide with free surface of water in 2nd tank. Then line *ABCD* is the total energy line.

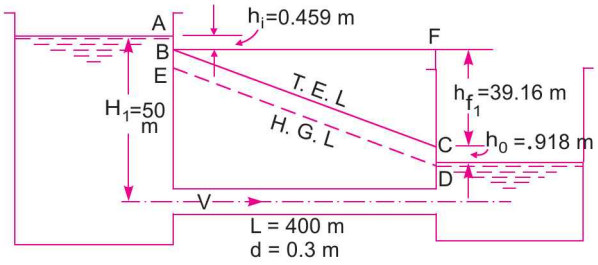


Fig. 11.10

(*b*) **H.G.L.** From *D*, draw a line *DE* parallel to line *BC*. Then *DE* is the H.G.L.

Or

From *B*, take $BE = \dfrac{V^2}{2g} = 0.918$ m and from *E* draw a line *ED* parallel to *BC*. The point *D* should coincide with free surface of water in the 2nd tank. Then line *ED* represents the H.G.L.

Problem 11.25 *The rate of flow of water pumped into a pipe ABC, which is 200 m long, is 20 litres/s. The pipe is laid on an upward slope of 1 in 40. The length of the portion AB is 100 m and its diameter is 100 mm, while the length of the portion BC is also 100 m but its diameter is 200 mm. The change of diameter at B is sudden. The flow is taking place from A to C, where the pressure at A is 19.62 N/cm^2 and end C is connected to a tank. Find the pressure at C and draw the hydraulic gradient and total energy line. Take f = .008.*

Solution. Given :

Length of pipe, $ABC = 200$ m

Discharge, $Q = 20$ litres/s $= 0.02$ m^3/s

Slope of pipe, $i = 1$ in $40 = \dfrac{1}{40}$

Length of pipe, $AB = 100$ m, Dia. of pipe $AB = 100$ mm $= 0.1$ m

Length of pipe, $BC = 100$ m, Dia. of pipe $BC = 200$ mm $= 0.2$ m

Pressure at *A*, $p_A = 19.62$ N/cm^2 $= 19.62 \times 10^4$ N/m^2

Co-efficient of friction, $f = .008$

Velocity of water in pipe *AB*, $V_1 = \dfrac{\text{Discharge}}{\text{Area of } AB} = \dfrac{0.02}{\frac{\pi}{4}(.1)^2} = 2.54$ m/s

Velocity of water in pipe BC, $V_2 = \dfrac{Q}{\text{Area of } BC} = \dfrac{0.02}{\frac{\pi}{4}(.2)^2} = 0.63$ m/s

Applying Bernoulli's equation to points A and C,

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_c}{\rho g} + \frac{V_c^2}{2g} + z_c + \text{total loss from } A \text{ to } C \qquad ...(i)$$

Total loss from A to C = Loss due to friction in pipe AB + loss of head due to enlargement at B + loss of head due to friction in pipe BC. ...(*ii*)

Now loss of head due to friction in pipe AB,

$$h_{f_1} = \frac{4fLV^2}{d \times 2g} = \frac{4 \times .008 \times 100 \times (2.54)^2}{0.1 \times 2 \times 9.81} = 10.52 \text{ m}$$

Loss of head due to friction in pipe BC,

$$h_{f_2} = \frac{4 \times .008 \times 100 \times (0.63)^2}{0.2 \times 2 \times 9.81} = 0.323 \text{ m}$$

Loss of head due to enlargement at B,

$$h_e = \frac{(V_1 - V_2)^2}{2g} = \frac{(2.54 - .63)^2}{2 \times 9.81} = 0.186 \text{ m}$$

$\therefore$ Total loss from A to C $= h_{f_1} + h_e + h_{f_2} = 10.52 + .186 + .323 = 11.029 \simeq 11.03$ m

Substituting this value in (*i*), we get

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_c}{\rho g} + \frac{V_c^2}{2g} + z_c + 11.03 \qquad ...(iii)$$

Taking datum line passing through A, we have

$$z_A = 0$$

$$z_c = \frac{1}{40} \times \text{total length of pipe} = \frac{1}{40} \times 200 = 5 \text{ m}$$

Also $p_A = 19.62 \times 10^4$ N/m^2

$V_A = V_1 = 2.54$ m/s, $V_c = V_2 = 0.63$ m/s

Substituting these values in (*iii*), we get

$$\frac{19.62 \times (10)^4}{1000 \times 9.81} + \frac{(2.54)^2}{2 \times 9.81} + 0 = \frac{p_c}{\rho g} + \frac{(0.63)^2}{2 \times 9.81} + 5.0 + 11.03$$

or $$20 + 0.328 = \frac{p_c}{\rho g} + 0.02 + 5.0 + 11.03$$

$\therefore$ $$20.328 = \frac{p_c}{\rho g} + 16.05$$

$\therefore$ $\dfrac{p_c}{\rho g} = 20.328 - 16.05 = 4.278$ m

or $p_c = 4.278 \times 1000 \times 9.81$ N/m^2

$= \dfrac{4.278 \times 1000 \times 9.81}{10^4}$ N/cm^2 = **4.196 N/cm^2. Ans.**

Hydraulic Gradient and Total Energy Line

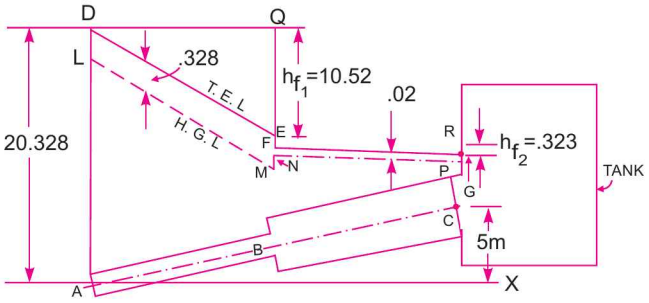


Fig. 11.11

Pipe AB. Assuming the datum line passing through A, then total energy at A

$$= \frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{19.62 \times 10^4}{1000 \times 9.81} + \frac{(2.54)^2}{2 \times 9.81} + 0 = 20 + 0.328$$

$= 20.328$ m

Total energy at B = Total energy at $A - h_{f_1} = 20.328 - 10.52 = 9.808$ m

Also $V_c^2/2g = \dfrac{(0.63)^2}{2 \times 9.81} = 0.02.$

Total Energy Line. Draw a horizontal line AX as shown in Fig. 11.11. The centre-line of the pipe is drawn in such a way that slope of pipe is 1 in 40. Thus the point C will be at a height of $\dfrac{1}{40} \times 200 = 5$ m from the line AX. Now draw a vertical line AD equal to total energy at A, *i.e.*, AD = 20.328 m. From point D, draw a horizontal line and from point B, a vertical line, meeting at Q. From Q, take vertical distance $QE = h_{f_1} = 10.52$ m. Join DE. From E, take $EF = h_e = 0.186$ m. From F, draw a horizontal line and from C, a vertical line meeting at R. From R take $RG = h_{f_2} = 0.323$ m. Join F to G. Then $DEFG$ represents the total energy line.

Hydraulic Gradient Line. Draw the line LM parallel to the line DE at a distance in the downward direction equal to 0.328 m. Also draw the line PN parallel to the line GF at a distance of $\dfrac{V_c^2}{2g} = 0.02$. Join point M to N. Then line $LMNP$ represents the hydraulic gradient line.

Problem 11.26 *A pipe line, 300 mm in diameter and 3200 m long is used to pump up 50 kg per second of an oil whose density is 950 kg/m^3 and whose kinematic viscosity is 2.1 stokes. The centre of the pipe line at the upper end is 40 m above than that at the lower end. The discharge at the upper end is atmospheric. Find the pressure at the lower end and draw the hydraulic gradient and the total energy line.*

Solution. Given :

Dia. of pipe, $d = 300 \text{ mm} = 0.3 \text{ m}$

Length of pipe, $L = 3200 \text{ m}$

Mass, $M = 50 \text{ kg/s} = \rho \, . \, Q$

$\therefore$ Discharge, $Q = \dfrac{50}{\rho} = \dfrac{50}{950} = 0.0526 \text{ m}^3/\text{s}$

$\because$ Density, $\rho = 950 \text{ kg/m}^3$

Kinematic viscosity, $\nu = 2.1 \text{ stokes} = 2.1 \text{ cm}^2/\text{s}$

$= 2.1 \times 10^{-4} \text{ m}^2/\text{s}$

Height of upper end $= 40 \text{ m}$

Pressure at upper end = atmospheric = 0

Reynolds number, $R_e = \dfrac{V \times d}{\nu}$, where $V = \dfrac{\text{Discharge}}{\text{Area}} = \dfrac{0.0526}{\frac{\pi}{4}(0.3)^2} = 0.744 \text{ m/s}$

$\therefore$ $R_e = \dfrac{0.744 \times 0.30}{2.1 \times 10^{-4}} = 1062.8$

$\therefore$ Co-efficient of friction, $f = \dfrac{16}{R_e} = \dfrac{16}{1062.8} = 0.015$

Head lost due to friction, $h_f = \dfrac{4 \times f \times L \times V^2}{d \times 2g}$

$= \dfrac{4 \times 0.015 \times 3200 \times (0.744)^2}{0.3 \times 2 \times 9.81} = 18.05 \text{ m of oil}$

Applying the Bernoulli's equation at the lower and upper end of the pipe and taking datum line passing through the lower end, we have

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + z_2 + h_f$$

But $z_1 = 0$, $z_2 = 40$ m, $V_1 = V_2$ as diameter is same

$p_2 = 0$, $h_f = 18.05$ m

$\therefore$ Substituting these values, we have

$$\frac{p_1}{\rho g} = 40 + 18.05 = 58.05 \text{ m of oil}$$

$\therefore$ $p_1 = 58.05 \times \rho g = 58.05 \times 950 \times 9.81$ [$\because$ ρ for oil = 950]

$= 540997 \text{ N/m}^2 = \dfrac{540997}{10^{-4}} \text{ N/cm}^2 =$ **54.099 N/cm^2. Ans.**

H.G.L. and T.E.L.

$$\frac{V^2}{2g} = \frac{(.744)^2}{2 \times 9.81} = 0.0282 \text{ m}$$

$$\frac{p_1}{\rho g} = 58.05 \text{ m of oil}$$

$$\frac{p_2}{\rho g} = 0$$

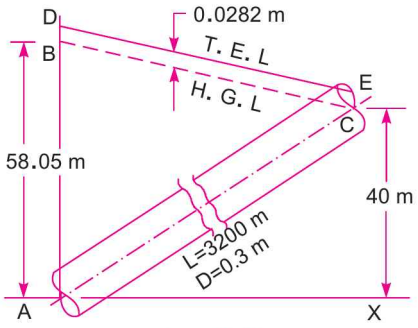


Fig. 11.12

Draw a horizontal line AX as shown in Fig. 11.12. From A, draw the centre line of the pipe in such a way that point C is a distance of 40 m above the horizontal line. Draw a vertical line AB through A such that AB = 58.05 m. Join B with C. Then BC is the hydraulic gradient line.

Draw a line DE parallel to BC at a height of 0.0282 m above the hydraulic gradient line. Then DE is the total energy line.

▶ 11.6 FLOW THROUGH SYPHON

Syphon is a long bent pipe which is used to transfer liquid from a reservoir at a higher elevation to another reservoir at a lower level when the two reservoirs are separated by a hill or high level ground as shown in Fig. 11.13.

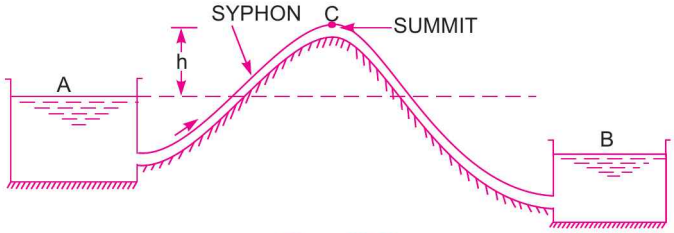


Fig. 11.13

The point C which is at the highest of the syphon is called the summit. As the point C is above the free surface of the water in the tank A, the pressure at C will be less than atmospheric pressure. Theoretically, the pressure at C may be reduced to – 10.3 m of water but in actual practice this pressure is only – 7.6 m of water or 10.3 – 7.6 = 2.7 m of water absolute. If the pressure at C becomes less than 2.7 m of water absolute, the dissolved air and other gases would come out from water and collect at the summit. The flow of water will be obstructed. Syphon is used in the following cases :

1. To carry water from one reservoir to another reservoir separated by a hill or ridge.
2. To take out the liquid from a tank which is not having any outlet.
3. To empty a channel not provided with any outlet sluice.

Problem 11.27 *A syphon of diameter 200 mm connects two reservoirs having a difference in elevation of 20 m. The length of the syphon is 500 m and the summit is 3.0 m above the water level in the upper reservoir. The length of the pipe from upper reservoir to the summit is 100 m. Determine the discharge through the syphon and also pressure at the summit. Neglect minor losses. The co-efficient of friction, f = .005.*

Solution. Given :

Dia. of syphon,	d = 200 mm = 0.20 m
Difference in level of two reservoirs,	H = 20 m
Length of syphon,	L = 500 m

Height of summit from upper reservoir, $h = 3.0$ m
Length of syphon upto summit, $L_1 = 100$ m
Co-efficient of friction, $f = .005$

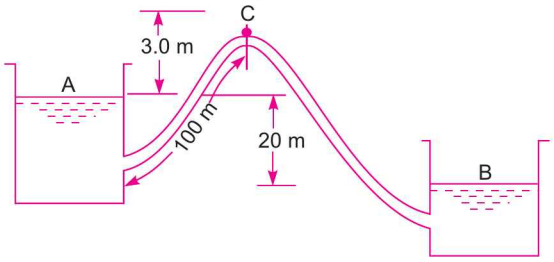


Fig. 11.14

If minor losses are neglected then the loss of head takes place only due to friction.
Applying Bernoulli's equation to points A and B,

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_B}{\rho g} + \frac{V_B^2}{2g} + z_B + \text{Loss of head due to friction from } A \text{ to } B$$

or $$0 + 0 + z_A = 0 + 0 + z_B + h_f \quad [\because p_A = p_B = \text{atmospheric pressure}, V_A = V_B = 0]$$

$$\therefore \quad z_A - z_B = h_f = \frac{4 \times f \times L \times V^2}{d \times 2g}$$

But $$z_A - z_B = 20 \text{ m}$$

$$\therefore \quad 20 = \frac{4 \times .005 \times 500 \times V^2}{0.20 \times 2 \times 9.81} = 2.548\ V^2$$

$$\therefore \quad V = \sqrt{\frac{20}{2.548}} = 2.80 \text{ m/s}$$

$\therefore$ Discharge, Q = Velocity × Area

$$= 2.80 \times \frac{\pi}{4} (.2)^2 = 0.0879 \text{ m}^3/\text{s} = \textbf{87.9 litres/s. Ans.}$$

Pressure at Summit. Applying Bernoulli's equation to points A and C,

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_c}{\rho g} + \frac{V_c^2}{2g} + z_c + \text{Loss of head due to friction between } A \text{ and } C$$

or $$0 + 0 + 0 = \frac{p_c}{\rho g} + \frac{V^2}{2g} + 3.0 + h_{f_1} \qquad [\text{Taking datum line passing through } A]$$

$$\therefore \quad 0 = \frac{p_c}{\rho g} + \frac{2.8^2}{2 \times 9.81} + 3.0 + \frac{4 \times .005 \times 100 \times (2.8)^2}{0.2 \times 2 \times 9.81} \quad [V_c = V = 2.80]$$

$$= \frac{p_c}{\rho g} + 0.399 + 3.0 + 4.00 = \frac{p_c}{\rho g} + 7.399$$

$$\therefore \quad \frac{p_c}{\rho g} = \textbf{– 7.399 m of water. Ans.}$$

Problem 11.28 *A syphon of diameter 200 mm connects two reservoirs having a difference in elevation of 15 m. The total length of the syphon is 600 m and the summit is 4 m above the water level in the upper reservoir. If the separation takes place at 2.8 m of water absolute, find the maximum length of syphon from upper reservoir to the summit. Take f = .004 and atmospheric pressure = 10.3 m of water.*

Solution. Given :

Dia. of syphon, $d = 200 \text{ mm} = 0.2 \text{ m}$

Difference of level in two reservoirs $= 15 \text{ m}$

Total length of pipe $= 600 \text{ m}$

Height of summit from upper reservoir $= 4 \text{ m}$

Pressure head at summit, $\frac{p_c}{\rho g} = 2.8 \text{ m of water absolute}$

Atmospheric pressure head, $\frac{p_c}{\rho g} = 10.3 \text{ m of water absolute}$

Co-efficient of friction, $f = .004$

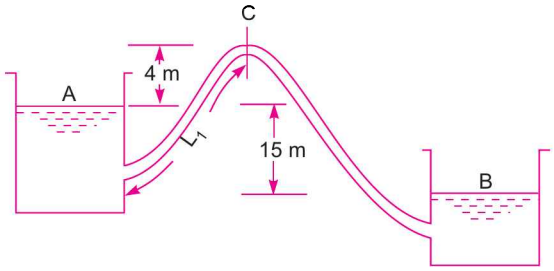


Fig. 11.15 (*a*)

Applying Bernoulli's equation to points A and C and taking the datum line passing through, A,

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_c}{\rho g} + \frac{V_c^2}{2g} + z_c + \text{Loss of head due to friction between } A \text{ and } C$$

Substituting the values of pressures in terms of absolute, we have

$$10.3 + 0 + 0 = 2.8 + \frac{V^2}{2g} + 4.0 + h_{f_1} \qquad [\because V_c = \text{velocity in pipe} = V]$$

$$\therefore \qquad h_{f_1} = 10.3 - 2.8 - 4.0 - \frac{V^2}{2g} = 3.5 - \frac{V^2}{2g} \qquad ...(i)$$

Applying Bernoulli's equation to points A and B and taking datum line passing through B,

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_B}{\rho g} + \frac{V_B^2}{2g} + z_B + \text{Loss of head due to friction from } A \text{ to } B$$

But $\frac{p_A}{\rho g} = \frac{p_B}{\rho g} = \text{atmospheric pressure}$

$V_A = 0, V_B = 0, z_A = 15, z_B = 0$

$\therefore \qquad 0 + 0 + 15 = 0 + 0 + 0 + h_f$

$$\therefore \qquad h_f = 15 \text{ or } \frac{4 \times f \times L \times V^2}{d \times 2g} = 15$$

or $$\frac{4 \times .004 \times 600 \times V^2}{0.2 \times 2 \times 9.81} = 15 \text{ or } V = \sqrt{\frac{15 \times 0.2 \times 2 \times 9.81}{4 \times .004 \times 600}} = 2.47 \text{ m/s}$$

Substituting this value of V in equation (i), we get

$$h_{f_1} = 3.5 - \frac{2.47^2}{2 \times 9.81} = 3.5 - 0.311 = 3.189 \text{ m} \qquad ...(ii)$$

But $$h_{f_2} = \frac{4 \times f \times L_1 \times V^2}{d \times 2g} \qquad ...(iii)$$

where L_1 = inlet leg of syphon or length of syphon from upper reservoir to the summit.

$$h_{f_1} = \frac{4 \times .004 \times L_1 \times (2.47)^2}{0.2 \times 2 \times 9.81} = 0.0248 \times L_1$$

Substituting this value in equation (ii),

$$0.0248\ L_1 = 3.189$$

$$\therefore \qquad L_1 = \frac{3.189}{.0248} = \textbf{128.58 m. Ans.}$$

Problem 11.29 *A syphon of diameter 200 mm connects two reservoirs whose water surface level differ by 40 m. The total length of the pipe is 8000 m. The pipe crosses a ridge. The summit of ridge is 8 m above the level of water in the upper reservoir. Determine the minimum depth of the pipe below the summit of the ridge, if the absolute pressure head at the summit of syphon is not to fall below 3.0 m of water. Take f = 0.006 and atmospheric pressure head = 10.3 m of water. The length of syphon from the upper reservoir to the summit is 500 m. Find the discharge also.*

Solution. Given :

Dia. of syphon, $d = 200 \text{ mm} = 0.20 \text{ m}$

Difference in levels of two reservoirs, $H = 40$ m

Total length of pipe, $L = 8000$ m

Height of ridge summit from water level in upper reservoir = 8 m

Let the depth of the pipe below the summit of ridge = x m

$\therefore$ Height of syphon from water surface in the upper reservoir = $(8 - x)$ m

Pressure head at C, $\frac{p_c}{\rho g} = 3.0$ m of water absolute

Atmospheric pressure head, $\frac{p_a}{\rho g} = 10.3$ m of water

Co-efficient of friction $f = .006$

Length of syphon from upper reservoir to the summit, $L_1 = 500$ m

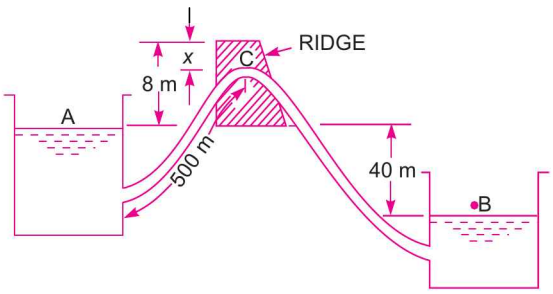


Fig. 11.15 (*b*)

Applying Bernoulli's equation to points *A* and B and taking datum line passing through *B*, we have

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_B}{\rho g} + \frac{V_B^2}{2g} + z_B + \text{head loss due to friction } A \text{ to } B$$

or $$0 + 0 + 40 = 0 + 0 + 0 + \frac{4 \times f \times L \times V^2}{d \times 2g}$$

$\therefore$ $$40 = \frac{4 \times 0.006 \times 8000 \times V^2}{0.2 \times 2 \times 9.81}$$

$\therefore$ $$V = \sqrt{\frac{40 \times 0.2 \times 2 \times 9.81}{4 \times .006 \times 8000}} = 0.904 \text{ m/s}$$

Now applying Bernoulli's equation to points *A* and *C* and assuming datum line passing through *A*, we have

$$\frac{p_A}{\rho g} + \frac{V_A^2}{2g} + z_A = \frac{p_c}{\rho g} + \frac{V_c^2}{2g} + z_c + \text{head loss due to friction from } A \text{ to } C$$

Substituting $\frac{p_A}{\rho g}$ and $\frac{p_c}{\rho g}$ in terms of absolute pressure

$$10.3 + 0 + 0 = 3.0 + \frac{V^2}{2g} + (8 - x) + \frac{4 \times f \times L_1 \times V^2}{d \times 2g}$$

or $$10.3 = 3.0 + \frac{(0.904)^2}{2 \times 9.81} + (8 - x) + \frac{4 \times .006 \times 500 \times (0.904)^2}{0.2 \times 2 \times 9.81}$$

$$= 3.0 + 0.041 + (8 - x) + 2.499 = 13.54 - x$$

$\therefore$ $x = 13.54 - 10.3 =$ **3.24 m. Ans.**

Discharge, $Q = \text{Area} \times \text{Velocity} = \frac{\pi}{4} \times (.2)^2 \times 0.904 =$ **0.0283 m^3/s. Ans.**

▶ 11.7 FLOW THROUGH PIPES IN SERIES OR FLOW THROUGH COMPOUND PIPES

Pipes in series or compound pipes are defined as the pipes of different lengths and different diameters connected end to end (in series) to form a pipe line as shown in Fig. 11.16.

Let, L_1, L_2, L_3 = length of pipes 1, 2 and 3 respectively

d_1, d_2, d_3 = diameter of pipes 1, 2, 3 respectively

V_1, V_2, V_3 = velocity of flow through pipes 1, 2, 3

f_1, f_2, f_3 = co-efficient of frictions for pipes 1, 2, 3

H = difference of water level in the two tanks.

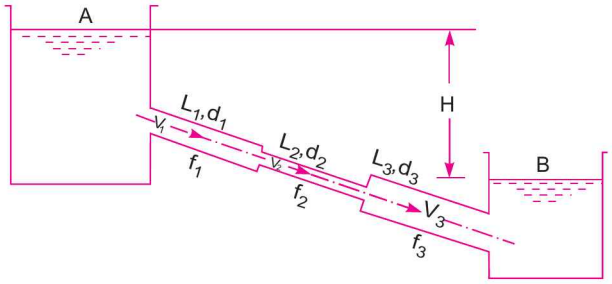


Fig. 11.16

The discharge passing through each pipe is same.

$$\therefore \quad Q = A_1V_1 = A_2V_2 = A_3V_3$$

The difference in liquid surface levels is equal to the sum of the total head loss in the pipes.

$$\therefore \quad H = \frac{0.5\,V_1^2}{2g} + \frac{4f_1L_1V_1^2}{d_1 \times 2g} + \frac{0.5\,V_2^2}{2g} + \frac{4f_2L_2V_2^2}{d_2 \times 2g}$$

$$+ \frac{(V_2 - V_3)^2}{2g} + \frac{4f_3L_3V_3^2}{d_3 \times 2g} + \frac{V_3^2}{2g} \quad ...(11.12)$$

If minor losses are neglected, then above equation becomes as

$$H = \frac{4f_1L_1V_1^2}{d_1 \times 2g} + \frac{4f_2L_2V_2^2}{d_2 \times 2g} + \frac{4f_3L_3V_3^2}{d_3 \times 2g} \quad ...(11.13)$$

If the co-efficient of friction is same for all pipes

i.e., $f_1 = f_2 = f_3 = f$, then equation (11.13) becomes as

$$H = \frac{4fL_1V_1^2}{d_1 \times 2g} + \frac{4fL_2V_2^2}{d_2 \times 2g} + \frac{4fL_3V_3^2}{d_3 \times 2g}$$

$$= \frac{4f}{2g}\left[\frac{L_1V_1^2}{d_1} + \frac{L_2V_2^2}{d_2} + \frac{L_3V_3^2}{d_3}\right] \quad ...(11.14)$$

Problem 11.30 *The difference in water surface levels in two tanks, which are connected by three pipes in series of lengths 300 m, 170 m and 210 m and of diameters 300 mm, 200 mm and 400 mm respectively, is 12 m. Determine the rate of flow of water if co-efficient of friction are .005, .0052 and .0048 respectively, considering : (i) minor losses also (ii) neglecting minor losses.*

Solution. Given :

Difference of water level, $H = 12$ m

Length of pipe 1, $L_1 = 300$ m and dia., $d_1 = 300$ mm= 0.3 m

Length of pipe 2, $L_2 = 170$ m and dia., $d_2 = 200$ mm = 0.2 m

Length of pipe 3, $L_3 = 210$ m and dia., $d_3 = 400$ mm = 0.4 m

Also, $f_1 = .005$, $f_2 = .0052$ and $f_3 = .0048$

(*i*) **Considering Minor Losses.** Let V_1, V_2 and V_3 are the velocities in the 1st, 2nd and 3rd pipe respectively.

From continuity, we have $A_1V_1 = A_2V_2 = A_3V_3$

$$\therefore \quad V_2 = \frac{A_1V_1}{A_2} = \frac{\frac{\pi}{4}d_1^2}{\frac{\pi}{4}d_2^2}V_1 = \frac{d_1^2}{d_2^2}V_1 = \left(\frac{0.3}{.2}\right)^2 \times V_1 = 2.25\ V_1$$

and $$V_3 = \frac{A_1V_1}{A_3} = \frac{d_1^2}{d_3^2}V_1 = \left(\frac{0.3}{0.4}\right)^2 V_1 = 0.5625\ V_1$$

Now using equation (11.12), we have

$$H = \frac{0.5\,V_1^2}{2g} + \frac{4f_1L_1V_1^2}{d_1 \times 2g} + \frac{0.5\,V_2^2}{2g} + \frac{4f_2L_2V_2^2}{d_2 \times 2g} + \frac{(V_2 - V_3)^2}{2g} + \frac{4f_3L_3V_3^2}{d_3 \times 2g} + \frac{V_3^2}{2g}$$

Substituting V_2 and V_3, $$12.0 = \frac{0.5\,V_1^2}{2g} + \frac{4 \times .005 \times 300 \times V_1^2}{0.3 \times 2g} + \frac{0.5 \times \left(2.25V_1^2\right)^2}{2g}$$

$$+ 4 \times 0.0052 \times 170 \times \frac{(2.25\,V_1)^2}{0.2 \times 2g} + \frac{(2.25\,V_1 - .562\,V_1)^2}{2g} + \frac{4 \times .0048 \times 210 \times (.5625\,V_1)^2}{0.4 \times 2g} + \frac{(.5625\,V_1)^2}{2g}$$

or $$12.0 = \frac{V_1^2}{2g}\,[0.5 + 20.0 + 2.53 + 89.505 + 2.847 + 3.189 + 0.316]$$

$$= \frac{V_1^2}{2g}\,[118.887]$$

$$\therefore \quad V_1 = \sqrt{\frac{12 \times 2 \times 9.81}{118.887}} = 1.407 \text{ m/s}$$

$\therefore$ Rate of flow, Q = Area × Velocity = $A_1 \times V_1$

$$= \frac{\pi}{4}(d_1)^2 \times V_1 = \frac{\pi}{4}(.3)^2 \times 1.407 = 0.09945 \text{ m}^3\text{/s}$$

$$= \mathbf{99.45 \text{ litres/s. Ans.}}$$

(*ii*) **Neglecting Minor Losses.** Using equation (11.13), we have

$$H = \frac{4f_1L_1V_1^2}{d_1 \times 2g} + \frac{4f_2L_2V_2^2}{d_2 \times 2g} + \frac{4f_3L_3V_3^2}{d_3 \times 2g}$$

or $$12.0 = \frac{V_1^2}{2g}\left[\frac{4 \times .005 \times 300}{0.3} + \frac{4 \times .0052 \times 170 \times (2.25)^2}{0.2} + \frac{4 \times .0048 \times 210 \times (.5625)^2}{0.4}\right]$$

$$= \frac{V_1^2}{2g} [20.0 + 89.505 + 3.189] = \frac{V_1^2}{2g} \times 112.694$$

$$\therefore \quad V_1 = \sqrt{\frac{2 \times 9.81 \times 12.0}{112.694}} = 1.445 \text{ m/s}$$

$$\therefore \quad \text{Discharge, } Q = V_1 \times A_1 = 1.445 \times \frac{\pi}{4} (.3)^2 = 0.1021 \text{ m}^3/\text{s} = \textbf{102.1 litres/s. Ans.}$$

Problem 11.30 (A). *Three pipes of 400 mm, 200 mm and 300 mm diameters have lengths of 400 m, 200 m, and 300 m respectively. They are connected in series to make a compound pipe. The ends of this compound pipe are connected with two tanks whose difference of water levels is 16 m. If co-efficient of friction for these pipes is same and equal to 0.005, determine the discharge through the compound pipe neglecting first the minor losses and then including them.*

Solution. Given :

Difference of water levels, $H = 16$ m

Length and dia. of pipe 1, $L_1 = 400$ m and $d_1 = 400$ mm $= 0.4$ m

Length and dia. of pipe 2, $L_2 = 200$ m and $d_2 = 200$ mm $= 0.2$ m

Length and dia. of pipe 3, $L_3 = 300$ m and $d_3 = 300$ mm $= 0.3$ m

Also $f_1 = f_2 = f_3 = 0.005$

(*i*) **Discharge through the compound pipe first neglecting minor losses.**

Let V_1, V_2 and V_3 are the velocities in the 1st, 2nd and 3rd pipe respectively.

From continuity, we have $A_1V_1 = A_2V_2 = A_3V_3$

$$\therefore \quad V_2 = \frac{A_1V_1}{A_2} = \frac{\frac{\pi}{4}d_1^2}{\frac{\pi}{4}d_2^2} \times V_1 = \frac{d_1^2}{d_2^2} V_1 = \left(\frac{0.4}{0.2}\right)^2 V_1 = 4V_1$$

and $$V_3 = \frac{A_1V_1}{A_3} = \frac{\frac{\pi}{4}d_1^2}{\frac{\pi}{4}d_3^2} \times V_1 = \frac{d_1^2}{d_3^2} V_1 = \left(\frac{0.4}{0.2}\right)^2 V_1 = 1.77V_1$$

Now using equation (11.13), we have

$$H = \frac{4f_1L_1V_1^2}{d_1 \times 2g} + \frac{4f_2L_2V_2^2}{d_2 \times 2g} + \frac{4f_3L_3V_3^2}{d_3 \times 2g}$$

or $$16 = \frac{4 \times 0.005 \times 400 \times V_1^2}{0.4 \times 2 \times 9.81} + \frac{4 \times 0.005 \times 200 \times (4V_1)^2}{0.2 \times 2 \times 9.81} + \frac{4 \times 0.005 \times 300}{0.3 \times 2 \times 9.81} \times (1.77\ V_1)^2$$

$$= \frac{V_1^2}{2 \times 9.81} \left(\frac{4 \times 0.005 \times 400}{0.4} + \frac{4 \times 0.005 \times 200 \times 16}{0.2} + \frac{4 \times 0.005 \times 300 \times 3.157}{0.3} \right)$$

$$16 = \frac{V_1^2}{2 \times 9.81} (20 + 320 + 63.14) = \frac{V_1^2}{2 \times 9.81} \times 403.14$$

$$\therefore \quad V_1 = \sqrt{\frac{16 \times 2 \times 9.81}{403.14}} = 0.882 \text{ m/s}$$

$\therefore$ Discharge, $Q = A_1 \times V_1 = \frac{\pi}{4}(0.4)^2 \times 0.882 =$ **0.1108 m³/s. Ans.**

(*ii*) **Discharge through the compound pipe considering minor losses also.**

Minor losses are :

(*a*) At inlet, $h_i = \frac{0.5\,V_1^2}{2g}$

(*b*) Between 1st pipe and 2nd pipe, due to contraction,

$$h_c = \frac{0.5\,V_2^2}{2g} = \frac{0.5\left(4V_1^2\right)}{2g} \qquad (\because V_2 = 4V_1)$$

$$= \frac{0.5 \times 16 \times V_1^2}{2g} = 8 \times \frac{V_1^2}{2g}$$

(*c*) Between 2nd pipe and 3rd pipe, due to sudden enlargement,

$$h_e = \frac{\left(V_2 - V_3\right)^2}{2g} = \frac{\left(4V_1 - 1.77V_1\right)^2}{2g} \qquad (\because V_3 = 1.77\ V_1)$$

$$= (2.23)^2 \times \frac{V_1^2}{2g} = 4.973\ \frac{V_1^2}{2g}$$

(*d*) At the outlet of 3rd pipe, $h_o = \frac{V_3^2}{2g} = \frac{\left(1.77V_1\right)^2}{2g} = 1.77^2 \times \frac{V_1^2}{2g} = 3.1329\ \frac{V_1^2}{2g}$

The major losses are $= \frac{4f_1 \times L_1 \times V_1^2}{d_1 \times 2g} + \frac{4f_2 \times L_2 \times V_2^2}{d_2 \times 2g} + \frac{4f_3 \times L_3 \times V_3^2}{d_3 \times 2g}$

$$= \frac{4 \times 0.005 \times 400 \times V_1^2}{0.4 \times 2 \times 9.81} + \frac{4 \times 0.005 \times 200 \times \left(4V_1\right)^2}{0.2 \times 2 \times 9.81} + \frac{4 \times 0.005 \times 300 \times \left(1.77V_1\right)^2}{0.3 \times 2 \times 9.81}$$

$$= 403.14 \times \frac{V_1^2}{2 \times 9.81}$$

$\therefore$ Sum of minor losses and major losses

$$= \left[\frac{0.5\,V_1^2}{2g} + 8 \times \frac{V_1^2}{2g} + 4.973\ \frac{V_1^2}{2g} + 3.1329\ \frac{V_1^2}{2g}\right] + 403.14\ \frac{V_1^2}{2g}$$

$$= 419.746\ \frac{V_1^2}{2g}$$

But total loss must be equal to H (or 16 m)

$\therefore$ $419.746 \times \frac{V_1^2}{2g} = 16 \quad \therefore \quad V_1 = \sqrt{\frac{16 \times 2 \times 9.81}{419.746}} = 0.864$ m/s

$\therefore$ Discharge, $Q = A_1 V_1 = \frac{\pi}{4}(0.4)^2 \times 0.864 =$ **0.1085 m³/s. Ans.**

▶ 11.8 EQUIVALENT PIPE

This is defined as the pipe of uniform diameter having loss of head and discharge equal to the loss of head and discharge of a compound pipe consisting of several pipes of different lengths and diameters. The uniform diameter of the equivalent pipe is called equivalent size of the pipe. The length of equivalent pipe is equal to sum of lengths of the compound pipe consisting of different pipes.

Let L_1 = length of pipe 1 and d_1 = diameter of pipe 1
L_2 = length of pipe 2 and d_2 = diameter of pipe 2
L_3 = length of pipe 3 and d_3 = diameter of pipe 3
H = total head loss
L = length of equivalent pipe
d = diameter of the equivalent pipe

Then $L = L_1 + L_2 + L_3$

Total head loss in the compound pipe, neglecting minor losses

$$H = \frac{4f_1L_1V_1^2}{d_1 \times 2g} + \frac{4f_2L_2V_2^2}{d_2 \times 2g} + \frac{4f_3L_3V_3^2}{d_3 \times 2g} \quad ...(11.14A)$$

Assuming $f_1 = f_2 = f_3 = f$

Discharge, $Q = A_1V_1 = A_2V_2 = A_3V_3 = \frac{\pi}{4} d_1^2V_1 = \frac{\pi}{4} d_2^2V_2 = \frac{\pi}{4} d_3^2V_3$

$\therefore$ $V_1 = \frac{4Q}{\pi d_1^2}$, $V_2 = \frac{4Q}{\pi d_2^2}$ and $V_3 = \frac{4Q}{\pi d_3^2}$

Substituting these values in equation (11.14*A*), we have

$$H = \frac{4fL_1 \times \left(\frac{4Q}{\pi d_1^2}\right)^2}{d_1 \times 2g} + \frac{4fL_2\left(\frac{4Q}{\pi d_2^2}\right)^2}{d_2 \times 2g} + \frac{4fL_3\left(\frac{4Q}{\pi d_3^2}\right)^2}{d_3 \times 2g}$$

$$= \frac{4 \times 16fQ^2}{\pi^2 \times 2g}\left[\frac{L_1}{d_1^5} + \frac{L_2}{d_2^5} + \frac{L_3}{d_3^5}\right] \quad ...(11.15)$$

Head loss in the equivalent pipe, $H = \frac{4f \,.\, L \,.\, V^2}{d \times 2g}$ [Taking same value of f as in compound pipe]

where $V = \frac{Q}{A} = \frac{Q}{\frac{\pi}{4}d^2} = \frac{4Q}{\pi d^2}$

$\therefore$ $$H = \frac{4f \,.\, L \,.\left(\frac{4Q}{\pi d^2}\right)^2}{d \times 2g} = \frac{4 \times 16Q^2f}{\pi^2 \times 2g}\left[\frac{L}{d^5}\right] \quad ...(11.16)$$

Head loss in compound pipe and in equivalent pipe is same hence equating equations (11.15) and (11.16), we have

$$\frac{4\times 16fQ^2}{\pi^2\times 2g}\left[\frac{L_1}{d_1^5}+\frac{L_2}{d_2^5}+\frac{L_3}{d_3^5}\right]=\frac{4\times 16Q^2f}{\pi^2\times 2g}\left[\frac{L}{d^5}\right]$$

or $$\frac{L_1}{d_1^5}+\frac{L_2}{d_2^5}+\frac{L_3}{d_3^5}=\frac{L}{d^5} \quad \text{or} \quad \frac{L}{d^5}=\frac{L_1}{d_1^5}+\frac{L_2}{d_2^5}+\frac{L_3}{d_3^5} \qquad ...(11.17)$$

Equation (11.17) is known as Dupuit's equation. In this equation $L = L_1 + L_2 + L_3$ and d_1, d_2 and d_3 are known. Hence the equivalent size of the pipe, *i.e.*, value of d can be obtained.

Problem 11.31 *Three pipes of lengths 800 m, 500 m and 400 m and of diameters 500 mm, 400 mm and 300 mm respectively are connected in series. These pipes are to be replaced by a single pipe of length 1700 m. Find the diameter of the single pipe.*

Solution. Given :

Length of pipe 1, $L_1 = 800$ m and dia., $d_1 = 500$ mm $= 0.5$ m

Length of pipe 2, $L_2 = 500$ m and dia., $d_2 = 400$ mm $= 0.4$ m

Length of pipe 3, $L_3 = 400$ m and dia., $d_3 = 300$ mm $= 0.3$ m

Length of single pipe, $L = 1700$ m

Let the diameter of equivalent single pipe $= d$

Applying equation (11.17), $\frac{L}{d^5}=\frac{L_1}{d_1^5}+\frac{L_2}{d_2^5}+\frac{L_3}{d_3^5}$

or $$\frac{1700}{d^5}=\frac{800}{.5^5}+\frac{500}{.4^5}+\frac{400}{0.3^5}=25600+48828.125+164609=239037$$

$$\therefore \quad d^5=\frac{1700}{239037}=.007118$$

$$\therefore \quad d=(.007188)^{0.2}=0.3718=\textbf{371.8 mm. Ans.}$$

▶ 11.9 FLOW THROUGH PARALLEL PIPES

Consider a main pipe which divides into two or more branches as shown in Fig. 11.17 and again join together downstream to form a single pipe, then the branch pipes are said to be connected in parallel. The discharge through the main is increased by connecting pipes in parallel.

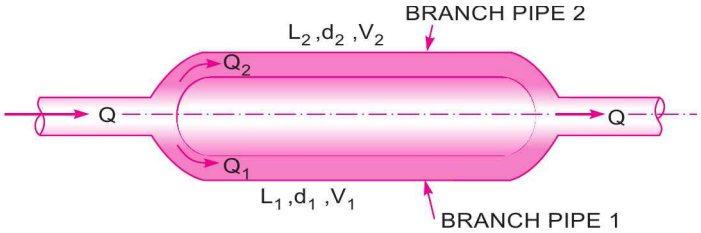


Fig. 11.17

The rate of flow in the main pipe is equal to the sum of rate of flow through branch pipes. Hence from Fig. 11.17, we have

$$Q = Q_1 + Q_2 \qquad ...(11.18)$$

In this, arrangement, the loss of head for each branch pipe is same.

$\therefore$ Loss of head for branch pipe 1 = Loss of head for branch pipe 2

or $$\frac{4f_1L_1V_1^2}{d_1 \times 2g} = \frac{4f_2L_2V_2^2}{d_2 \times 2g} \quad ...(11.19)$$

If $f_1 = f_2$, then $$\frac{L_1V_1^2}{d_1 \times 2g} = \frac{L_2V_2^2}{d_2 \times 2g} \quad ...(11.20)$$

Problem 11.32 *A main pipe divides into two parallel pipes which again forms one pipe as shown in Fig. 11.17. The length and diameter for the first parallel pipe are 2000 m and 1.0 m respectively, while the length and diameter of 2nd parallel pipe are 2000 m and 0.8 m. Find the rate of flow in each parallel pipe, if total flow in the main is 3.0 m^3/s. The co-efficient of friction for each parallel pipe is same and equal to .005.*

Solution. Given :

Length of pipe 1, $L_1 = 2000$ m
Dia. of pipe 1, $d_1 = 1.0$ m
Length of pipe 2, $L_2 = 2000$ m
Dia. of pipe 2, $d_2 = 0.8$ m
Total flow, $Q = 3.0$ m^3/s
$f_1 = f_2 = f = .005$
Let Q_1 = discharge in pipe 1
Q_2 = discharge in pipe 2
From equation (11.18), $Q = Q_1 + Q_2 = 3.0$ ...(*i*)

Using equation (11.19), we have

$$\frac{4f_1L_1V_1^2}{d_1 \times 2g} = \frac{4f_2L_2V_2^2}{d_2 \times 2g}$$

$$\frac{4 \times .005 \times 2000 \times V_1}{1.0 \times 2 \times 9.81} = \frac{4 \times .005 \times 2000 \times V_2^2}{0.8 \times 2 \times 9.81}$$

or $$\frac{V_1^2}{1.0} = \frac{V_2^2}{0.8} \text{ or } V_1^2 = \frac{V_2^2}{0.8}$$

$\therefore$ $$V_1 = \frac{V_2}{\sqrt{0.8}} = \frac{V_2}{.894} \quad ...(ii)$$

Now $$Q_1 = \frac{\pi}{4} d_1^2 \times V_1 = \frac{\pi}{4} (1)^2 \times \frac{V_2}{.894} \quad \left[\because V_1 = \frac{V_2}{.894}\right]$$

and $$Q_2 = \frac{\pi}{4} d_2^2 \times V_2 = \frac{\pi}{4} (.8)^2 \times V_2 = \frac{\pi}{4} \times .64 \times V_2$$

Substituting the value of Q_1 and Q_2 in equation (*i*), we get

$$\frac{\pi}{4} \times \frac{V_2}{0.894} + \frac{\pi}{4} \times .64\ V_2 = 3.0 \text{ or } 0.8785\ V_2 + 0.5026\ V_2 = 3.0$$

or $$V_2[.8785 + .5026] = 3.0 \text{ or } V = \frac{3.0}{1.3811} = 2.17 \text{ m/s.}$$

Substituting this value in equation (*ii*),

$$V_1 = \frac{V_2}{.894} = \frac{2.17}{0.894} = 2.427 \text{ m/s}$$

Hence $$Q_1 = \frac{\pi}{4} d_1^2 \times V_1 = \frac{\pi}{4} \times 1^2 \times 2.427 = \mathbf{1.906\ m^3/s.\ Ans.}$$

$\therefore$ $$Q_2 = Q - Q_1 = 3.0 - 1.906 = \mathbf{1.094\ m^3/s.\ Ans.}$$

Problem 11.33 *A pipe line of 0.6 m diameter is 1.5 km long. To increase the discharge, another line of the same diameter is introduced parallel to the first in the second half of the length. Neglecting minor losses, find the increase in discharge if 4f = 0.04. The head at inlet is 300 mm.*

Solution. Given :

Dia. of pipe line, $D = 0.6$ m

Length of pipe line, $L = 1.5 \text{ km} = 1.5 \times 1000 = 1500$ m

$4f = 0.04$ or $f = .01$

Head at inlet, $h = 300 \text{ mm} = 0.3$ m

Head at outlet, = atmospheric head = 0

$\therefore$ Head loss, $h_f = 0.3$ m

Length of another parallel pipe, $L_1 = \frac{1500}{2} = 750$ m

Dia. of another parallel pipe, $d_1 = 0.6$ m

Fig. 11.18 shows the arrangement of pipe system.

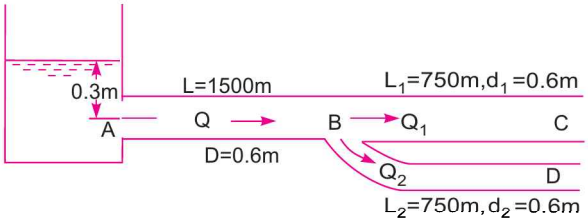


Fig. 11.18

1st Case. Discharge for a single pipe of length 1500 m and dia. = 0.6 m.

This head lost due to friction in single pipe is $h_f = \frac{4fLV^{*2}}{d \times 2g}$

where V^* = velocity of flow for single pipe

or $$0.3 = \frac{4 \times .01 \times 1500 \times V^{*2}}{0.6 \times 2g}$$

$\therefore$ $$V^* = \sqrt{\frac{0.3 \times 0.6 \times 2 \times 9.81}{4 \times .01 \times 1500}} = 0.2426 \text{ m/s}$$

$\therefore$ Discharge, $Q^* = V^* \times \text{Area} = 0.2426 \times \frac{\pi}{4} (.6)^2 = 0.0685\ \text{m}^3/\text{s}$ ...(*i*)

2nd Case. When an additional pipe of length 750 m and diameter 0.6 m is connected in parallel with the last half length of the pipe.

Let Q_1 = discharge in 1st parallel pipe

Q_2 = discharge in 2nd parallel pipe

$\therefore$ $Q = Q_1 + Q_2$

where Q = discharge in main pipe when pipes are parallel.

But as the length and diameters of each parallel pipe is same

$$\therefore \quad Q_1 = Q_2 = Q/2$$

Consider the flow through pipe ABC or ABD

Head loss through ABC = Head lost through AB + head lost through BC ...(*ii*)

But head lost due to friction through ABC = 0.3 m given

Head loss due to friction through $AB = \dfrac{4 \times f \times 750 \times V^2}{0.6 \times 2 \times 9.81}$, where V = velocity of flow through AB

$$= \frac{Q}{\text{Area}} = \frac{Q}{\frac{\pi}{4}(0.6)^2} = \frac{40}{\pi \times .36}$$

$\therefore$ Head loss due to friction through AB

$$= \frac{4 \times .01 \times 750}{0.6 \times 2 \times 9.81} \times \left(\frac{4Q}{\pi \times .36}\right)^2 = 31.87\ Q^2$$

Head loss due to friction through BC

$$= \frac{4 \times f \times L_1 \times V_1^2}{d_1 \times 2g}$$

$$= \frac{4 \times .01 \times 750}{0.6 \times 2 \times 9.81} \times \left[\frac{Q}{2 \times \frac{\pi}{4}(.6)^2}\right] \quad \left[\because V_1 = \frac{\text{Distance}}{\frac{\pi}{4}(.6)^2} = \frac{Q}{2 \times \frac{\pi}{4} \times (.6)^2}\right]$$

$$= \frac{4 \times .01 \times 750}{0.6 \times 2 \times 9.81} \times \frac{16}{4 \times \pi^2 \times .36^2}\ Q^2 = 7.969\ Q^2$$

Substituting these values in equation (*ii*), we get

$$0.3 = 31.87\ Q^2 + 7.969\ Q^2 = 39.839\ Q^2$$

$$\therefore \quad Q = \sqrt{\frac{0.3}{39.839}} = 0.0867 \text{ m}^3/\text{s}$$

$\therefore$ Increase in discharge = $Q - Q^*$ = 0.0867 – 0.0685 = **0.0182 m^3/s. Ans.**

Problem 11.34 *A pumping plant forces water through a 600 mm diameter main, the friction head being 27 m. In order to reduce the power consumption, it is proposed to lay another main of appropriate diameter along the side of the existing one, so that two pipes may work in parallel for the entire length and reduce the friction head to 9.6 m only. Find the diameter of the new main if, with the exception of diameter, it is similar to the existing one in every respect.*

Solution. Given :

Dia. of single main pipe, d = 600 mm = 0.6 m

Friction head, h_f = 27 m

Friction head for two parallel pipes = 9.6 m

1st Case.

For a single main [Fig. 11.19 (*a*)]

$$h_f = \frac{4 . f . L . V^2}{d \times 2g} \text{ or } 27.0 = \frac{4 \times f \times L \times V^2}{0.6 \times 2 \times 9.81}$$

$$\therefore \quad fLV^2 = \frac{27.0 \times 0.6 \times 2 \times 9.81}{4} = \frac{317.844}{5} = 79.461, \text{ where } V = \frac{Q}{A}$$

$$\therefore \quad f.L. \frac{Q^2}{A^2} = 79.461 \qquad ...(i)$$

2nd Case. Two pipes are in parallel [Fig. 11.19 (*b*)]
Loss of head in any one pipe = 9.6 m

$$\therefore \text{ For 1st pipe,} \quad h_{f_1} = \frac{4 . f . L . V_1^2}{d_1 \times 2g} = 9.6$$

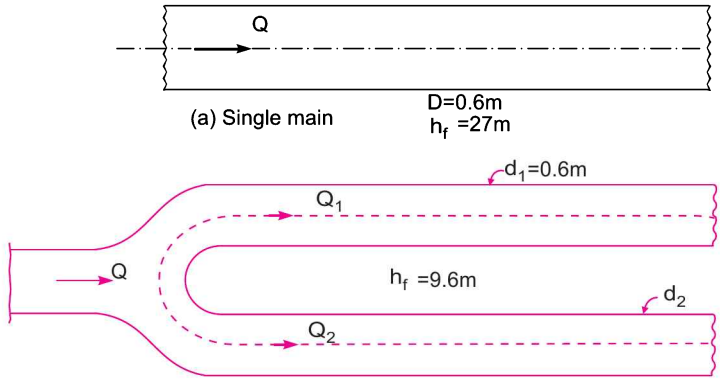


Fig. 11.19

But $$L_1 = L, V_1 = \frac{Q_1}{A_1} = \frac{Q_1}{A} \qquad \left[\because A_1 = A = \frac{\pi}{4}(.6)^2\right]$$

$$d_1 = d = 0.6$$

$$\therefore \quad \frac{4 . f . L}{0.6 \times 2 \times 9.81} \frac{Q_1^2}{A^2} = 0.6$$

or $$f \times L \times \frac{Q_1^2}{A^2} = \frac{9.6 \times 0.6 \times 2 \times 9.81}{4} = 28.2528 \qquad ...(ii)$$

For the 2nd pipe, $$h_{f_2} = \frac{4f \times L_2 \times V_2^2}{d_2 \times 2g} = 9.6, \quad \text{where } L_2 = L, V_2 = \frac{Q_2}{A_2}$$

$$\therefore \quad \frac{4f \times L \times Q_2^2}{d_2 \times 2g \times A_2^2} = 9.6$$

or $$\frac{f \times L \times Q_2^2}{d_2 \times A_2^2} = \frac{9.6 \times 2 \times 9.81}{4} = 47.088 \qquad ...(iii)$$

Dividing equation (*i*) by equation (*ii*), we get

$$\frac{Q^2}{Q_1^2} = \frac{79.461}{28.2528} = 2.8125$$

$$\therefore \quad \frac{Q}{Q_1} = \sqrt{2.8125} = 1.667$$

$$\therefore \quad Q_1 = \frac{Q}{1.667} = .596\ Q$$

But $Q_1 + Q_2 = Q$

$$\therefore \quad Q_2 = Q - Q_1 = Q - .596\ Q = 0.404\ Q$$

Dividing equation (*ii*) by equation (*iii*),

$$\frac{Q_1^2 \times d_2 \times A_2^2}{A^2 \times Q_2^2} = \frac{28.2528}{47.088} = 0.6$$

But $$A_2 = \frac{\pi}{4} d_2{}^2 \text{ and } A = \frac{\pi}{4} d^2 = \frac{\pi}{4} (.6)^2 = \frac{\pi}{4} \times .36$$

$$\therefore \quad \frac{Q_1^2}{Q_2^2} \times \frac{d_2 \times \left(\frac{\pi}{4}\right)^2 \times d_2^4}{\left(\frac{\pi}{4}\right)^2 \times (.36)^2} = 0.6 \quad \text{or} \quad \left(\frac{.596\ Q}{.404\ Q}\right)^2 \times \frac{d_2^5}{.36^2} = 0.6$$

or $$d_2{}^5 = 0.6 \times .36^2 \times \left(\frac{.404}{.596}\right)^2 = 0.03537$$

$$\therefore \quad d_2 = (.03537)^{1/5} = 0.5125 \text{ m} = \mathbf{512.5\ mm.\ Ans.}$$

Problem 11.35 *A pipe of diameter 20 cm and length 2000 m connects two reservoirs, having difference of water levels as 20 m. Determine the discharge through the pipe.*

If an additional pipe of diameter 20 cm and length 1200 m is attached to the last 1200 m length of the existing pipe, find the increase in the discharge. Take f = .015 and neglect minor losses.

Solution. Given :

Dia. of pipe, $d = 20 \text{ cm} = 0.20 \text{ m}$

Length of pipe, $L = 2000 \text{ m}$

Difference of water levels, $H = 20 \text{ m}$

Co-efficient of friction, $f = 0.015$

1st Case. When a single pipe connects the two reservoirs

$$H = \frac{4 . f . L . V^2}{d \times 2g} = \frac{4f . L}{d \times 2g} \left(\frac{Q}{\frac{\pi}{4} d^2}\right)^2 \qquad \left[\because V = \frac{Q}{\frac{\pi}{4} d^2}\right]$$

$$= \frac{32 f . L . Q^2}{\pi^2 \times g \times d^5}$$

or $$20 = \frac{32 \times .015 \times 2000 \times Q^2}{\pi^2 \times 9.81 \times (0.2)^5} = 30985.07\ Q^2$$

$$\therefore \quad Q = \sqrt{\frac{20}{30985.07}} = \mathbf{0.0254\ m^3/s.\ Ans.}$$

2nd Case.

Let Q_1 = discharge through pipe CD,
Q_2 = discharge through pipe DE,
Q_3 = discharge through pipe DF.

Length of pipe CD, L_1 = 800 m and its dia., d_1 = 0.20 m
Length of pipe DE, L_2 = 1200 m and its dia., d_2 = 0.20 m
Length of pipe DF, L_3 = 1200 m and its dia., d_3 = 0.20 m.

Since the diameters and lengths of the pipes DE and DF are equal. Hence Q_2 will be equal to Q_3. Also for parallel pipes, we have

$$Q_1 = Q_2 + Q_3 = Q_2 + Q_2 = 2Q_2 \qquad [\because\ Q_2 = Q_3]$$

$$\therefore \quad Q_2 = \frac{Q_1}{2}$$

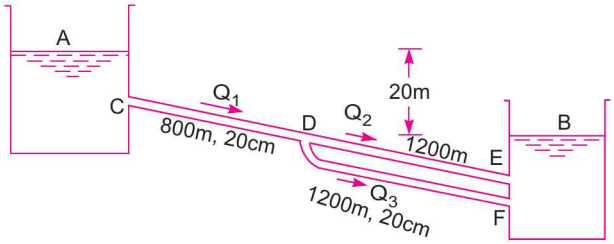


Fig. 11.20

Applying Bernoulli's equation to points A and B and taking the flow through CDE, we have

$$20 = \frac{4f.L_1.V_1^2}{d_1 \times 2g} + \frac{4f.L_2.V_2^2}{d_2 \times 2g}$$

where $V_1 = \dfrac{Q_1}{\frac{\pi}{4}(.2)^2} = \dfrac{4Q_1}{\pi \times .04}, V_2 = \dfrac{Q_2}{\frac{\pi}{4}(.2)^2} = \dfrac{4Q_2}{\pi \times .04} = \dfrac{4 \times \frac{Q_1}{2}}{\pi \times .04} = \dfrac{2Q_1}{\pi \times .04}$

$$= \frac{4 \times .015 \times 800}{0.2 \times 2 \times 9.81} \times \left(\frac{4Q_1}{\pi \times .04}\right)^2 + \frac{4 \times .015 \times 1200}{0.2 \times 2 \times 9.81} \times \left(\frac{2Q_1}{\pi \times .04}\right)^2$$

$$= 12394\ Q_1^2 + 4647\ Q_1^2 = 17041\ Q_1^2$$

$$\therefore \quad Q_1 = \sqrt{\frac{20}{17041}} = 0.0342\ \text{m}^3/\text{s}$$

Increase in discharge = $Q_1 - Q = 0.0342 - 0.0254 =$ **.0088 m³/s. Ans.**

Problem 11.36 *Two pipes have a length L each. One of them has a diameter D, and the other a diameter d. If the pipes are arranged in parallel, the loss of head, when a total quantity of water Q flows through them is h, but, if the pipes are arranged in series and the same quantity Q flows through them, the loss of head is H. If* $d = \frac{D}{2}$, *find the ratio of H to h, neglecting secondary losses and assuming the pipe co-efficient f has a constant value.*

Solution. Given :

Length of pipe 1, $L_1 = L$ and its dia. $d_1 = D$

Length of pipe 2, $L_2 = L$ and its dia., $d_2 = d$

Total discharge $= Q$

Head loss when pipes are arranged in parallel = h

Head loss when pipes are arranged in series = H

$$d = \frac{D}{2} \text{ and } f \text{ is constant}$$

1st Case. When pipes are connected to parallel

$$Q = Q_1 + Q_2 \qquad ...(i)$$

Loss of head in each pipe = h

For pipe AB, $\dfrac{4fL_1V_1^2}{d_1 \times 2g} = h$, where $V_1 = \dfrac{Q_1}{A_1} = \dfrac{Q_1}{\frac{\pi}{4}D^2} = \dfrac{4Q_1}{\pi D^2}$

$$d_1 = D$$

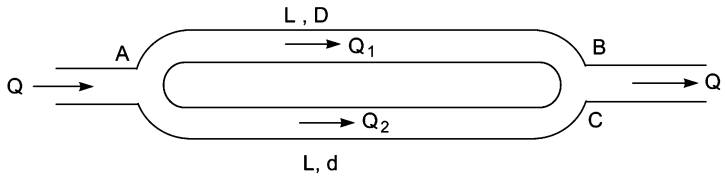


Fig. 11.21

$$\therefore \quad \frac{4fL \times \left(\frac{4Q_1}{\pi D^2}\right)^2}{D \times 2g} = h \text{ or } \frac{32fLQ_1^2}{\pi^2 D^5 \times g} = h \qquad ...(ii)$$

For pipe AC, $\dfrac{32fLQ_2^2}{\pi^2 d^5 \times g} = h$ ...(iii)

$$\therefore \quad \frac{32fLQ_1^2}{\pi^2 D^5 g} = \frac{32fLQ_2^2}{\pi^2 d^5 g} \text{ or } \frac{Q_1^2}{D^5} = \frac{Q_2^2}{d^5}$$

or $$\left(\frac{Q_1}{Q_2}\right)^2 = \frac{D^5}{d^5} = \frac{(2d)^5}{d^5} \qquad [\because D = 2d]$$

$$= 2^5 = 32$$

$$\therefore \quad \frac{Q_1}{Q_2} = \sqrt{32} = 5.657 \text{ or } Q_1 = 5.657\, Q_2$$

Substituting the values of Q_1 in equation (*i*), we get

$$Q = 5.657\ Q_2 + Q_2 = 6.657\ Q_2$$

$$\therefore \quad Q_2 = \frac{Q}{6.657} = 0.15\ Q \qquad ...(iv)$$

From (*i*) $\therefore$ $\quad Q_1 = Q - Q_2 = Q - 0.15\ Q = 0.85\ Q \qquad ...(v)$

2nd Case. When the pipes are connected in series.

Total loss = Sum of head losses in the two pipes

$$\therefore \quad H = \frac{4f\,.\,L\,.\,V_1^2}{d_1 \times 2g} + \frac{4f\,.\,L\,.\,V_2^2}{d_2 \times 2g}$$

where $V_1 = \dfrac{Q}{\frac{\pi}{4}D^2} = \dfrac{4Q}{\pi D^2}$, $V_2 = \dfrac{Q}{\frac{\pi}{4}d^2} = \dfrac{4Q}{\pi d^2}$

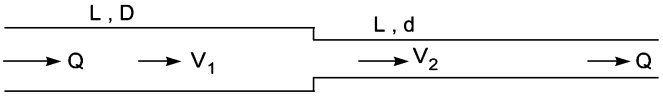


Fig. 11.22

$$\therefore \quad H = \frac{4f\,.\,L \times \left(\frac{4Q}{\pi D^2}\right)^2}{D \times 2g} + \frac{4fL\left(\frac{4Q}{\pi d^2}\right)^2}{d \times 2g}$$

or $$H = \frac{32\,fLQ^2}{D^5\pi^2 \times g} + \frac{32\,fLQ^2}{d^5\pi^2 \times g} \qquad ...(vi)$$

From equation (*ii*), $\dfrac{32\,fL}{\pi^2 D^5 \times g} = \dfrac{h}{Q_1^2}$

and from equation (*iii*), $\dfrac{32\,fL}{\pi^2 d^5 \times g} = \dfrac{h}{Q_2^2}$

Substituting these values in equation (*vi*), we have

$$H = Q^2 \times \frac{h}{Q_1^2} + Q^2 \times \frac{h}{Q_2^2} = \frac{Q^2}{Q_1^2}h + \frac{Q^2}{Q_2^2}h = h\left[\frac{Q^2}{Q_1^2} + \frac{Q^2}{Q_2^2}\right]$$

$$\therefore \quad \frac{H}{h} = \frac{Q^2}{Q_1^2} + \frac{Q^2}{Q_2^2}$$

But from equations (*iv*) and (*v*), $Q_1 = .85\ Q$ and $Q_2 = 0.15\ Q$

$$\therefore \quad \frac{H}{h} = \frac{Q^2}{.85^2 Q^2} + \frac{Q^2}{.15^2 Q^2} = \frac{1}{.85^2} + \frac{1}{.15^2} = 1.384 + 44.444 = \mathbf{45.828.\ Ans.}$$

Problem 11.36 (A). *Three pipes of the same length L, diameter D, and friction factor f are connected in parallel. Determine the diameter of the pipe of length L and friction factor f which will carry the same discharge for the same head loss. Use the formula* $h_f = f \times L \times V^2/2g\ D$.

Solution. Given :

Length of each pipe $= L$

Diameter of each pipe $= D$

Friction factor of each pipe $= f$

Head loss, $h_f = f \times L \times V^2/2gD$

When the three pipes are connected in parallel, then head loss in each pipe will be same. And total head loss will be equal to the head loss in each pipe.

Let h_f = Total head loss,

h_{f_1} = Head loss in 1st pipe,

h_{f_2} = Head loss in 2nd pipe, and h_{f_3} = Head loss in 3rd pipe.

Then $$h_f = h_{f_1} = h_{f_2} = h_{f_3} \text{ or } h_f = \frac{f \times L \times V^2}{2gD} \quad ...(i)$$

Let Q_1 = Discharge through 1st pipe, Q_2 = Discharge through 2nd pipe,

Q_3 = Discharge through 3rd pipe, and Q = Total discharge.

When the three pipes are connected in parallel, then

$$Q = Q_1 + Q_2 + Q_3 = 3 \times Q_1 \qquad (\because Q_1 = Q_2 = Q_3)$$

$$= 3 \times A_1 \times V_1$$

$$= 3 \times \frac{\pi}{4} D^2 \times V \left(\text{where } A_1 = \frac{\pi}{4} D^2 \text{ and } V_1 = V \right) \quad ...(ii)$$

For a single pipe (or length L ; friction factor f) which will carry same discharge as the three pipes in parallel

Let d = dia. of the single pipe

v = velocity through single pipe

Then discharge, Q = Area × Velocity $= \left(\frac{\pi}{4} d^2 \right) \times v$ ...(iii)

Equating the two values of discharge, given by equations (*ii*) and (*iii*), we get

$$3 \times \frac{\pi}{4} D^2 \times V = \frac{\pi}{4} d^2 \times v \text{ or } 3 \times \frac{D^2}{d^2} = \frac{v}{V} \quad ...(iv)$$

The head loss for the single pipe is also equal to the total head loss for three pipes when they are in parallel.

But head loss for the single pipe of length L, dia. d, friction factor f and velocity v is given by

$$h_f = \frac{f \times L \times v^2}{d \times 2g} \quad ...(v)$$

Equating the two values of h_f given by equations (*i*) and (*v*), we get

$$\frac{f \times L \times V^2}{D \times 2g} = \frac{f \times L \times v^2}{d \times 2g} \text{ or } \frac{V^2}{D} = \frac{v^2}{d}$$

or $$\frac{d}{D} = \frac{v^2}{V^2} \text{ or } \left(\frac{d}{D} \right)^{1/2} = \frac{v}{V}$$

Substituting the value of v/V in equation (*iv*), we get

$$3 \times \frac{D^2}{d^2} = \left(\frac{d}{D}\right)^{1/2} \text{ or } 3 = \left(\frac{d}{D}\right)^{1/2} \times \left(\frac{d}{D}\right)^2 = \left(\frac{d}{D}\right)^{5/2}$$

or $$\frac{d}{D} = 3^{2/5} = 3^{0.4} = 1.55$$

$\therefore$ $$d = \mathbf{1.55\ D.\ Ans.}$$

Hence dia. of single pipe should be 1.55 times the dia. of the three pipes connected in parallel.

Problem 11.37 *For a town water supply, a main pipe line of diameter 0.4 m is required. As pipes more than 0.35 m diameter are not readily available, two parallel pipes of the same diameter were used for water supply. If the total discharge in the parallel pipes is same as in the single main pipe, find the diameter of the parallel pipe. Assume the co-efficient of friction same for all pipes.*

Solution. Given :

Dia. of single main pipe line, $d = 0.4$ m

Let the length of single pipe line $= L$

Co-efficient of friction $= f$

Loss of head due to friction in single pipe $= \dfrac{4fLV^2}{d \times 2g} = \dfrac{4fLV^2}{0.4 \times 2 \times g}$ ...(*i*)

where V = Velocity of flow in the single pipe.

In case of parallel pipe, as the diameters and lengths of the two pipes are same. Hence discharge in each pipe will be half the discharge of single main pipe. As discharge in each parallel pipe is same, hence velocity will also be same.

Let V_* = Velocity in each parallel pipe

d_* = Dia. of each parallel pipe

Then loss of head due to friction in parallel pipes $= \dfrac{4f \times L \times V_*^2}{d_* \times 2g}$ ...(*ii*)

Equating the two losses given by equations (*i*) and (*ii*), we have

$$\frac{4f \,.\, L \,.\, V^2}{0.4 \times 2g} = \frac{4f \times L \times V_*^2}{d_* \times 2g}$$

Cancelling $\dfrac{4fL}{2g}$, $$\frac{V^2}{0.4} = \frac{V_*^2}{d_*^2} \text{ or } \frac{V^2}{V_*^2} = \frac{0.4}{d_*}$$ ...(*iii*)

From continuity

Total flow in single main = sum of flow in two parallel pipes

or Velocity of main × Area = 2 × Velocity in each parallel pipe × Area

$$V \times \frac{\pi}{4}(0.4)^2 = 2 \times V_* \times \frac{\pi}{4} d^2_* \text{ or } \frac{V}{V_*} = \frac{2 \times \frac{\pi}{4} d_*^2}{\frac{\pi}{4}(0.4)^2} = \frac{2d_*^2}{0.16}$$

Squaring both sides, $$\frac{V^2}{V_*^2} = \frac{4d_*^4}{0.0256}$$ ...(*iv*)

Comparing equations (*iii*) and (*iv*), we get

$$\frac{0.4}{d_*} = \frac{4d_*^4}{.0256} \quad \text{or} \quad d_*^5 = \frac{0.4 \times .0256}{4} = .00256$$

$\therefore$ $d_* = (.00256)^{1/5} = 0.303$ m = **30.3 cm. Ans.**

$\therefore$ Use two pipes of 30.3 cm diameter.

Problem 11.38 *An old water supply distribution pipe of 250 mm diameter of a city is to be replaced by two parallel pipes of smaller equal diameter having equal lengths and identical friction factor values. Find out the new diameter required.*

Solution. Given :

Dia. of old pipe, D = 250 mm = 0.25 m

Let d = Dia. of each of parallel pipes

Q = Discharge in old pipe

Q_1 = Discharge in first parallel pipe

Q_2 = Discharge in second parallel pipe

f = Friction factor.

When a single pipe is replaced by two parallel pipes, the head loss will be same in the single pipe and in each of the parallel pipes. Also the discharge in single pipe will be equal to the total discharge in two parallel pipes *i.e.*,

$$h_f = h_{f_1} = h_2 \qquad ...(i)$$

and

$$Q = Q_1 + Q_2 \qquad ...(ii)$$

As the dia. of each parallel pipe is same and also length of each parallel pipe is equal, hence

$$Q_1 = Q_2 \text{ or } Q_1 = Q_2 = Q/2$$

Now h_f = Head loss in single pipe

$$= \frac{f \times L \times V^2}{D \times 2g}, \qquad \text{where } f = \text{Friction factor}$$

$$= \frac{f \times L \times \left(\dfrac{Q}{\dfrac{\pi}{4} \times 0.25^2}\right)^2}{0.25 \times 2 \times 9.81} \qquad \left(\because V = \frac{Q}{\text{Area}} = \frac{Q}{\frac{\pi}{4}D^2}\right)$$

$$= \frac{f \times L \times (4Q)^2}{0.25 \times 2 \times 9.81 \times \left(\pi \times 0.25^2\right)^2} \qquad ...(iii)$$

h_{f_1} = Head loss in 1st parallel pipe

$$= \frac{f \times L \times (V_1)^2}{d \times 2g} \quad (\because \text{Dia. of parallel pipe} = d \text{ and } V_1 \text{ is the velocity in 1st parallel pipe})$$

$$= \frac{f \times L \times \left(\dfrac{Q}{2 \times \dfrac{\pi}{4} d^2}\right)^2}{d \times 2g} \qquad \left(\because V_1 = \frac{Q_1}{A_1} = \frac{\frac{Q}{2}}{\frac{\pi}{4}d^2} \text{ as } Q_1 = \frac{Q}{2}\right)$$

$$= \frac{f \times L \times (4Q)^2}{d \times 2 \times 9.81 \times \left(2 \times \pi \times d^2\right)^2} \qquad ...(iv)$$

But $h_f = h_{f_1}$

or $$\frac{f \times L \times (4Q)^2}{0.25 \times 2 \times 9.81 \times (\pi \times 0.25^2)^2} = \frac{f \times L \times (4Q)^2}{d \times 2 \times 9.81 \times \left(2 \times \pi \times d^2\right)^2}$$

or $$d \times (2\pi d^2)^2 = 0.25 \times (\pi \times 0.25^2)^2$$

or $$d \times 4 \times d^4 = 0.25 \times 0.25^4$$

or $$d^5 = \frac{0.25^5}{4} \text{ or } d = \frac{0.25}{(4)^{1/5}} = \frac{0.25}{1.3195} = \mathbf{0.1894\ m \simeq 0.19\ m.\ Ans.}$$

Problem 11.39 *A pipe of diameter 0.4 m and of length 2000 m is connected to a reservoir at one end. The other end of the pipe is connected to a junction from which two pipes of lengths 1000 m and diameter 300 mm run in parallel. These parallel pipes are connected to another reservoir, which is having level of water 10 m below the water level of the above reservoir. Determine the total discharge if f = 0.015. Neglect minor losses.*

Solution. Given :

Dia. of pipe, $d = 0.4$ m

Length of pipe, $L = 2000$ m

Dia. of parallel pipes, $d_1 = d_2 = 300$ mm $= 0.30$ m

Length of parallel pipes, $L_1 = L_2 = 1000$ m

Difference of water level in two reservoir, $H = 10$ m, $f = .015$

Applying Bernoulli's equation to points E and F. Taking flow through ABC.

$$10 = \frac{4fLV^2}{d \times 2g} + \frac{4f \times L_1 \times V_1^2}{d_1 \times 2g}$$

$$= \frac{4 \times .015 \times 2000 \times V^2}{0.4 \times 2 \times 9.81} + \frac{4 \times .015 \times 1000 \times V_1^2}{0.3 \times 2 \times 9.81}$$

$$= 15.29\ V^2 + 10.19\ V_1^2 \qquad ...(i)$$

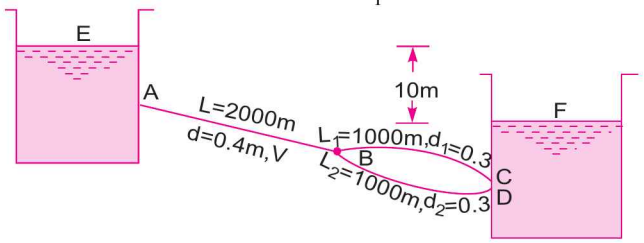


Fig. 11.23

From continuity equation

Discharge through AB = discharge through BC + discharge through BD

or $$\frac{\pi}{4} d^2 \times V = \frac{\pi}{4} d_1^2 \times V_1 + \frac{\pi}{4} d_1^2 V_2$$

But $d_1 = d_2$ and also the lengths of pipes BC and BD are equal and hence discharge through BC and BD will be same. This means $V_1 = V_2$ also

$\therefore$ $$\frac{\pi}{4} d^2 V = \frac{\pi}{4} d_1^2 \times V_1 + \frac{\pi}{4} d_1^2 \times V_1 \qquad [\because d_1 = d_2, V_1 = V_2]$$

$$= 2 \times \frac{\pi}{4} d_1^2 \times V_1 \text{ or } d^2 V = 2 d_1^2 V_1$$

or $$(0.4)^2 \times V = 2 \times (0.3)^2 V_1 \text{ or } .16V = 0.18\ V_1$$

$\therefore$ $$V_1 = \frac{0.16}{0.18} V = 0.888\ V$$

Substituting this value of V_1 in equation (i), we get

$$10 = 15.29\ V^2 + (10.19)(.888)^2 V^2 = 15.29\ V^2 + 8.035\ V^2 = 23.325\ V^2$$

$\therefore$ $$V = \sqrt{\frac{10}{23.325}} = 0.654 \text{ m/s}$$

$\therefore$ Discharge $= V \times$ Area

$$= 0.654 \times \frac{\pi}{4} d^2 = 0.654 \times \frac{\pi}{4} (0.4)^2 = \textbf{.0822 m}^3\textbf{/s. Ans.}$$

Problem 11.40 *Two sharp ended pipes of diameters 50 mm and 100 mm respectively, each of length 100 m are connected in parallel between two reservoirs which have a difference of level of 10 m. If the co-efficient of friction for each pipe is (4f) 0.32, calculate the rate of flow for each pipe and also the diameter of a single pipe 100 m long which would give the same discharge, if it were substituted for the original two pipes.*

Solution. Given :

Dia. of 1st pipe, $d_1 = 50$ mm $= 0.05$ m
Length of 1st pipe, $L_1 = 100$ m
Dia. of 2nd pipe, $d_2 = 100$ mm $= 0.10$ m
Length of 2nd pipe, $L_2 = 100$ m
Difference in level in reservoirs, $H = 10$ m
Co-efficient of friction $4f = 0.32$

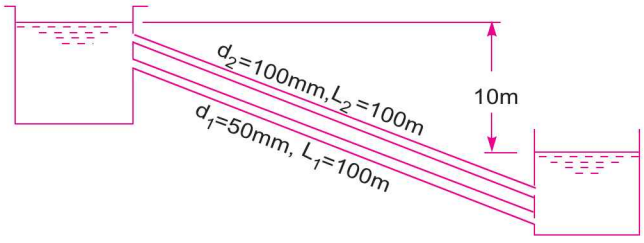


Fig. 11.24

Let V_1 = velocity of flow in pipe 1, and
V_2 = velocity of flow in pipe 2.

When the pipes are connected in parallel, the loss of head will be same in both the pipes.
For the first pipe, loss of head is given as

$$H = \frac{4f \times L_1 \times V_1^2}{d_1 \times 2g} = \frac{0.32 \times 100 \times V_1^2}{0.05 \times 2 \times 9.81} \qquad (\because 4f = .32)$$

or $$10 = 32.619\ V_1^2$$

$$\therefore \quad V_1 = \sqrt{\frac{10}{32.619}} = 0.5535 \text{ m/s}$$

$\therefore$ Rate of flow in 1st pipe, $Q_1 = V_1 \times A_1 = 0.5536 \times \frac{\pi}{4} (d_1)^2$

$$= .5536 \times \frac{\pi}{4} (0.05)^2 = .001087 \text{ m}^3/\text{s} = \textbf{1.087 litres/s. Ans.}$$

For the 2nd pipe, loss of head is given by,

$$10 = H = \frac{4f \times L_2 \times V_2^2}{d_2 \times 2g} = \frac{0.32 \times 100 \times V_2^2}{0.10 \times 2 \times 9.81}$$

$$\therefore \quad V_2 = \sqrt{\frac{10 \times .10 \times 2 \times 9.81}{.32 \times 100}} = 0.783 \text{ m/s}$$

$\therefore$ Rate of flow in 2nd pipe, $Q_2 = A_2 \times V_2 = \frac{\pi}{4} d_2^2 \times V_2$

$$= \frac{\pi}{4} (.1)^2 \times .783 = 0.00615 \text{ m}^3/\text{s} = \textbf{6.15 litres/s. Ans.}$$

Let D = diameter of a single pipe which is substituted for the two original pipes
L = length of single pipe = 100 m
V = velocity through pipe

The discharge through single pipe,

$$Q = Q_1 + Q_2 = 1.087 + 6.15 = 7.237 \text{ litres/s} = .007237 \text{ m}^3/\text{s}$$

$$\therefore \quad V = \frac{Q}{\text{Area}} = \frac{.007237}{\frac{\pi}{4} D^2} = \frac{4 \times .007237}{\pi D^2} = \frac{.009214}{D^2} \text{ m/s}$$

Loss of head through single pipe is

$$H = \frac{4f \times L \times V^2}{D \times 2g} = \frac{0.32 \times 100 \times \left(\frac{.009214}{D^2}\right)^2}{D \times 2 \times 9.81}$$

or $$10.0 = \frac{.32 \times 100 \times .009214^2}{2 \times 9.81 \times D^5} = \frac{.0001384}{D^5}$$

or $$D^5 = \frac{.0001384}{10} = .00001384$$

$$\therefore \quad D = (.00001384)^{1/5} = 0.1067 \text{ m} = \textbf{106.7 mm. Ans.}$$

Problem 11.41 *Two reservoir are connected by a pipe line of diameter 600 mm and length 4000 m. The difference of water level in the reservoirs is 20 m. At a distance of 1000 m from the upper reservoir, a small pipe is connected to the pipe line. The water can be taken from the small pipe. Find the discharge to the lower reservoir, if*

(i) No water is taken from the small pipe, and

(ii) 100 litres/s of water is taken from small pipe.

Take f = .005 and neglect minor losses.

Solution. Given :

Dia. of pipe, $d = 600 \text{ mm} = 0.60 \text{ m}$

Length of pipe, $L = 400 \text{ m}$

Difference of water level, $H = 20 \text{ m}, f = .005$

(*i*) No water is taken from small pipe

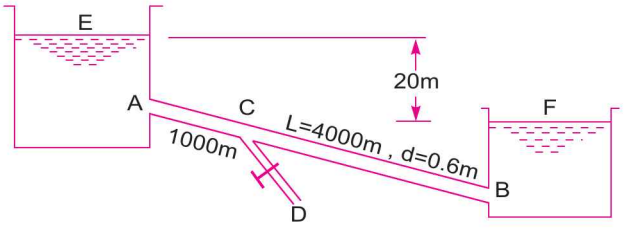


Fig. 11.25

The head loss due to friction in pipe $AB = \dfrac{4f \times L \times V^2}{d \times 2g}$ or $20 = \dfrac{4 \times .005 \times 4000 \times V^2}{0.6 \times 2 \times 9.81}$

$\therefore \quad V = \sqrt{\dfrac{20 \times 0.6 \times 2 \times 9.81}{4 \times .005 \times 4000}} = \sqrt{2.943} = 1.715 \text{ m/s}$

$\therefore$ Discharge, $Q = \text{Area} \times V = \dfrac{\pi}{4}(0.6)^2 \times 1.715 = \mathbf{0.485 \text{ m}^3\text{/s}}$. **Ans.**

(*ii*) 100 litres of water is taken from small pipe

Let Q_1 = discharge through pipe AC

Q_2 = discharge through pipe CB

Then for parallel pipes $Q_1 = Q_2 + 100 \text{ litres/s} = Q_2 + 0.1 \text{ m}^3\text{/s}$

$\therefore \quad Q_2 = (Q_1 - 0.1) \text{ m}^3\text{/s}$ ...(*i*)

Length of pipe AC, $L_1 = 1000 \text{ m}$

Length of pipe CB, $L_2 = 4000 - 1000 = 3000 \text{ m}$

Applying Bernoulli's equation to points E and F and taking flow through ABC, we have

$$20 = \frac{4fL_1V_1^2}{d_1 \times 2g} + \frac{4fL_2V_2^2}{d_2 \times 2g} \qquad \text{...(ii)}$$

where V_1 = velocity through pipe $AC = \dfrac{Q_1}{\frac{\pi}{4}(0.6)^2} = \dfrac{4Q_1}{\pi \times .36}$

d_1 = dia. of pipe $AC = 0.6$

V_2 = velocity through pipe $CB = \dfrac{Q_2}{\frac{\pi}{4}(0.6)^2} = \dfrac{4Q_2}{\pi \times .36}$

d_2 = dia. of pipe CB = 0.6

Substituting these values in equation (*ii*), we get

$$20 = \frac{4 \times .005 \times 1000}{0.6 \times 2 \times 9.81} \times \left(\frac{4Q_1}{\pi \times .36}\right)^2 + \frac{4 \times .005 \times 3000}{0.6 \times 2 \times 9.81} \times \left(\frac{4Q_2}{\pi \times .36}\right)^2$$

$$20 = 21.25\ Q_1^2 + 63.75\ Q_2^2 \qquad ...(iii)$$

But from (*i*), $Q_2 = Q_1 - 0.1$ or $Q_1 = Q_2 + 0.1$

Substituting the value of Q_1 in equation (*iii*), we get

$$20 = 21.25\ (Q_2 + 0.1)^2 + 63.75\ Q_2^2$$
$$= 21.55\ [Q_2^2 + .01 + 0.2\ Q_2] + 63.75\ Q_2^2$$
$$= 21.25\ Q_2^2 + 0.2125 + 4.250\ Q_2 + 63.75\ Q_2^2$$
$$= 85\ Q_2^2 + 4.25\ Q_2 + .2125$$

or $85\ Q_2^2 + 4.25\ Q_2 - 19.7875 = 0$

This is a quadratic equation in Q_2

$$\therefore \quad Q_2 = \frac{-4.25 \pm \sqrt{4.25^2 + 4 \times 85 \times 19.7875}}{2 \times 85}$$

$$= \frac{-4.25 \pm \sqrt{18.0625 + 6727.75}}{170} = \frac{-44.25 \pm 82.13}{170} = \frac{82.13 - 4.25}{170}$$

$= 0.458\ m^3/s$ (Neglecting negative root)

$\therefore$ Discharge to lower reservoir = Q_2 = **0.458 m^3/s. Ans.**

▶ 11.10 FLOW THROUGH BRANCHED PIPES

When three or more reservoirs are connected by means of pipes, having one or more junctions, the system is called a branching pipe system. Fig. 11.26 shows three reservoirs at different levels connected to a single junction, by means of pipes which are called branched pipes. The lengths, diameters and co-efficient of friction of each pipes is given. It is required to find the discharge and direction of flow in each pipe. The basic equations used for solving such problems are :

1. **Continuity equation** which means the inflow of fluid at the junction should be equal to the outflow of fluid.
2. **Bernoulli's equation, and**
3. **Darcy-Weisbach equation**

Also it is assumed that reservoirs are very large and the water surface levels in the reservoirs are constant so that steady conditions exist in the pipes. Also minor losses are assumed very small. The flow from reservoir A takes place to junction D. The flow from junction D is towards reservoirs C. Now the flow from junction D towards reservoir B will take place only when piezometric head at D $\left(\text{which is equal to } \dfrac{p_D}{\rho g} + Z_D\right)$ is more than the piezometric head at B (*i.e.*, Z_B). Let us consider that flow is from D to reservoir B.

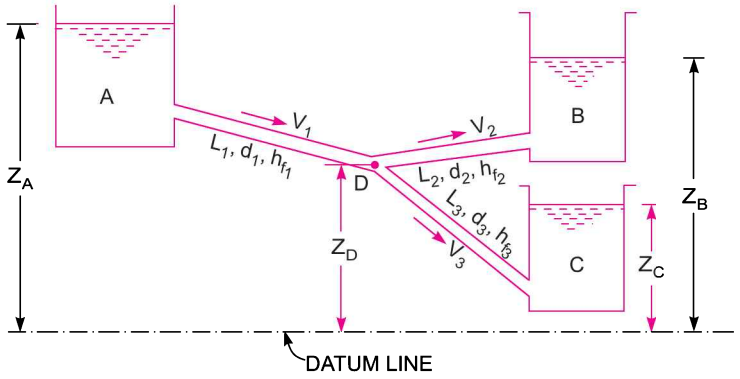


Fig. 11.26

For flow from A to D from Bernoulli's equation

$$Z_A = Z_D + \frac{p_D}{\rho g} + h_{f_1} \qquad ...(i)$$

For flow from D to B from Bernoulli's equation

$$Z_D + \frac{p_D}{\rho g} = Z_B + h_{f_2} \qquad ...(ii)$$

For flow from D to C from Bernoulli's equation

$$Z_D + \frac{p_D}{\rho g} = Z_C + h_{f_3} \qquad ...(iii)$$

From continuity equation,

Discharge through AD = Discharge through DB + Discharge through DC

$$\therefore \quad \frac{\pi}{4} d_1^2 V_1 = \frac{\pi}{4} d_2^2 \times V_2 + \frac{\pi}{4} d_3^2 V_3$$

or
$$d_1^2 V_1 = d_2^2 V_2 + d_3^2 V_3 \qquad ...(iv)$$

There are four unknowns *i.e.*, V_1, V_2, V_3 and $\frac{p_D}{\rho g}$ and there are four equations (i), (ii), (iii) and (iv).

Hence unknown can be calculated.

Problem 11.42 *Three reservoirs A, B and C are connected by a pipe system shown in Fig. 11.27. Find the discharge into or from the reservoirs B and C if the rate of flow from reservoirs A is 60 litres/s. Find the height of water level in the reservoir C. Take f = .006 for all pipes.*

Solution. Given :

Length of pipe AD, $L_1 = 1200$ m

Dia. of pipe AD, $d_1 = 30$ cm $= 0.30$ m

Discharge through AD, $Q_1 = 60$ litres/s $= 0.06$ m^3/s

Height of water level in A from reference line, $Z_A = 40$ m

For pipe DB, length $L_2 = 600$ m, dia., $d_2 = 20$ cm $= 0.20$ m, $Z_B = 38.0$

For pipe DC, length $L_3 = 800$ m, dia., $d_3 = 30$ cm $= 0.30$ m

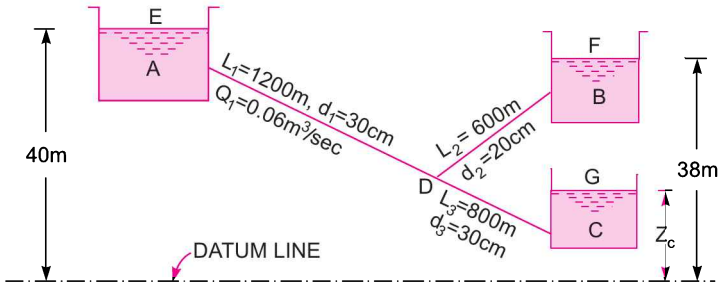


Fig. 11.27

Applying Bernoulli's equations to points E and D, $Z_A = Z_D + \frac{p_D}{\rho g} + h_{f_1}$

where $h_{f_1} = \frac{4 \,.\, f \,.\, L_1 \,.\, V_1^2}{d_1 \times 2g}$, where $V_1 = \frac{Q_1}{\text{Area}} = \frac{0.06}{\frac{\pi}{4}(.3)^2} = 0.848$ m/sec

$$h_{f_1} = \frac{4 \times .006 \times 1200 \times .848^2}{0.3 \times 2 \times 9.81} = 3.518 \text{ m}$$

$$\therefore \quad Z_A = Z_D + \frac{p_D}{\rho g} + 3.518 \text{ or } 40.0 = Z_D + \frac{p_D}{\rho g} + 3.518$$

$$\therefore \quad \left(Z_D + \frac{p_1}{\rho g}\right) = 40.0 - 3.518 = 36.482 \text{ m}$$

Hence piezometric head at D = 36.482. But Z_B = 38 m. Hence water flows from B to D.
Applying Bernoulli's equation to points B and D

$$Z_B = \left(Z_D + \frac{p_D}{\rho g}\right) + h_{f_2} \text{ or } 38 = 36.482 + h_{f_2}$$

$$\therefore \quad h_{f_2} = 38 - 36.482 = 1.518 \text{ m}$$

But $$h_{f_2} = \frac{4 \,.\, f \,.\, L_2 \,.\, V_2^2}{d_2 \times 2g} = \frac{4 \times .006 \times 600 \times V_2^2}{0.2 \times 2 \times 9.81}$$

$$\therefore \quad 1.518 = \frac{4 \times .006 \times 600 \times V_2^2}{0.2 \times 2 \times 9.81}$$

$$\therefore \quad V_2 = \sqrt{\frac{1.518 \times 0.2 \times 2 \times 9.81}{4 \times .006 \times 600}} = 0.643 \text{ m/s.}$$

$\therefore$ Discharge, $$Q_2 = V_2 \times \frac{\pi}{4}(d_2)^2 = 0.643 \times \frac{\pi}{2} \times (.2)^2$$

$$= 0.0202 \text{ m}^3/\text{s} = \textbf{20.2 litres/s. Ans.}$$

Applying Bernoulli's equation to points D and C

$$Z_D + \frac{p_D}{\rho g} = Z_C + h_{f_3}$$

or $$36.482 = Z_C + \frac{4f \,.\, L_3 \,.\, V_3^2}{d_3 \times 2g}, \text{ where } V_3 = \frac{Q_3}{\frac{\pi}{4} d_3^2}$$

But from continuity $Q_1 + Q_2 = Q_3$

$\therefore$ $$Q_3 = Q_1 + Q_2 = 0.06 + 0.0202 = 0.0802 \text{ m}^3/\text{s}$$

$\therefore$ $$V_3 = \frac{Q_3}{\frac{\pi}{4}(.3)^2} = \frac{0.0802}{\frac{\pi}{4}(.09)} = 1.134 \text{ m/s}$$

$\therefore$ $$36.482 = Z_C + \frac{4 \times .006 \times 800 \times 1.134^2}{0.3 \times 2 \times 9.81} = Z_C + 4.194$$

$\therefore$ $$Z_C = 36.482 - 4.194 = \mathbf{32.288 \text{ m. Ans.}}$$

Problem 11.43 *Three reservoirs, A, B and C are connected by a pipe system shown in Fig. 11.28. The lengths and diameters of pipes 1, 2 and 3 are 800 m, 1000 m, 800 m, and 300 mm, 200 mm and 150 mm respectively. Determine the piezometric head at junction D. Take f = .005.*

Solution. Given :

The length of pipe 1, $L_1 = 800$ m and its dia., $d_1 = 300$ mm $= 0.3$ m

The length of pipe 2, $L_2 = 1000$ m and its dia., $d_2 = 200$ mm $= 0.2$ m

The length of pipe 3, $L_3 = 800$ m and its dia., $d_3 = 150$ mm $= 0.15$ m

Height of reservoir, A from datum line, $Z_A = 60$ m

Similarly, $Z_B = 40$ m and $Z_C = 30$ m.

The direction of flow in pipes are shown (given) in Fig. 11.28. Applying Bernoulli's equation to points A and D

$$Z_A = \left(Z_D + \frac{p_D}{\rho g}\right) + h_{f_1}$$

or $$\left[Z_A - \left(Z_D + \frac{p_D}{\rho g}\right)\right] = h_{f_1} = \frac{4 \times f \times L_1 \times V_1^2}{d_1 \times 2g} = \frac{4 \times .005 \times 800 \times V_1^2}{0.3 \times 2 \times 9.81}$$

or $$60 - \left(Z_D + \frac{p_D}{\rho g}\right) = 2.718 \, V_1^2 \qquad ...(i)$$

Applying Bernoulli's equation to points D and B

$$\left(Z_D + \frac{p_D}{\rho g}\right) = Z_B + h_{f_2} = 40 + \frac{4f \times L_2 \times V_2^2}{d_2 \times 2g}$$

$$= 40 + \frac{4 \times .005 \times 1000 \times V_2^2}{0.2 \times 2 \times 9.81} = 40.0 + 5.09 \, V_2^2$$

or $$\left(Z_D + \frac{p_D}{\rho g}\right) - 40.0 = 5.09 \, V_2^2 \qquad ...(ii)$$

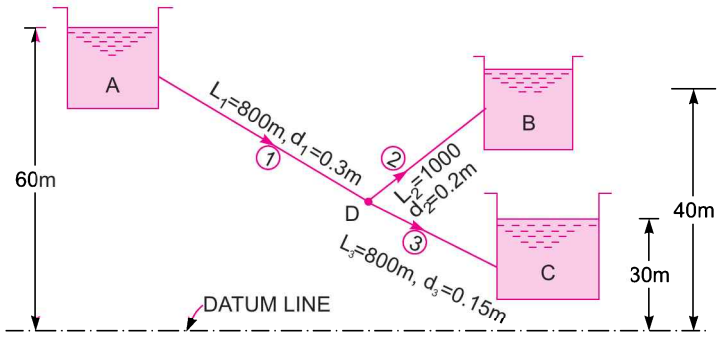


Fig. 11.28

Applying Bernoulli's equation to points D and C

$$\left(Z_D + \frac{p_D}{\rho g}\right) = Z_C + h_{f_3} = 30 + \frac{4f \times L_3 \times V_3^2}{d_3 \times 2g} = 30 + \frac{4 \times .005 \times 800 \times V_3^2}{0.15 \times 2 \times 9.81}$$

or $$\left(Z_D + \frac{p_D}{\rho g}\right) = 30.0 + 5.436\ V_3^2 \qquad ...(iii)$$

Adding (i) and (ii), we have $60 - 40 = 2.718\ V_1^2 + 5.09\ V_2^2$

or $$20 = 2.718\ V_1^2 + 5.09\ V_2^2 \qquad ...(iv)$$

Adding (i) and (iii), we have $60 = 2.718\ V_1^2 + 30.0 + 5.436\ V_3^2$

or $$60 - 30 = 30 = 2.718\ V_1^2 + 5.436\ V_3^2 \qquad ...(v)$$

Also from continuity equation, we have

$$Q_1 = Q_2 + Q_3$$

or $$\frac{\pi}{4} d_1^2 \times V_1 = \frac{\pi}{4} d_2^2 V_2 + \frac{\pi}{4} d_3^2 V_3 \text{ or } d_1^2 V_1 = d_2^2 V_2 + d_3^2 V_3$$

or $$0.3^2 V_1 = 0.2^2 V_2 + 0.15^2 \times V_3 \text{ or } .09 V_1 = .04\ V_2 + .0225\ V_3 \qquad ...(vi)$$

Now from (iv), $$V_2 = \sqrt{\frac{20 - 2.718\ V_1^2}{5.09}} \qquad ...(vii)$$

And from (v), $$V_3 = \sqrt{\frac{30 - 2.718\ V_1^2}{5.436}} \qquad ...(viii)$$

Substituting the value of V_2 and V_3 in (vi), we get

$$0.09\ V_1 = .04 \sqrt{\frac{20 - 2.718\ V_1^2}{5.09}} + .0225 \sqrt{\frac{30 - 2.718\ V_1^2}{5.436}}$$

Squaring both sides, we get

$$(0.09\ V_1)^2 = (.04)^2 \times \left(\frac{20 - 2.718\ V_1^2}{5.09}\right) + (0.0225)^2 \times \frac{30 - 2.718\ V_1^2}{5.436} + 2 \times .04$$

$$\times .0225 \times \sqrt{\frac{20 - 2.718\ V_1^2}{5.09}} \times \sqrt{\frac{30 - 2.718\ V_1^2}{5.436}}$$

or $.0081\ V_1^2 = .00628 - .000854\ V_1^2 + .00279 - .000253\ V_1^2 + .0018$

or $.0081\ V_1^2 + .000854\ V_1^2 + .000253\ V_1^2 = .00628 + .00279 + .0018 = .01087$

or $.009207\ V_1^2 = .01087$

$$\therefore \quad V_1 = \sqrt{\frac{.01087}{.009207}} = 1.086 \text{ m/s}$$

Substituting this value of V_1 in (*vii*) and (*viii*)

$$V_2 = \sqrt{\frac{20 - 2.718 \times V_1^2}{5.09}} = \sqrt{\frac{20 - 2.718 \times 1.086^2}{5.09}} = 1.816 \text{ m/s}$$

$$\therefore \quad V_3 = \sqrt{\frac{30 - 2.718 \times 1.086^2}{5.436}} = 2.22 \text{ m/s}$$

Piezometric head at $D = Z_D + \frac{p_D}{\rho g} = 30.0 + 5.436 \times V_3^2$

$= 30.0 + 5.436 \times (2.22)^2 =$ **56.79 m. Ans.**

Problem 11.44 *A pipe line 60 cm diameter bifurcates at a Y-junction into two branches 40 cm and 30 cm in diameter. If the rate of flow in the main pipe is 1.5 m^3/s and mean velocity of flow in 30 cm diameter pipe is 7.5 m/s, determine the rate of flow in the 40 cm diameter pipe.*

Solution. Given :

Dia. of main pipe, $D = 60$ cm $= 0.6$ m

Dia. of branch pipe 1, $D_1 = 40$ cm $= 0.4$ m

Dia. of branch pipe 2, $D_2 = 30$ cm $= 0.3$ m

Velocity in branch pipe 2, $V_2 = 7.5$ m/s

Rate of flow in main pipe, $Q = 1.5$ m^3/s

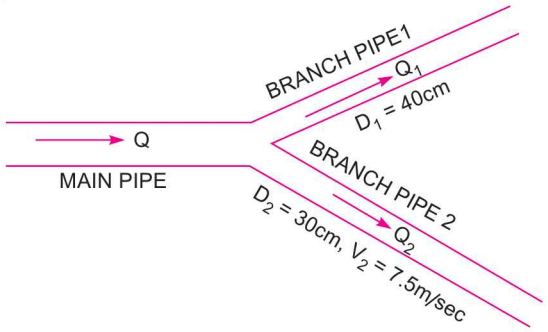


Fig. 11.29

Let Q_1 = Rate of flow in branch pipe 1,

Q_2 = Rate of flow in branch pipe 2,

Q = Rate of flow in main pipe,

Now rate of flow in main pipe is equal to the sum of rate of flow in branch pipes.

$$\therefore \quad Q = Q_1 + Q_2 \qquad ...(i)$$

But Q_2 = Area of branch pipe 2 × Velocity in branch pipe 2

$$= A_2 \times V_2 = \frac{\pi}{4} D_2^2 \times V_2 = \frac{\pi}{4} (0.3)^2 \times 7.5 = 0.53 \text{ m}^3/\text{s}$$

Substituting the values of Q and Q_2 in equation (i), we get

$$1.5 = Q_1 + 0.53$$

$$\therefore \quad Q_1 = 1.5 - 0.53 = \mathbf{0.97\ m^3/s.\ Ans.}$$

▶ 11.11 POWER TRANSMISSION THROUGH PIPES

Power is transmitted through pipes by flowing water or other liquids flowing through them. The power transmitted depends upon (i) the weight of liquid flowing through the pipe and (ii) the total head available at the end of the pipe. Consider a pipe AB connected to a tank as shown in Fig. 11.30. The power available at the end B of the pipe and the condition for maximum transmission of power will be obtained as mentioned below :

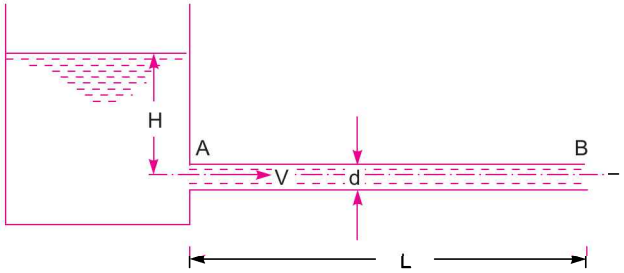


Fig. 11.30 *Power transmission through pipe.*

Let L = length of the pipe,
d = diameter of the pipe,
H = total head available at the inlet of pipe,
V = velocity of flow in pipe,
h_f = loss of head due to friction, and f = co-efficient of friction.

The head available at the outlet of the pipe, if minor losses are neglected
= Total head at inlet – loss of head due to friction

$$= H - h_f = H - \frac{4f \times L \times V^2}{d \times 2g} \qquad \left\{\because\ h_f = \frac{4f \times L \times V^2}{d \times 2g}\right\}$$

Weight of water flowing through pipe per sec,

$$W = \rho g \times \text{volume of water per sec} = \rho g \times \text{Area} \times \text{Velocity}$$

$$= \rho g \times \frac{\pi}{4} d^2 \times V$$

$\therefore$ The power transmitted at the outlet of the pipe
= weight of water per sec × head at outlet

$$= \left(\rho g \times \frac{\pi}{4} d^2 \times V\right) \times \left(H - \frac{4f \times L \times V^2}{d \times 2g}\right) \text{ Watts}$$

$\therefore$ Power transmitted at outlet of the pipe,

$$P = \frac{\rho g}{1000} \times \frac{\pi}{4} d^2 \times V \left(H - \frac{4fLV^2}{d \times 2g}\right) \text{ kW} \qquad \text{...(11.21)}$$

Efficiency of power transmission,

$$\eta = \frac{\text{Power available at outlet of the pipe}}{\text{Power supplied at the inlet of the pipe}}$$

$$= \frac{\text{Weight of water per sec} \times \text{Head available at outlet}}{\text{Weight of water per sec} \times \text{Head at inlet}}$$

$$= \frac{W \times (H - h_f)}{W \times H} = \frac{H - h_f}{H}. \qquad ...(11.22)$$

11.11.1 Condition for Maximum Transmission of Power. The condition for maximum transmission of power is obtained by differentiating equation (11.21) with respect to V and equating the same to zero.

Thus $$\frac{d}{dV}(P) = 0$$

or $$\frac{d}{dV}\left[\frac{\rho g}{1000} \times \frac{\pi}{4} d^2 \left(HV - \frac{4fLV^3}{d \times 2g}\right)\right] = 0$$

or $$\frac{\rho g}{1000} \times \frac{\pi}{4} d^2 \left(H - \frac{4 \times 3 \times f \times L \times V^2}{d \times 2g}\right) = 0$$

or $$H - 3 \times \frac{4fLV^2}{d \times 2g} = 0 \text{ or } H - 3 \times h_f = 0 \qquad \left(\because \frac{4fLV^2}{d \times 2g} = h_f\right)$$

$\therefore$ $$H = 3h_f \text{ or } h_f = \frac{H}{3} \qquad ...(11.23)$$

Equating (11.23) is the condition for maximum transmission of power. It states that power transmitted through a pipe is maximum when the loss of head due to friction is one-third of the total head at inlet.

11.11.2 Maximum Efficiency of Transmission of Power. Efficiency of power transmission through pipe is given by equation (11.22) as

$$\eta = \frac{H - h_f}{H}$$

For maximum power transmission through pipe the condition is given by equation (11.23) as

$$h_f = \frac{H}{3}$$

Substituting the value of h_f in efficiency, we get maximum η,

$$\eta_{max} = \frac{H - H/3}{H} = 1 - \frac{1}{3} = \frac{2}{3} \text{ or } 66.7\%. \qquad ...(11.24)$$

Problem 11.45 *A pipe of diameter 300 mm and length 3500 m is used for the transmission of power by water. The total head at the inlet of the pipe is 500 m. Find the maximum power available at the outlet of the pipe, if the value of f = .006.*

Solution. Given :

Diameter of the pipe, d = 300 mm = 0.30 m

Length of the pipe, $L = 3500$ m

Total head at inlet, $H = 500$ m

Co-efficient of friction, $f = .006$

For maximum power transmission, using equation (11.23)

$$h_f = \frac{H}{3} = \frac{500}{3} = 166.7 \text{ m}$$

Now $$h_f = \frac{4 \times f \times L \times V^2}{d \times 2g} = \frac{4 \times .006 \times 3500 \times V^2}{0.3 \times 2 \times 9.81} = 14.27\ V^2$$

Equating the two values of h_f, we get

$$166.7 = 14.27\ V^2 \text{ or } V = \sqrt{\frac{166.7}{14.27}} = 3.417 \text{ m/s}$$

$\therefore$ Discharge, $Q = V \times \text{Area}$

$$= 3.417 \times \frac{\pi}{4} (d)^2 = 3.417 \times \frac{\pi}{4} (.3)^2 = 0.2415 \text{ m}^3/\text{s}$$

Head available at the end of the pipe

$$= H - h_f = H - \frac{H}{3} = \frac{2H}{3} = \frac{2 \times 500}{3} = 333.33 \text{ m}$$

$\therefore$ Maximum power available $= \dfrac{\rho g \times Q \times \text{head at the end of pipe}}{1000}$ kW

$$= \frac{1000 \times 9.81 \times .2415 \times 333.33}{1000} \text{ kW} = \textbf{689.7 kW. Ans.}$$

Problem 11.46 *A pipe line of length 2000 m is used for power transmission. If 110.3625 kW power is to be transmitted through the pipe in which water having a pressure of 490.5 N/cm^2 at inlet is flowing. Find the diameter of the pipe and efficiency of transmission if the pressure drop over the length of pipe is 98.1 N/cm^2. Take f = .0065.*

Solution. Given :

Length of pipe, $L = 2000$ m

Power transmitted $= 110.3625$ kW

Pressure at inlet, $p = 490.5 \text{ N/cm}^2 = 490.5 \times 10^4 \text{ N/m}^2$

$\therefore$ Pressure head at inlet, $H = \dfrac{p}{\rho g} = \dfrac{490.5 \times 10^4}{1000 \times 9.81} = 500$ m $\quad [\because \rho = 1000]$

Pressure drop $= 98.1 \text{ N/cm}^2 = 98.1 \times 10^4 \text{ N/m}^2$

$\therefore$ Loss of head, $h_f = \dfrac{98.1 \times 10^4}{\rho g} = \dfrac{98.1 \times 10^4}{1000 \times 9.81} = 100$ m

Co-efficient of friction, $f = .0065$

Head available at the end of the pipe $= H - h_f = 500 - 100 = 400$ m

Let the diameter of the pipe $= d$

Now power transmitted is given by, $P = \frac{\rho g \times Q \times (H - h_f)}{1000}$ kW

or $$110.3625 = \frac{1000 \times 9.81 \times Q \times 400}{1000}$$

$\therefore$ $$Q = \frac{110.3625 \times 1000}{1000 \times 9.81 \times 400} = 0.02812$$

But discharge, Q = Area × Velocity = $\frac{\pi}{4} d^2 \times V$

$\therefore$ $$\frac{\pi}{4} d^2 \times V = .02812$$

$\therefore$ $$V = \frac{.2812 \times 4}{\pi d^2} = \frac{0.0358}{d^2} \quad ...(i)$$

The head lost due to friction, $h_f = \frac{4f \times L \times V^2}{d \times 2g}$

But $h_f = 100$ m

$\therefore$ $$100 = \frac{4 \times f \times L \times V^2}{d \times 2g} = \frac{4 \times .0065 \times 2000 \times V^2}{d \times 2 \times 9.81}$$

$$= \frac{2.65 \times V^2}{d} = \frac{2.65}{d} \times \left(\frac{.0358}{d^2}\right)^2 = \frac{.003396}{d^5}$$

$\because$ From equation (i), $V = \frac{.0358}{d^2}$

$\therefore$ $$100 = \frac{.003396}{d^5}$$

or $$d = \left(\frac{.003396}{100}\right)^{1/5} = 0.1277 \text{ m} = \mathbf{127.7 \text{ mm. Ans.}}$$

Efficiency of power transmission is given by equation (11.22),

$$\eta = \frac{H - h_f}{H} = \frac{500 - 100}{500} = \mathbf{0.80 = 80\%. \text{ Ans.}}$$

Problem 11.47 *For Problem 11.46, find : (i) the diameter of the pipe corresponding to maximum efficiency of transmission, (ii) diameter of the pipe corresponding to 90% efficiency of transmission.*

Solution. (i) Diameter of pipe corresponding to maximum efficiency.

Let the dia. of pipe for $\eta_{max} = d$

But from equation (11.24), $\eta_{max} = 66.67\% = \frac{2}{3}$

or $$\frac{H - h_f}{H} = \frac{2}{3} \text{ or } \frac{500 - h_f}{500} = \frac{2}{3}$$

or $$h_f = 500 - 500 \times \frac{2}{3} = \frac{1500 - 1000}{3} = \frac{500}{3} = 166.7 \text{ m}$$

The other data given from Problem 11.46,

Power transmitted = 110.3625

Length of pipe, $L = 2000$ m

Co-efficient of friction, $f = .0065$

Power transmitted is given by the relation,

$$P = \frac{\rho g \times Q \times \left(H - h_f\right)}{1000}$$

or $$110.3625 = \frac{1000 \times 9.81 \times Q \times (500 - 166.7)}{1000}$$

or $$Q = \frac{110.3625 \times 1000}{1000 \times 9.81 \times (500 - 166.7)} = 0.03375 \text{ m}^3/\text{s}$$

But Q = area of pipe × velocity of flow

$$= \frac{\pi}{4} d^2 \times V \{\text{where } V = \text{velocity of flow}\}$$

$\therefore$ $$0.03375 = \frac{\pi}{4} d^2 \times V$$

$\therefore$ $$V = \frac{0.03375 \times 4}{\pi \times d^2} = \frac{0.04297}{d^2} \quad ...(i)$$

Now the head lost due to friction, $h_f = \frac{4fLV^2}{d \times 2g}$

But $h_f = 166.7$ m

$\therefore$ $$166.7 = \frac{4 \times .0065 \times 2000 \times V^2}{d \times 2 \times 9.81}$$

$$= \frac{2.65\, V^2}{d} = \frac{2.65}{d} \times \left(\frac{.04297}{d^2}\right)^2 = \frac{.00489}{d^5} \qquad \left(\because V = \frac{.04297}{d^2}\right)$$

$\therefore$ $$d^5 = \frac{.00489}{166.7} = .00002933$$

$\therefore$ $$d = (.00002933)^{1/5} = 0.1240 \text{ m} = \mathbf{124\ mm.\ Ans.}$$

(*ii*) Let the diameter of pipe, when efficiency of transmission is 90% = d

$$\eta = 90\% = 0.9$$

But η is given by equation (11.22) as, $\eta = \frac{H - h_f}{H} = 0.9$

But $H = 500$ m

$$\therefore \quad \frac{500 - h_f}{500} = 0.9 \quad \text{or} \quad 500 - 500 \times 0.9 = h_f \quad \text{or} \quad 500 - 450 = h_f$$

$$\therefore \quad h_f = 500 - 450 = 50 \text{ m}$$

The other given data is, $P = 110.3625$, $L = 2000$, $f = .0065$

Using relation for power transmission, $P = \dfrac{\rho g \times Q \times (H - h_f)}{1000}$

or $$110.3625 = \frac{1000 \times 9.81 \times Q \times (500 - 50)}{1000}$$

$$Q = \frac{110.3625 \times 1000}{1000 \times 9.81 \times (500 - 50)} = .025 \text{ m}^3/\text{s}$$

But $$Q = \frac{\pi}{4} d^2 \times V$$

$$\therefore \quad \frac{\pi}{4} d^2 \times V = .025 \quad \text{or} \quad V = \frac{.025 \times 4}{\pi d^2} = \frac{0.03183}{d^2} \qquad ...(i)$$

Now the head lost due to friction, $h_f = \dfrac{4fLV^2}{d \times 2g}$

or $$50 = \frac{4 \times .0065 \times 2000 \times}{d \times 2g} \times \left(\frac{.03183}{d^2}\right)^2 = \frac{.002685}{d^5}$$

$$\therefore \quad d^5 = \frac{.002685}{50} = .0000537$$

$$d = (.0000537)^{1/5} = .1399 \text{ m} \simeq \textbf{140 mm. Ans.}$$

▶ 11.12 FLOW THROUGH NOZZLES

Fig. 11.31 shows a nozzle fitted at the end of a long pipe. The total energy at the end of the pipe consists of pressure energy and kinetic energy. By fitting the nozzle at the end of the pipe, the total energy is converted into kinetic energy. Thus nozzles are used, where higher velocities of flow are required. The examples are :

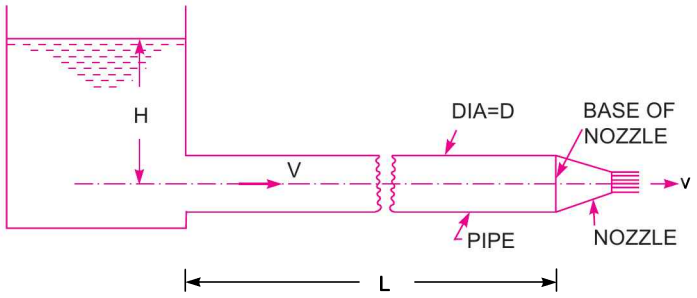


Fig. 11.31 *Nozzle fitted to a pipe.*

1. In case of Pelton turbine, the nozzle is fitted at the end of the pipe (called penstock) to increase velocity.
2. In case of the extinguishing fire, a nozzle is fitted at the end of the hose pipe to increase velocity.

Let D = diameter of the pipe, L = length of the pipe,

A = area of the pipe = $\frac{\pi}{4} D^2$,

V = velocity of flow in pipe,

H = total head at the inlet of the pipe,

d = diameter of nozzle at outlet,

v = velocity of flow at outlet of nozzle,

a = area of the nozzle at outlet = $\frac{\pi}{4} d^2$,

f = co-efficient of friction for pipe.

Loss of head due to friction in pipe, $h_f = \frac{4fLV^2}{2g \times D}$

$\therefore$ Head available at the end of the pipe or at the base of nozzle

= Head at inlet of pipe – head lost due to friction

$$= H - h_f = \left(H - \frac{4fLV^2}{2g \times D} \right)$$

Neglecting minor losses and also assuming losses in the nozzle negligible, we have

Total head at inlet of pipe = total head (energy) at the outlet of nozzle + losses

But total head at outlet of nozzle = kinetic head = $\frac{v^2}{2g}$

$$\therefore \quad H = \frac{v^2}{2g} + h_f = \frac{v^2}{2g} + \frac{4fLV^2}{2gD} \qquad \left(\because h_f = \frac{4fLV^2}{2gD} \right) \quad ...(i)$$

From continuity equation in the pipe and outlet of nozzle,

$$AV = av$$

$$\therefore \quad V = \frac{av}{A}$$

Substituting this value in equation (i), we get

$$H = \frac{v^2}{2g} + \frac{4fL}{2gD} \times \left(\frac{av}{A} \right)^2 = \frac{v^2}{2g} + \frac{4fLa^2v^2}{2g \times D \times A^2} = \frac{v^2}{2g} \left(1 + \frac{4fLa^2}{DA^2} \right)$$

$$\therefore \quad v = \sqrt{\frac{2gH}{\left(1 + \frac{4fL}{D} \times \frac{a^2}{A^2} \right)}} \qquad ...(11.25)$$

$\therefore$ Discharge through nozzle = $a \times v$.

11.12.1 Power Transmitted Through Nozzle. The kinetic energy of the jet at the outlet of nozzle = $\frac{1}{2} mv^2$

Now mass of liquid at the outlet of nozzle per second = ρav

$\therefore$ Kinetic energy of the jet at the outlet per sec. = $\frac{1}{2} \rho av \times v^2 = \frac{1}{2} \rho av^3$

$\therefore$ Power in kW at the outlet of nozzle = (K.E./sec) $\times \frac{1}{1000} = \frac{\frac{1}{2}\rho av^3}{1000}$

$\therefore$ Efficiency of power transmission through nozzle,

$$\eta = \frac{\text{Power at outlet of nozzele}}{\text{Power at the inlet of pipe}} = \frac{\frac{\frac{1}{2}\rho av^3}{1000}}{\frac{\rho g . Q . H}{1000}}$$

$$= \frac{\frac{1}{2}\rho av . v^2}{\rho g . Q . H} = \frac{\frac{1}{2}\rho av . v^2}{\rho g . av . H} \qquad \{\because Q = av\}$$

$$= \frac{v^2}{2gH} = \left[\frac{1}{1 + \frac{4fL}{D} \times \frac{a^2}{A^2}}\right] \qquad ...(11.26)$$

$$\left(\because \text{From equation (11.25)}, \frac{v^2}{2gH} = \left[\frac{1}{1 + \frac{4fL}{D}\frac{a^2}{A^2}}\right]\right)$$

11.12.2 Condition for Maximum Power Transmitted Through Nozzle. We know that, the total head at inlet of pipe = total head at the outlet of the nozzle + losses

i.e.,
$$H = \frac{v^2}{2g} + h_f \qquad \left[\because \text{total head at outlet of nozzle} = \frac{v^2}{2g} \text{ and } h_f = \frac{4 . f . L . V^2}{D \times 2g} = \text{loss of liquid in pipe}\right]$$

$$= \frac{v^2}{2g} + \frac{4 . f . L . V^2}{D \times 2g}$$

$\therefore$
$$\frac{v^2}{2g} = \left(H - \frac{4 . f . L . V^2}{D \times 2g}\right)$$

But power transmitted through nozzle = $\frac{\frac{1}{2}\rho av^3}{1000} = \frac{\frac{1}{2}\rho av}{1000} \times v^2 = \frac{\frac{1}{2}\rho av}{1000}\left[2g\left(H - \frac{4 . f . L . V^2}{D \times 2g}\right)\right]$

$$= \frac{\rho g a v}{1000}\left[H - \frac{4fLV^2}{D \times 2g}\right] \qquad ...(11.27)$$

Now from continuity equation, $AV = av$

$$\therefore \qquad V = \frac{av}{A}$$

Substituting the value of V in equation (11.27), we get

$$\text{Power transmitted through nozzle} = \frac{\rho g a v}{1000}\left[H - \frac{4fLa^2}{D \times 2g}\frac{v^2}{A^2}\right]$$

The power (P) will be maximum, when $\frac{d(P)}{dv} = 0$

or $$\frac{d}{dv}\left[\frac{\rho g a v}{1000}\left(H - \frac{4fL}{D \times 2g}\frac{a^2v^2}{A^2}\right)\right] = 0$$

or $$\frac{d}{dv}\left[\frac{\rho g a}{1000}\left(Hv - \frac{4fL}{D \times 2g}\frac{a^2v^3}{A^2}\right)\right] = 0$$

or $$\left[\frac{\rho g a}{1000}\left(H - 3\frac{4fL}{D \times 2g}\frac{a^2v^2}{A^2}\right)\right] = 0 \text{ or } H - 3 \times \frac{4fL}{D \times 2g} \times V^2 = 0 \left(\because V = \frac{av}{A}\right)$$

or $$H - 3h_f = 0 \qquad \left(\because \frac{4fLV^2}{D \times 2g} = h_f = \text{head loss in pipe}\right)$$

or $$h_f = \frac{H}{3} \qquad ...(11.28)$$

Equation (11.28) gives the condition for maximum power transmitted through nozzle. It states that power transmitted through nozzle is maximum when the head lost due to friction in pipe is one-third the total head supplied at the inlet of pipe.

11.12.3 Diameter of Nozzle for Maximum Transmission of Power Through Nozzle. For maximum transmission of power, the condition is given by equation (11.28) as, $h_f = \frac{H}{3}$

But $$h_f = \frac{4f \,.\, L \,.\, V^2}{D \times 2g}$$

$$\therefore \qquad \frac{4fLV^2}{D \times 2g} = \frac{H}{3} \text{ or } H = 3 \times \frac{4fLV^2}{D \times 2g}$$

But H is also = total head at outlet of nozzle + losses

$$= \frac{v^2}{2g} + h_f = \frac{v^2}{2g} + \frac{4fLV^2}{D \times 2g}$$

Equating the two values of H, we get

$$3 \times \frac{4fLV^2}{D \times 2g} = \frac{v^2}{2g} + \frac{4fLV^2}{D \times 2g} \quad \text{or} \quad \frac{12fLV^2}{D \times 2g} - \frac{4fLV^2}{D \times 2g} = \frac{v^2}{2g}$$

or $$\frac{8fLV^2}{D \times 2g} = \frac{v^2}{2g} \qquad ...(i)$$

But from continuity, $AV = av$ or $V = \frac{av}{A}$.

Substituting this value of V in equation (i), we get

$$\frac{8fL}{D \times 2g} \times \frac{a^2v^2}{A^2} = \frac{v^2}{2g} \quad \text{or} \quad \frac{8fL}{D} \times \frac{a^2}{A^2} = 1 \qquad \left(\text{Divide by } \frac{v^2}{2g}\right) ...(ii)$$

or $$\frac{8fL}{D} \times \frac{\left(\frac{\pi}{4}d^2\right)^2}{\left(\frac{\pi}{4}D^2\right)^2} = 1 \text{ or } \frac{8fL}{D} \times \frac{d^4}{D^4} = 1 \text{ or } d^4 = \frac{D^5}{8fL}$$

$\therefore$ $$d = \left(\frac{D^5}{8fL}\right)^{1/4} \qquad ...(11.29)$$

From equation (ii), $$\frac{8fL}{D} = \frac{A^2}{a^2}$$

$\therefore$ $$\frac{A}{a} = \sqrt{\frac{8fL}{D}} \qquad ...(11.30)$$

Equation (11.30) gives the ratio of the area of the supply pipe to the area of the nozzle and hence from this equation, the diameter of the nozzle can be obtained.

Problem 11.48 *A nozzle is fitted at the end of a pipe of length 300 m and of diameter 100 mm. For the maximum transmission of power through the nozzle, find the diameter of nozzle. Take f = .009.*

Solution. Given :

Length of pipe, $L = 300$ m
Diameter of pipe, $D = 100$ mm $= 0.1$ m
Co-efficient of friction, $f = .009$
Let the diameter of nozzle $= d$

For maximum transmission of power, the diameter of nozzle is given by relation (11.29) as

$$d = \left(\frac{D^5}{8fL}\right)^{1/4} = \left(\frac{0.1^5}{8 \times .009 \times 300}\right)^{1/4} = 0.02608 \text{ m} = \textbf{26.08 mm. Ans.}$$

Problem 11.49 *The head of water at the inlet of a pipe 2000 m long and 500 mm diameter is 60 m. A nozzle of diameter 100 mm at its outlet is fitted to the pipe. Find the velocity of water at the outlet of the nozzle if f = .01 for the pipe.*

Solution. Given :

Head of water at inlet of pipe, $H = 60$ m

Length of pipe, $L = 2000$ m
Dia. of pipe, $D = 500$ mm $= 0.50$ m
Dia. of nozzle at outlet, $d = 100$ mm $= 0.1$ m
Co-efficient of friction, $f = .01$
The velocity at outlet of nozzle is given by equation (11.25) as

$$v = \sqrt{\frac{2gH}{\left(1 + \frac{4fL}{D} \times \frac{a^2}{A^2}\right)}} = \sqrt{\frac{2 \times 9.81 \times 60}{1 + \frac{4 \times .01 \times 2000}{0.5}\left(\frac{\frac{\pi}{4}d^2}{\frac{\pi}{4}D^2}\right)^2}}$$

$$= \sqrt{\frac{2 \times 9.81 \times 60}{1 + \frac{4 \times .01 \times 2000}{0.5} \times \left(\frac{0.1 \times .1}{0.5 \times .5}\right)^2}} = \mathbf{30.61\ m/s.\ Ans.}$$

Problem 11.50 *Find the maximum power transmitted by a jet of water discharging freely out of nozzle fitted to a pipe = 300 m long and 100 mm diameter with co-efficient of friction as 0.01. The available head at the nozzle is 90 m.*

Solution. Given :
Length of pipe, $L = 300$ m
Dia. of pipe, $D = 100$ mm $= 0.1$ m
Co-efficient of friction, $f = .01$
Head available at nozzle, $= 90$ m
For maximum power transmission through the nozzle, the diameter at the outlet of nozzle is given by equation (11.29) as

$$d = \left(\frac{D^5}{8fL}\right)^{1/4} = \left[\frac{(0.1)^5}{8 \times .01 \times 300}\right]^{1/4} = .0254 \text{ m}$$

$\therefore$ Area at the nozzle, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} (.0254)^2 = .0005067\ \text{m}^2$.

The nozzle at the outlet, discharges water into atmosphere and hence the total head available at the nozzle is converted into kinetic head.

$\therefore$ Head available at outlet $= v^2/2g$ or $90 = v^2/2g$

$\therefore$ $v = \sqrt{2 \times 9.81 \times 90} = 42.02$ m/s

Discharge through nozzle, $Q = a \times v = .0005067 \times 42.02 = 0.02129\ \text{m}^3/\text{s}$

$\therefore$ Maximum power transmitted $= \dfrac{\rho g \times Q \times \text{Head at outlet of nozzle}}{1000}$

$$= \frac{1000 \times 9.81 \times 0.02129 \times 90}{1000} = \mathbf{18.796\ kW.\ Ans.}$$

Problem 11.51 *The rate of flow of water through a pipe of length 2000 m and diameter 1 m is 2 m^3/s. At the end of the pipe a nozzle of outside diameter 300 mm is fitted. Find the power transmitted*

through the nozzle if the head of water at inlet of the pipe is 200 m and co-efficient of friction for pipe is 0.01.

Solution. Given :

Length of pipe, $L = 2000$ m
Dia. of pipe, $D = 1$ m
Discharge, $Q = 2$ m^3/s
Dia. of nozzle, $d = 300$ mm $= 0.3$ m
Head at inlet of pipe, $H = 200$ m
Co-efficient of friction, $f = .01$

Now area of pipe, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} \times 1^2 = 0.7854$ m^2

Velocity of water through pipe, $V = \frac{Q}{A} = \frac{2.0}{0.7854} = 2.546$ m/s

Power transmitted through nozzle is given by equation (11.27) as

$$P = \frac{\rho g \,.\, a \,.\, v}{1000}\left[H - \frac{4fLV^2}{D \times 2g}\right]$$

$$= \frac{1000 \times 9.81 \times 2.0}{1000}\left[200 - \frac{4 \times .01 \times 2000 \times (2.546)^2}{1 \times 2 \times 9.81}\right] \quad (\because av = Q)$$

$$= \mathbf{3405.43\ kW.\ Ans.}$$

▶ 11.13 WATER HAMMER IN PIPES

Consider a long pipe AB as shown in Fig. 11.32 connected at one end to a tank containing water at a height of H from the centre of the pipe. At the other end of the pipe, a valve to regulate the flow of water is provided. When the valve is completely open, the water is flowing with a velocity, V in the pipe. If now the valve is suddenly closed, the momentum of the flowing water will be destroyed and consequently a wave of high pressure will be set up. This wave of high pressure will be transmitted along the pipe with a velocity equal to the velocity of sound wave and may create noise called knocking. Also this wave of high pressure has the effect of hammering action on the walls of the pipe and hence it is also known as water hammer.

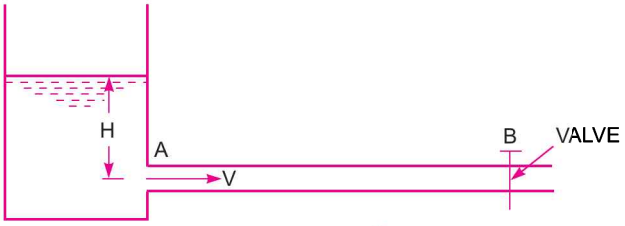


Fig. 11.32 *Water hammer.*

The pressure rise due to water hammer depends upon : (*i*) the velocity of flow of water in pipe, (*ii*) the length of pipe, (*iii*) time taken to close the valve, (*iv*) elastic properties of the material of the pipe. The following cases of water hammer in pipes will be considered :

1. Gradual closure of valve,
2. Sudden closure of valve and considering pipe rigid, and

3. Sudden closure of valve and considering pipe elastic.

11.13.1 Gradual Closure of Valve. Let the water is flowing through the pipe AB shown in Fig. 11.32, and the valve provided at the end of the pipe is closed gradually.

Let A = area of cross-section of the pipe AB,
L = length of pipe,
V = velocity of flow of water through pipe,
T = time in second required to close the valve, and
p = intensity of pressure wave produced.

Mass of water in pipe $AB = \rho \times$ volume of water $= \rho \times A \times L$

The valve is closed gradually in time 'T' seconds and hence the water is brought from initial velocity V to zero velocity in time seconds.

$$\therefore \text{ Retardation of water} = \frac{\text{Change of velocity}}{\text{Time}} = \frac{V - 0}{T} = \frac{V}{T}$$

$$\therefore \text{ Retarding force} = \text{Mass} \times \text{Retardation} = \rho AL \times \frac{V}{T} \quad ...(i)$$

If p is the intensity of pressure wave produced due to closure of the valve, the force due to pressure wave,

$$= p \times \text{area of pipe} = p \times A \quad ...(ii)$$

Equating the two forces, given by equations (i) and (ii),

$$\rho AL \times \frac{V}{T} = p \times A$$

$$\therefore \quad p = \frac{\rho LV}{T} \quad ...(11.31)$$

$$\text{Head of pressure, } H = \frac{p}{\rho g} = \frac{\rho LV}{\rho g \times T} = \frac{\rho LV}{\rho \times g \times T} \text{ or } H = \frac{LV}{gT} \quad ...(11.32)$$

(i) The valve closure is said to be gradual if $T > \frac{2L}{C}$ ...(11.33)

where t = time in sec, C = velocity of pressure wave

(ii) The valve closure is said to be sudden if $T < \frac{2L}{C}$ ...(11.34)

where C = velocity of pressure wave.

11.13.2 Sudden Closure of Valve and Pipe is Rigid. Equation (11.31) gives the relation between increase of pressure due to water hammer in pipe and the time required to close the valve. If $t = 0$, the increase in pressure will be infinite. But from experiments, it is observed that the increase in pressure due to water hammer is finite, even for a very rapid closure of valve. Thus equation (11.31) is valid only for (i) incompressible fluids and (ii) when pipe is rigid. But when a wave of high pressure is created, the liquids get compressed to some extent and also pipe material gets stretched. For a sudden closure of valve [the valve of t is small and hence a wave of high pressure is created] the following two cases will be considered :

(i) Sudden closure of valve and pipe is rigid, and
(ii) Sudden closure of valve and pipe is elastic.

Consider a pipe AB in which water is flowing as shown in Fig. 11.32. Let the pipe is rigid and valve fitted at the end B is closed suddenly.

Let A = Area of cross-section of pipe AB,
L = Length of pipe,
V = Velocity of flow of water through pipe,
p = Intensity of pressure wave produced,
K = Bulk modulus of water.

When the valve is closed suddenly, the kinetic energy of the flowing water is converted into strain energy of water if the effect of friction is neglected and pipe wall is assumed perfectly rigid.

$$\therefore \quad \text{Loss of kinetic energy} = \frac{1}{2} \times \text{mass of water in pipe} \times V^2$$

$$= \frac{1}{2} \times \rho AL \times V^2 \qquad (\because \text{mass} = \rho \times \text{volume} = \rho \times A \times L)$$

$$\text{Gain of strain energy} = \frac{1}{2}\left(\frac{p^2}{K}\right) \times \text{volume} = \frac{1}{2}\frac{p^2}{K} \times AL$$

Equating loss of kinetic energy to gain of strain energy

$$\therefore \quad \frac{1}{2} \rho AL \times V^2 = \frac{1}{2}\frac{p^2}{K} \times AL$$

or
$$p^2 = \frac{1}{2} \rho AL \times V^2 \times \frac{2K}{AL} = \rho K V^2$$

$$\therefore \quad p = \sqrt{\rho K V^2} = V\sqrt{K\rho} = V\sqrt{\frac{K\rho^2}{\rho}} \qquad \text{...(11.35)}$$

$$= \rho V \times C \qquad \left(\because \sqrt{K/\rho} = C\right) \text{...(11.36)}$$

where C = velocity* of pressure wave.

11.13.3 Sudden Closure of Valve and Pipe is Elastic. Consider the pipe AB in which water is flowing as shown in Fig. 11.32. Let the thickness 't' of the pipe wall is small compared to the diameter D of the pipe and also let the pipe is elastic.

Let E = Modulus of Elasticity of the pipe material,
$\frac{1}{m}$ = Poisson's ratio for pipe material,
p = Increase of pressure due to water hammer,
t = Thickness of the pipe wall,
D = Diameter of the pipe.

When the valve is closed suddenly, a wave of high pressure of intensity p will be produced in the water. Due to this high pressure p, circumferential and longitudinal stresses in the pipe wall will be produced.

Let f_l = Longitudinal stress in pipe
f_c = Circumferential stress in pipe,

The magnitude of these stresses are given as $f_l = \frac{pD}{4t}$ and $f_c = \frac{pD}{2t}$

Now from the knowledge of strength of material we know, strain energy stored in pipe material per unit volume

* For derivation of velocity of pressure wave, please refer to chapter 15.

$$= \frac{1}{2E}\left[f_l^2 + f_c^2 - \frac{2f_l \times f_c}{m}\right]$$

$$= \frac{1}{2E}\left[\left(\frac{pD}{4t}\right)^2 + \left(\frac{pD}{2t}\right)^2 - \frac{2 \times \frac{pD}{4t} \times \frac{pD}{2t}}{m}\right]$$

$$= \frac{1}{2E}\left[\frac{p^2D^2}{16t^2} + \frac{p^2D^2}{4t^2} - \frac{p^2D^2}{4mt^2}\right]$$

Taking $\frac{1}{m} = \frac{1}{4}$ $\left(i.e., \text{Poisson ratio} = \frac{1}{4}\right)$

$\therefore$ Strain energy stored in pipe material per unit volume

$$= \frac{1}{2E}\left[\frac{p^2D^2}{16t^2} + \frac{p^2D^2}{4t^2} - \frac{p^2D^2}{4t^2 \times 4}\right] = \frac{1}{2E} \times \frac{p^2D^2}{4t^2} = \frac{p^2D^2}{8Et^2}$$

Total volume of pipe material = $\pi D \times t \times L$.

$\therefore$ Total strain energy stored in pipe material

= Strain energy per unit volume × total volume

$$= \frac{p^2D^2}{8Et^2} \times \pi D \times t \times L = \frac{p^2\pi D^3 L}{8Et}$$

$$= \frac{p^2 \times \pi D^2 \times DL}{8Et} = \frac{p^2 A \times DL}{2Et} \qquad \left(\because \frac{\pi D^2}{4} = \text{Area of pipe} = A\right)$$

Now loss of kinetic energy of water $= \frac{1}{2} mV^2 = \frac{1}{2} \rho AL \times V^2$ $(\because m = \rho AL)$

Gain of strain energy in water $= \frac{1}{2}\left(\frac{p^2}{K}\right) \times$ volume $= \frac{1}{2}\frac{p^2}{K} \times AL$

Then, loss of kinetic energy of water = Gain of strain energy in water + Strain energy stored in pipe material.

$\therefore$ $$\frac{1}{2} \rho AL \times V^2 = \frac{1}{2}\left(\frac{p^2}{K}\right) \times AL + \frac{p^2 A \times DL}{2Et}$$

Divide by AL, $$\frac{\rho V^2}{2} = \frac{1}{2}\frac{p^2}{K} + \frac{p^2 D}{2Et} = \frac{p^2}{2}\left[\frac{1}{K} + \frac{D}{Et}\right] \text{ or } \rho V^2 = p^2\left[\frac{1}{K} + \frac{D}{Et}\right]$$

$\therefore$ $$p^2 = \frac{\rho V^2}{\frac{1}{K} + \frac{D}{Et}} \text{ or } p = \sqrt{\frac{\rho V^2}{\frac{1}{K} + \frac{D}{Et}}} = V \times \sqrt{\frac{\rho}{\frac{1}{K} + \frac{D}{Et}}} \qquad ...(11.37)$$

11.13.4 Time Taken by Pressure Wave to Travel from the Valve to the Tank and from Tank to the Valve

Let T = The required time taken by pressure wave
L = Length of the pipe
C = Velocity of pressure wave

Then total distance = $L + L = 2L$

$$\therefore \text{ Time, } T = \frac{\text{Distance}}{\text{Velocity of pressure wave}} = \frac{2L}{C}. \quad ...(11.38)$$

Problem 11.52 *The water is flowing with a velocity of 1.5 m/s in a pipe of length 2500 m and of diameter 500 mm. At the end of the pipe, a valve is provided. Find the rise in pressure if the valve is closed in 25 seconds. Take the value of C = 1460 m/s.*

Solution. Given :

Velocity of water, $V = 1.5$ m/s
Length of pipe, $L = 2500$ m
Diameter of pipe, $D = 500$ mm $= 0.5$ m
Time to close the valve, $T = 25$ seconds
Value of, $C = 1460$ m/s
Let the rise in pressure $= p$

The ratio, $\frac{2L}{C} = \frac{2 \times 2500}{1460} = 3.42$

From equation (11.33), we have if $T > \frac{2L}{C}$, the closure of valve is said to be gradual.

Here $T = 25$ sec and $\frac{2L}{C} = 3.42$

$\therefore$ $T > \frac{2L}{C}$ and hence valve is closed gradually.

For gradually closure of valve, the rise in pressure is given by equation (11.31) as

$$p = \frac{\rho VL}{T} = 1000 \times 2500 \times \frac{1.5}{25} = 150000 \text{ N/m}^2$$

$$= \frac{150000}{10^4} \frac{\text{N}}{\text{cm}^2} = \mathbf{15.0} \frac{\mathbf{N}}{\mathbf{cm^2}}. \textbf{ Ans.}$$

Problem 11.53 *If in Problem 11.52, the valve is closed in 2 sec, find the rise in pressure behind the valve. Assume the pipe to be rigid one and take Bulk modulus of water.*
i.e., $K = 19.62 \times 10^4$ N/cm^2.

Solution. Given :

$V = 1.5$ m/s, $L = 2500$ m
$D = 500$ mm $= 0.5$ m
Time to close the valve, $T = 2$ sec
Bulk modulus of water, $K = 19.62 \times 10^4$ N/cm^2
$= 19.62 \times 10^4 \times 10^4$ N/m^2 $= 19.62 \times 10^8$ N/m^2

Velocity of pressure wave is given by,

$$C = \sqrt{\frac{K}{\rho}} = \sqrt{\frac{19.62 \times 10^8}{1000}} = 1400 \text{ m/s} \qquad (\because \rho = 1000)$$

The ratio, $\frac{2L}{C} = \frac{2 \times 2500}{1400} = 3.57$ $\quad \therefore \quad T < \frac{2L}{C}$.

$\therefore$ From equation (11.34), if $T < \frac{2L}{C}$, valve is closed suddenly. For sudden closure of valve, when pipe is rigid, the rise in pressure is given by equation (11.35) or (11.36) as

$$p = V\sqrt{K\rho} = 1.5\sqrt{19.62 \times 10^8 \times 1000} \qquad (\because \rho = 1000)$$

$$= 210.1 \times 10^4 \text{ N/m}^2 = \mathbf{210.1 \text{ N/cm}^2. \text{ Ans.}}$$

Problem 11.54 *If in Problem 11.52, the thickness of the pipe is 10 mm and the valve is suddenly closed at the end of the pipe, find the rise in pressure if the pipe is considered to be elastic. Take $E = 19.62 \times 10^{10}$ N/m^2 for pipe material and $K = 19.62 \times 10^4$ N/cm^2 for water. Calculate the circumferential stress and longitudinal stress developed in the pipe wall.*

Solution. Given :

$V = 1.5$ m/s, $L = 2500$ m, $D = 0.5$ m

Thickness of pipe, $t = 10$ mm $= .01$ m

Modulus of elasticity, $E = 19.62 \times 10^{10}$ N/m^2

Bulk modulus, $K = 19.62 \times 10^4$ N/cm^2 $= 19.62 \times 10^8$ N/m^2

For sudden closure of the valve for an elastic pipe, the rise in pressure is given by equation (11.37) as

$$p = V \times \sqrt{\frac{\rho}{\left(\frac{1}{K} + \frac{D}{Et}\right)}} = 1.5 \times \sqrt{\frac{1000}{\left(\frac{1}{19.62 \times 10^8} + \frac{0.5}{19.62 \times 10^{10} \times .01}\right)}}$$

$$= 1.5 \times \sqrt{\frac{1000}{\left(5.09 \times 10^{-10} + 2.54 \times 10^{-10}\right)}}$$

$$= 1715510 \text{ N/m}^2 = \mathbf{171.55 \text{ N/cm}^2. \text{ Ans.}}$$

Circumferential stress (f_c) is given by

$$= \frac{p \times D}{2t} = \frac{171.55 \times 0.5}{2 \times .01} = 4286.9 \text{ N/m}^2$$

Longitudinal stress is given by, $f_l = \frac{p \times D}{4t} = \frac{171.55 \times 0.5}{4 \times .01} = \mathbf{2143.45 \text{ N/m}^2. \text{ Ans.}}$

Problem 11.55 *A valve is provided at the end of a cast iron pipe of diameter 150 mm and of thickness 10 mm. The water is flowing through the pipe, which is suddenly stopped by closing the valve. Find the maximum velocity of water, when the rise of pressure due to sudden closure of valve is 196.2 N/cm^2. Take K for water as 19.62×10^4 N/cm^2 and E for cast iron pipe as 11.772×10^6 N/cm^2.*

Solution. Given :

Diameter of pipe, $D = 150$ mm $= 0.15$ m

Thickness of pipe, $t = 10$ mm $= .01$ m

Rise of pressure, $p = 196.2 \text{ N/cm}^2 = 196.2 \times 10^4 \text{ N/m}^2$

Bulk modulus, $K = 19.62 \times 10^4 \text{ N/cm}^2 = 19.62 \times 10^8 \text{ N/m}^2$

Modulus of elasticity, $E = 11.772 \times 10^6 \text{ N/cm}^2 = 11.772 \times 10^{10} \text{ N/m}^2$

For sudden closure of valve and when pipe is elastic, the pressure rise is given by equation (11.37) as

$$p = V \times \sqrt{\frac{\rho}{\left(\frac{1}{K} + \frac{D}{Et}\right)}} = V \times \sqrt{\frac{1000}{\left(\frac{1}{19.62 \times 10^8} + \frac{0.15}{11.772 \times 10^{10} \times .01}\right)}}$$

or
$$196.2 \times 10^4 = V \times \sqrt{\frac{1000}{5.09 \times 10^{-10} + 1.274 \times 10^{-10}}}$$

$$= V \times \sqrt{\frac{1000}{6.364 \times 10^{-10}}} = V \times 125.27 \times 10^4$$

$\therefore$
$$V = \frac{196.2 \times 10^4}{125.27 \times 10^4} = 1.566 \text{ m/s}$$

$\therefore$ Maximum velocity = **1.566 m/s. Ans.**

▶ 11.14 PIPE NETWORK

A pipe network is an interconnected system of pipes forming several loops or circuits. The pipe network is shown in Fig. 11.33. The examples of such networks of pipes are the municipal water distribution systems in cities and laboratory supply system. In such system, it is required to determine the distribution of flow through the various pipes of the network. The following are the necessary conditions for any network of pipes :

(*i*) The flow into each junction must be equal to the flow out of the junction. This is due to continuity equation.

(*ii*) The algebraic sum of head losses round each loop must be zero. This means that in each loop, the loss of head due to flow in clockwise direction must be equal to the loss of head due to flow in anticlockwise direction.

(*iii*) The head loss in each pipe is expressed as $h_f = rQ^n$. The value of r depends upon the length of pipe, diameter of pipe and co-efficient of friction of pipe. The value of n for turbulent flow is 2. We know that,

$$h_f = \frac{4 \times f \times L \times V^2}{D \times 2g} = \frac{4fL \times \left(\frac{Q}{A}\right)^2}{D \times 2g} \qquad \left(\because V = \frac{Q}{A} = \frac{Q}{\frac{\pi}{4}D^2}\right)$$

$$= \frac{4fL \times Q^2}{D \times 2g \times \left(\frac{\pi}{4}D^2\right)^2} = \frac{4fL \times Q^2}{D \times 2g \times \left(\frac{\pi}{4}\right)^2 \times D^4}$$

$$= \frac{4f \times L \times Q^2}{2g \times \left(\frac{\pi}{4}\right)^2 \times D^5}$$

$$= rQ^2 \qquad \text{...(11.39)} \quad \left(\text{where} \frac{4f \times L}{2g \times \left(\frac{\pi}{4}\right)^2 \times D^5} = r \right)$$

This head loss will be positive, when the pipe is a part of loop and the flow in the pipe is clockwise.

Generally, the pipe network problems are difficult to solve analytically. Hence the methods of successive approximations are used. '**Hardy Cross Method**' is one such method which is commonly used.

11.14.1 Hardy Cross Method. The procedure for Hardy Cross Method is as follows :

1. In this method a trial distribution of discharges is made arbitrary but in such a way that continuity equation is satisfied at each junction (or node).
2. With the assumed values of Q, the head loss in each pipe is calculated according to equation (11.39).
3. Now consider any loop (or circuits). The algebraic sum of head losses round each loop must be zero. This means that in each loop, the loss of head due to flow in clockwise direction must be equal to the loss of head due to flow in anticlockwise direction.
4. Now calculate the net head loss around each loop considering the head loss to be positive in clockwise flow and to be negative in anticlockwise flow.

If the net head loss due to assumed values of Q round the loop is zero, then the assumed values of Q in that loop is correct. But if the net head loss due to assumed values of Q is not zero, then the assumed values of Q are corrected by introducing a correction ΔQ for the flows, till the circuit is balanced.

The correction factor ΔQ* is obtained by

$$\Delta Q = \frac{-\sum r Q_0^n}{\sum r n Q_0^{n-1}} \qquad \text{...(11.40)}$$

For turbulent flow, the value of $n = 2$ and hence above correction factor becomes as

$$\Delta Q = \frac{-\sum r Q_0^2}{\sum 2r Q_0} \qquad \text{...(11.41)}$$

5. If the value of ΔQ comes out to be positive, then it should be added to the flows in the clockwise direction ($\because$ the flows in clockwise direction in a loops are considered positive) and subtracted from the flows in the anticlockwise direction.
6. Some pipes may be common to two circuits (or two loops), then the two corrections are applied to these pipes.

---

* Let for any pipe Q_0 = assumed discharge and Q = correct discharge, then

$$Q = Q_0 + \Delta Q$$

$\therefore$ Head loss for the pipe, $h_f = rQ^2 = r(Q_0 + \Delta Q)^2$.

For complete circuit, the net head loss, $\Sigma h_f = \Sigma (rQ^2) = \Sigma r (Q_0 + \Delta Q)^2 = \Sigma r(Q_0^2 + 2Q_0 \Delta Q + \Delta Q^2)$

$= \Sigma r (Q_0^2 + 2Q_0 \Delta Q)$ As ΔQ is small compared with Q_0 and hence ΔQ^2 can be neglected.

$\therefore$ $\Sigma rQ^2 = \Sigma rQ_0^2 + \Sigma r \times 2Q_0 \Delta Q$

For the correct distribution, the net head loss for a circuit should be zero (*i.e.*, $\Sigma h_f = \Sigma (rQ^2) = 0$)

$\therefore$ $\Sigma rQ_0^2 + \Sigma r \times 2Q_0 \Delta Q = 0$

or $\Sigma rQ_0^2 + \Delta Q \, \Sigma r \times 2Q_0 = 0$ [As ΔQ is same for one circuit, hence it can be taken out of the summation]

$\therefore$ $$\Delta Q = \frac{-\Sigma r Q_0^2}{\Sigma 2r Q_0}.$$

7. After the corrections have been applied to each pipe in a loop and to all loops, a second trial calculation is made for all loops. The procedure is repeated till ΔQ becomes negligible.

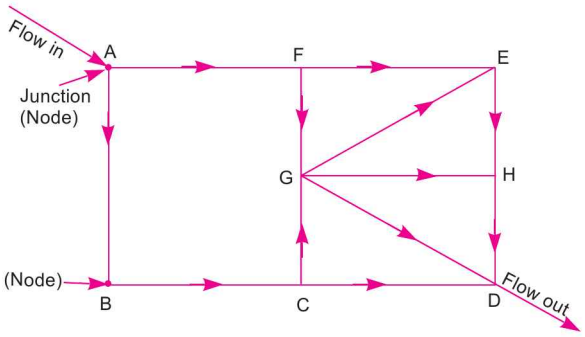


Fig. 11.33 *Pipe network.*

[Loops are : *ABCGFA, FEGF, GEHG, GHDG and GCDG*]

Problem 11.56 *Calculate the discharge in each pipe of the network shown in Fig. 11.34. The pipe network consists of 5 pipes. The head loss h_f in a pipe is given by $h_f = rQ^2$. The values of r for various pipes and also the inflow or outflows at nodes are shown in the figure.*

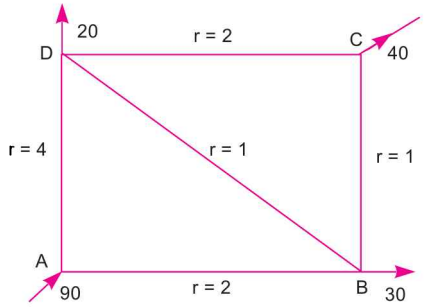


Fig. 11.34

Solution. Given :

Inflow at node A = 90, outflow at B = 30, at C = 40 and at D = 20.

Values of r for AB = 2, for BC = 1, for CD = 2, for AD = 4 and for BD = 1.

For the first trial, the discharges are assumed as shown in Fig. 11.34 (*a*) so that continuity is satisfied at each node (*i.e.*, flow into a node = flow out of the node). For this distribution of discharge, the corrections ΔQ for the loops ABD and BCD are calculated.

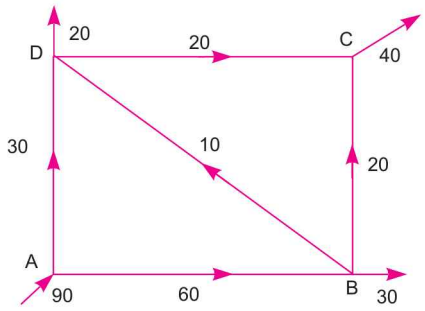


Fig. 11.34(*a*)

First Trial

Loop ADB					Loop DCB				
Pipe	*r*	Q_0	$h_f = rQ_0^2$	$2rQ_0$	*Pipe*	*r*	Q_0	$h_f = rQ_0^2$	$2rQ_0$
AD	4	30	$4 \times 30^2 = 3600$	$2 \times 4 \times 30 = 240$	*DC*	2	20	$2 \times 20^2 = 800$	$2 \times 2 \times 20 = 80$
DB	1	10	$-1 \times 10^2 = -100$	$2 \times 1 \times 10 = 20$	*CB*	1	20	$-1 \times 20^2 = -400$	$2 \times 1 \times 20 = 40$
AB	2	60	$-2 \times 60^2 = -7200$	$2 \times 2 \times 60 = 240$	*BD*	1	10	$1 \times 10^2 = 100$	$2 \times 1 \times 10 = 20.$
			$\Sigma rQ_0^2 = -3700,$	$\Sigma 2rQ_0 = 500,$				$\Sigma 2\, rQ_0^2 = 500$	$\Sigma 2rQ_0 = 140$
$\therefore$			$\Delta Q = \frac{-\Sigma\, r\, Q_0^2}{\Sigma\, 2r\, Q_0} = \frac{-(-3700)}{500} = 7.4$		$\therefore$			$\Delta Q = \frac{-\Sigma\, r\, Q_0^2}{\Sigma\, 2r\, Q_0} \frac{-500}{140} = -3.57 \simeq -3.6.$	
In the loop *ADB*, the head loss h_f is negative in pipes *DB* and *AB* as the direction of discharges in these pipes is anticlockwise.					The head loss in pipe *BC* for loop *DCB* is negative as the direction of discharge in pipe *BC* in anticlockwise.				
As ΔQ is positive for loop *ADB,* hence it should be added to the flow in the clockwise direction and subtractedfrom the flow in the anticlockwise direction. Hence the corrected flow for second trial for loop *ADB* will be as follows :					As ΔQ is negative for loop *DCB*, hence it should be subtracted from the flow in the clockwise direction and added to the flow in the anticlockwise direction. Hence corrected flow for second trial for loop *DCB* will be as follows :				
Pipe *AD* = 30 + 7.4 = 37.4 (flow is clockwise)					Pipe *DC* = 20 – 3.6 = 16.4				
Pipe *AB* = 60 – 7.4 = 52.6 (flow is anticlockwise)					Pipe *BC* = 20 + 3.6 = 23.6				
Pipe *BD* = 10 – 7.4 = 2. 6 (flow is anticlockwise)					Pipe *BD** = 2.6 – 3.6 = – 1				

Note. The pipe *BD* is common to two loops (*i.e.*, loop *ADB* and loop *DCB*). Hence this pipe will get two corrections. After the two corrections, the resultant flow in pipe *BD* is negative in loop *DCB*. Hence the direction of flow will be anticlockwise in pipe *BD* for loop *DCB*.

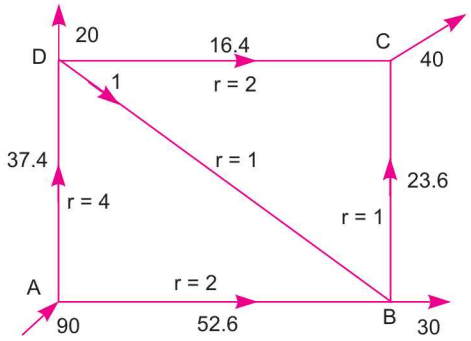


Fig. 11.34 (*b*)

The distribution of discharges in various pipes for second trial is shown in Fig. 11.34 (*b*). For second trial the correction ΔQ for loops *ADB* and *DCB* are calculated as follows :

Loop ADB

Pipe	r	Q_0	$h_f = rQ_0^2$	$2rQ_0$
AD	4	37.4	$4 \times 37.4^2 = 5595$	$2 \times 4 \times 37.4 = 299.2$
DB	1	1	$1 \times 1^2 = 1$	$2 \times 1 \times 1 = 2$
AB	2	52.6	$-2 \times 52.6^2 = -5533.5$	$2 \times 2 \times 52.6 = 210.4$

$\Sigma rQ_0^2 = 62.54, \quad \Sigma 2rQ_0 = 511.6$

$$\therefore \Delta Q = \frac{-\sum r Q_0^2}{\sum 2r Q_0} = \frac{62.54}{-511.6}$$

$$= -0.122 \simeq -0.1$$

As ΔQ is negative, hence it should be subtracted from the flow in the clockwise direction and added to the flow in the anticlockwise direction

As the correction (ΔQ) is small (*i.e.*, $\Delta Q = -0.1$), this correction is applied and further trials are discontinued.

Hence corrected flow for loop ADB will be as follows :

For pipe AD, $Q_0 = 37.4 - 0.1 = 37.3$ (as flow is clockwise)

For pipe DB, $Q_0 = 1 - 0.1 = 0.9$ (as flow is clockwise)

For pipe AB, $Q_0 = 52.6 + 0.1 = 52.7$ (as flow is anti-clockwise)

Loop DCB

Pipe	r	Q_0	$h_f = rQ_0^2$	$2rQ_0$
DC	2	16.4	$2 \times 16.4^2 = 537.9$	$2 \times 2 \times 16.4 = 65.6$
CB	1	23.6	$-1 \times 23.6^2 = -556.9$	$2 \times 1 \times 23.6 = 47.2$
BD	1	1	$-1 \times 1^2 = -1$	$2 \times 1 \times 1 = 2$

$\Sigma rQ_0^2 = -20, \quad \Sigma 2rQ_0 = 114.8$

$$\therefore \Delta Q = \frac{-\sum r Q_0^2}{\sum 2r Q_0} = \frac{-(-20)}{114.8}$$

$$= \frac{20}{114.8} = 0.174$$

$$\simeq 0.2$$

As ΔQ is positive, hence it should be added to the flow in the clockwise direction and subtracted from the flow in the anticlockwise direction.

As the correction (ΔQ) is small (*i.e.*, $\Delta Q = 0.2$), this correction is applied and further trials are discontinued.

Hence corrected flow for loop DCB will be as follows :

For pipe DC, $Q_0 = 16.4 + 0.2 = 16.6$ (clockwise flow)

For pipe CB, $Q_0 = 23.6 - 0.2 = 23.4$ (anticlockwise flow)

For pipe BD, $Q_0 = 0.9 - 0.2 = 0.7$ (anticlockwise flow)

The final distribution of discharges in each pipe is as follows :

Discharge in pipe $AD = 37.3$ from A to D

$AB = 52.7$ from A to B

$DB = 0.7$ from D to B

$DC = 16.6$. from D to C

$BC = 23.4$ from B to C

The final discharge in each pipe is shown in Fig. 11.34 (*c*)

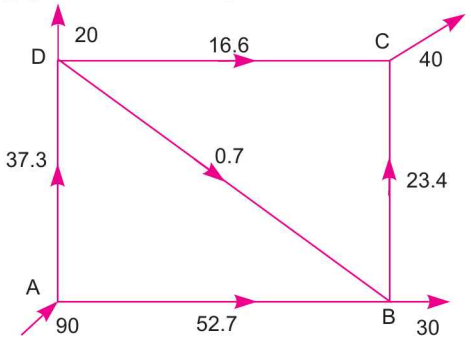


Fig. 11.34 (*c*)

Note. The pipe *DB* is common to two loop (*i.e.*, loops *ADB* and loop *DBC*). Hence this pipe will get two corrections. For loop *ADB,* the correction $\Delta Q = -0.1$ and hence the corrected flow in pipe *DB* is $1 - 0.1 = 0.9$. Now again, the correction is applied to pipe *DB* when we consider loop *DBC*. For loop *DBC*, the correction $\Delta Q = 0.2$ but flow is anticlockwise and hence the final correct flow in pipe *DB* will be $0.9 - 0.2 = 0.7$.

HIGHLIGHTS

1. The energy loss in pipe is classified as major energy loss and minor energy losses. Major energy loss is due to friction while minor energy losses are due to sudden expansion of pipe, sudden contraction of pipe, bend in pipe and an obstruction in pipe.
2. Energy loss due to friction is given by Darcy Formula, $h_f = \dfrac{4fLV^2}{d \times 2g}$.
3. The head loss due to friction in pipe can also be calculated by Chezy's formula.

   $V = C\sqrt{mi}$ Chezy's formula

   where C = Chezy's Constant

   m = Hydrualic mean depth = $\dfrac{d}{4}$ (for pipe running full)

   V = Velocity of flow

   i = Loss of head per unit length = $\dfrac{h_f}{L}$

   $\therefore$ $h_f = L \times i$, where i is obtained from Chezy's formula.
4. Loss of head due to sudden expansion of pipe, $h_c = \dfrac{(V_1 - V_2)^2}{2g}$

   where V_1 = Velocity in small pipe, V_2 = Velocity in large pipe.
5. Loss of head due to sudden contraction of pipe, $h_c = \left(\dfrac{1}{C_c} - 1\right)\dfrac{V_2^2}{2g}$

   where C_c = co-efficient of contraction = $0.375\ \dfrac{V_2^2}{2g}$ ...(For $C_c = 0.62$)

   $= 0.5\ \dfrac{V_2^2}{2g}$ ...(if value of C_c is not given)
6. Loss of head at the entrance of a pipe, $h_i = 0.5\ \dfrac{V^2}{2g}$.
7. Loss of head at the exit of pipe, $h_o = \dfrac{V^2}{2g}$.
8. The line representing the sum of pressure head and datum head with respect to some reference line is called hydraulic gradient line (H.G.L.) while the line representing the sum of pressure head, datum head and velocity head with respect to some reference line is known as total energy line (T.E.L.).
9. Syphon is a long bent pipe used to transfer liquids from a reservoir at a higher level to another reservoir at a lower level, when the two reservoirs are separated by a high level ground.
10. The maximum vacuum created at the summit of syphon is only 7.4 m of water.
11. When pipes of different lengths and different diameters are connected end to end, pipes are called in series or compound pipes. The rate of flow through each pipe connected in series is same.
12. A single pipe of uniform diameter, having same discharge and same loss of head as compound pipe consisting of several pipes of different lengths and diameters, is known as equivalent pipe. The diameter of equivalent pipe is called equivalent size of the pipe.

13. The equivalent size of the pipe is obtained from

$$\frac{L}{d^5} = \frac{L_1}{d_1^5} + \frac{L_2}{d_2^5} + \frac{L_3}{d_3^5}$$

where L = equivalent length of pipe = $L_1 + L_2 + L_3$
d_1, d_2, d_3 = diameters of pipes connected in series
d = equivalent size of the pipes.

14. When the pipes are connected in parallel, the loss of head in each pipe is same. The rate of flow in main pipe is equal to sum of the rate of flow in each pipe, connected in parallel.

15. For solving problems for branched pipes, the three basic, equations *i.e.*, continuity, Bernoulli's and Darcy's equations are used.

16. Power transmitted in kW through pipe is given by $P = \dfrac{\rho g \times Q \times (H - h_f)}{1000}$

where Q = discharge through pipe = area × velocity = $\dfrac{\pi}{4} d^2 \times V$

H = total head at inlet of pipe

h_f = head lost due to friction

$= \dfrac{4fLV^2}{d \times 2g}$, where L = Length of pipe

In S.I. units, power transmitted is given by, Power = $\dfrac{\rho g \times Q \times (H - h_f)}{1000}$ kW.

17. Efficiency of power transmission through pipes, $\eta = \dfrac{H - h_f}{H}$.

18. Condition for maximum transmission of power through pipe, $h_f = \dfrac{H}{3}$ and maximum efficiency = 66.7%.

19. The velocity of water at the outlet of the nozzle is $v = \sqrt{\dfrac{2gH}{\left(1 + \dfrac{4fL}{D} \times \dfrac{a^2}{A^2}\right)}}$

where H = head at the inlet of the pipe, L = length of the pipe,
D = diameter of the pipe, a = area of the nozzle at outlet,
A = area of the pipe.

20. The power transmitted through nozzle, $P = \dfrac{\rho g \times Q}{1000}\left[H - \dfrac{4fLV^2}{D \times 2g}\right]$

and the efficiency of power transmission through nozzle, $\eta = \dfrac{1}{\left[1 + \dfrac{4fL}{D} \times \dfrac{a^2}{A^2}\right]}$

21. Condition for maximum power transmission through nozzle, $h_f = \dfrac{H}{3}$.

22. Diameter of nozzle for maximum power transmission through nozzle is, $d = \left(\frac{D^5}{8fL}\right)^{1/4}$

where d = diameter of the nozzle at outlet, D = diameter of the pipe,
L = length of the pipe, f = co-efficient of friction for pipe.

23. When a liquid is flowing through a long pipe fitted with a valve at the end of the pipe and the valve is closed suddenly, a pressure wave of high intensity is produced behind the valve. This pressure wave of high intensity is having the effect of hammering action on the walls of the pipe. This phenomenon is known as water hammer.

24. The intensity of pressure rise due to water hammer is given by

$$p = \frac{\rho LV}{T}$$ ... when valve is closed gradually.

$$= V\sqrt{K\rho}$$ ... when valve is closed suddenly and pipe is assumed rigid

$$= V \times \sqrt{\frac{\rho}{\frac{1}{K} + \frac{D}{Et}}}$$ ... when valve is closed suddenly and pipe is elastic.

where L = Length of pipe,
T = Time required to close the valve,
D = Diameter of the pipe,
t = Thickness of the pipe wall,
V = Velocity of flow,
K = Bulk modulus of water,
E = Modulus of elasticity for pipe material.

25. If the time required to close the valve :

$$T > \frac{2L}{C}$$ ... the valve closure is said to be gradual,

$$T < \frac{2L}{C}$$ ... the valve closure is said to be sudden

where L = length of pipe,

C = velocity of pressure wave produced due to water hammer = $\sqrt{\frac{K}{\rho}}$.

EXERCISE

(A) THEORETICAL PROBLEMS

1. How will you determine the loss of head due to friction in pipes by using (*i*) Darcy Formula and (*ii*) Chezy's formula ?

2. (*a*) What do you understand by the terms : Major energy loss and minor energy losses in pipes ?
(*b*) What do you understand by total energy line, hydraulic gradient line, pipes in series, pipes in parallel and equivalent pipe ?

3. (*a*) Derive an expression for the loss of head due to : (*i*) Sudden enlargement and (*ii*) Sudden contraction of a pipe.
(*b*) Obtain expression for head loss in a sudden expansion in the pipe. List all the assumptions made in the derivation.

4. Define and explain the terms : (*i*) Hydraulic gradient line and (*ii*) Total energy line.

5. Show that the loss of head due to sudden expansion in pipe line is a function of velocity head.
6. What is a syphon ? On what principle it works ?
7. What is a compound pipe ? What will be loss of head when pipes are connected in series ?
8. Explain the terms : (*i*) Pipes in parallel (*ii*) Equivalent pipe and (*iii*) Equivalent size of the pipe.
9. Find an expression for the power transmission through pipes. What is the condition for maximum transmission of power and corresponding efficiency of transmission ?
10. Prove that the head loss due to friction is equal to one-third of the total head at inlet for maximum power transmission through pipes or nozzles.
11. Prove that the velocity through nozzle is given by $v = \sqrt{\frac{2gH}{1 + \frac{4fL}{D} \times \frac{a^2}{A^2}}}$

    where a = Area of nozzle at outlet, A = Area of the pipe.
12. Show that the diameter of the nozzle for maximum transmission of power is given by $d = \left(\frac{D^5}{8fL}\right)^{1/4}$

    where D = Diameter of pipe, L = Length of pipe.
13. Find an expression for the ratio of the outlet area of the nozzle to the area of the pipe for maximum transmission of power.
14. Explain the phenomenon of Water Hammer. Obtain an expression for the rise of pressure when the flowing water in a pipe is brought to rest by closing the valve gradually.
15. Show that the pressure rise due to sudden closure of a valve at the end of a pipe, through which water is flowing is given by $p = V\sqrt{\frac{d}{\frac{1}{K} + \frac{D}{Et}}}$

    where V = Velocity of flow, D = Diameter of pipe, E = Young's Modulus, K = Bulk Modulus and t = Thickness of pipe.
16. Three pipes of different diameters and different lengths are connected in series to make a compound pipe. The ends of this compound pipe are connected with two tanks whose difference of water level is H. If co-efficient of friction for these pipes is same, then derive the formula for the total head loss, neglecting first the minor losses and then including them.
17. For the two cases of flow in a sudden contraction in a pipeline and flow in a sudden expansion in a pipe line, draw the flow pattern, piezometric grade line and total energy line.
18. What do you mean by "equivalent pipe" and "flow through parallel pipes" ?
19. (*a*) Define and explain the terms : (*i*) Hydraulic gradient line and (*ii*) total energy line.
    (*b*) What do you mean by equivalent pipe ? Obtain an expression for equivalent pipe.

    (Delhi University, December 2002)

(B) NUMERICAL PROBLEMS

1. Find the head loss due to friction in a pipe of diameter 250 mm and length 60 m, through which water is flowing at a velocity of 3.0 m/s using (*i*) Darcy formula and (*ii*) Chezy's Formula for which $C = 55$. Take ν for water = .01 stoke. [**Ans.** (*i*) 1.182, (*ii*) 2.856]
2. Find the diameter of a pipe of length 2500 m when the rate of flow of water through the pipe is 0.25 m^3/s and head loss due to friction is 5 m. Take $C = 50$ in Chezy's formula. [**Ans.** 605 mm]

3. An oil of Kinematic Viscosity 0.5 stoke is flowing through a pipe of diameter 300 mm at the rate of 320 litres per sec. Find the head lost due to friction for a length of 60 m of the pipe. [**Ans.** 5.14 m]
4. Calculate the rate of flow of water through a pipe of diameter 300 mm, when the difference of pressure head between the two ends of a pipe 400 m apart is 5 m of water. Take the value of $f = .009$ in the formula

$$h_f = \frac{4fLV^2}{d \times 2g}$$

[**Ans.** 0.101 m^3/s]

5. The discharge through a pipe is 200 litres/s. Find the loss of head when the pipe is suddenly enlarged from 150 mm to 300 mm diameter. [**Ans.** 3.672 m]
6. The rate of flow of water through a horizontal pipe is 0.3 m^3/s. The diameter of the pipe is suddenly enlarged from 250 mm to 500 mm. The pressure intensity in the smaller pipe is 13.734 N/cm^2. Determine : (*i*) loss of head due to sudden enlargement, (*ii*) pressure intensity in the large pipe and (*iii*) power lost due to enlargement. [**Ans.** (*i*) 1.07 m, (*ii*) 14.43 N/cm^2, (*iii*) 3.15 kW]
7. A horizontal pipe of diameter 400 mm is suddenly contracted to a diameter of 200 mm. The pressure intensities in the large and smaller pipe is given as 14.715 N/cm^2 and 12.753 N/cm^2 respectively. If $C_c = 0.62$, find the loss of head due to contraction. Also determine the rate of flow of water. [**Ans.** (*i*) 0.571 m, (*ii*) 171. 7 litres/s]
8. Water is flowing through a horizontal pipe of diameter 300 mm at a velocity of 4 m/s. A circular solid plate of diameter 200 mm is placed in the pipe to obstruct the flow. If $C_c = 0.62$, find the loss of head due to obstruction in the pipe. [**Ans.** 2.953 m]
9. Determine the rate of flow of water through a pipe of diameter 10 cm and length 60 cm when one end of the pipe is connected to a tank and other end of the pipe is open to the atmosphere. The height of water in the tank from the centre of the pipe is 5 cm. Pipe is given as horizontal and value of $f = .01$. Consider minor losses. [**Ans.** 15.4 litres/s]
10. A horizontal pipe-line 50 m long is connected to a water tank at one end and discharges freely into the atmosphere at the other end. For the first 30 m of its length from the tank, the pipe is 200 mm diameter and its diameter is suddenly enlarged to 400 mm. The height of water level in the tank is 10 m above the centre of the pipe. Considering all minor losses, determine the rate of flow. Take $f = .01$ for both sections of the pipe. [**Ans.** 164.13 litres/s]
11. Determine the difference in the elevations between the water surfaces in the two tanks which are connected by a horizontal pipe of diameter 400 mm and length 500 m. The rate of flow of water through the pipe is 200 litres/s. Consider all losses and take the value of $f = .009$. [**Ans.** 11.79 m]
12. For the problems 9, 10 and 11 draw the hydraulic gradient lines (H.G.L.) and total energy lines (T.E.L.)
13. A syphon of diameter 150 mm connects two reservoirs having a difference in elevation of 15 m. The length of the syphon is 400 m and summit is 4.0 m above the water level in the upper reservoir. The length of the pipe from upper reservoir to the summit is 80 m. Determine the discharge through the syphon and also pressure at the summit. Neglect minor losses. The co-efficient of friction, $f = .005$. [**Ans.** 41.52 litres/s, – 7.281 m of water]
14. A syphon of diameter 200 mm connects two reservoirs having a difference in elevation as 20 m. The total length of the syphon is 800 m and the summit is 5 m above the water level in the upper reservoir. If the separation takes place at 2.8 m of water absolute find the maximum length of syphon from upper reservoir to the summit. Take $f = .004$ and atmospheric pressure = 10.3 m of water. [**Ans.** 87.52 m]
15. Three pipes of lengths 800 m, 600 m and 300 m and of diameters 400 mm, 300 mm and 200 mm respectively are connected in series. The ends of the compound pipe is connected to two tanks, whose water surface levels are maintained at a difference of 15 m. Determine the rate of flow of water through the pipes if $f = .005$. What will be diameter of a single pipe of length 1700 m and $f = .005$, which replaces the three pipes ? [**Ans.** 0.0848 m^3/s, 266.5 mm]

16. Two pipes of lengths 2500 m each and diameters 80 cm and 60 cm respectively, are connected in parallel. The co-efficient of friction for each pipe is 0.006. The total flow is equal to 250 litres/s. Find the rate of flow in each pipe. [**Ans.** 0.1683 m^3/s, 0.0817 m^3/s]

17. A pipe of diameter 300 mm and length 1000 m connects two reservoirs, having difference of water levels as 15 m. Determine the discharge through the pipe. If an additional pipe of diameter 300 mm and length 600 m is attached to the last 600 m length of the existing pipe, find the increase in the discharge. Take $f = .02$ and neglect minor losses. [**Ans.** 0.0742 m^3/s, 0.0258 m^3/s]

18. Two sharp ended pipes of diameters 60 mm and 100 mm respectively, each of length 150 m are connected in parallel between two reservoirs which have a difference of level of 15 m. If co-efficient of friction for each pipe is 0.08, calculate the rate of flow for each pipe and also the diameter of a single pipe 150 m long which would give the same discharge if it were substituted for the original two pipes. [**Ans.** 0.0017, .00615, 110 mm]

19. Three reservoirs A, B and C are connected by a pipe system having length 700 m, 1200 m and 500 m and diameters 400 mm, 300 mm and 200 mm respectively. The water levels in reservoir A and B from a datum line are 50 m and 45 m respectively. The level of water in reservoir C is below the level of water in reservoir B. Find the discharge into or from the reservoirs B and C if the rate of flow from reservoir A is 150 litres per sec. Find the height of water level in the reservoir C. Take $f = .005$ for all pipes. [**Ans.** .005 m^3/s, .095 m^3/s, 24.16 m]

20. A pipe of diameter 300 mm and length 3000 m is used for the transmission of power by water. The total head at the inlet of the pipe is 400 m. Find the maximum power available at the outlet of the pipe. Take $f = .005$. [**Ans.** 667.07 kW]

21. A pipe line of length 2100 m is used for transmitting 103 kW. The pressure at the inlet of the pipe is 392.4 N/cm^2. If the efficiency of transmission is 80%, find the diameter of the pipe. Take $f = .005$. [**Ans.** 136 mm]

22. A nozzle is fitted at the end of a pipe of length 400 m and of diameter 150 mm. For the maximum transmission of power through the nozzle, find the diameter of the nozzle. Take $f = .008$. [**Ans.** 41.5 mm]

23. The head of water at the inlet of a pipe of length 1500 m and of diameter 400 mm is 50 m. A nozzle of diameter 80 mm at the outlet, is fitted to the pipe. Find the velocity of water at the outlet of the nozzle if $f = .01$ for the pipe. [**Ans.** 28.12 m/s]

24. The rate of flow of water through a pipe of length 1500 m and diameter 800 mm is 2 m^3/s. At the end of the pipe a nozzle of outside diameter 200 mm is fitted. Find the power transmitted through the nozzle if the head of water at the inlet of the pipe is 180 m and $f = .01$ for pipe. [**Ans.** 2344.7 kW]

25. The water is flowing with a velocity of 2 m/s in a pipe of length 2000 m and of diameter 600 mm. At the end of the pipe, a valve is provided. Find the rise in pressure if the valve is closed in 20 seconds. Take the value of $C = 1420$ m/s. [**Ans.** 20 N/cm^2]

26. If the valve in the problem 25 is closed in 1.5 sec, find the rise in pressure. Take bulk modulus of water $= 19.62 \times 10^4$ N/cm^2 and consider pipe as rigid one. [**Ans.** 186.75 N/cm^2]

27. If in the problem 25, the thickness of the pipe is 10 mm and the valve is closed suddenly. Find the rise in pressure if the pipe is considered to be elastic. Take value of $E = 19.62 \times 10^6$ N/cm^2 for pipe material and $K = 19.62 \times 10^4$ N/cm^2 for water. Calculate the circumferential stress and longitudinal stress developed in the pipe wall. [**Ans.** $p = 221.47$ N/cm^2, $f_c = 6644.1$ N/cm^2, $f_l = 3322$ N/cm^2]

28. The difference in water surface levels in two tanks, which are connected by two pipes in series of lengths 600 m and 400 m and of diameters 30 cm and 20 cm respectively, is 15 m. Determine the rate of flow of water if the co-efficient of friction is 0.005 for both the pipes. Neglect minor losses.

29. Water is flowing vertically downwards through a 10 cm diameter pipe at the rate of 50 l.p.s. At a particular location the pipe suddenly enlarges to 20 cm diameter. A point P is located 50 cm above the section of enlargement and another point Q is located 50 cm below it in the enlarged portion. A pressure gauge connected at P gives a reading of 19.62 N/cm^2. Calculate the pressure at location Q neglecting friction loss between P and Q but considering the loss due to sudden enlargement. What

will be the pressure at Q if the same discharge flows upwards assuming that the pressure P remains the same ? Consider the loss due to contraction with $C_c = 0.60$ but neglect friction loss between P and Q. [**Ans.** 21.36 N/cm^2, 23.4 N/cm^2]

30. Two tanks are connected with the help of two pipes in series. The lengths of the pipes are 1000 m and 800 m whereas the diameters are 400 mm and 200 mm respectively. The co-efficient of friction for both the pipes is 0.008. The difference of water level in the two tanks is 15 m. Find the rate of flow of water through the pipes, considering all losses. Also draw the total energy line and hydraulic gradient lines for the system. [**Ans.** 0.0464 m^3/s]

[**Hint .** $L_1 = 1000$ m ; $L_2 = 800$ m ; $d_1 = 400$ mm $= 0.4$ m ; $d_2 = 200$ mm $= 0.2$ m, $f = 0.008$; $H = 15$ m.

Now $$H = h_i + hf_1 + h_c + hf_2 + h_o,$$

$$H = \frac{0.5\,V_1^2}{2g} + \frac{4f \times L_1 \times V_1^2}{d \times 2g} + \frac{0.5\,V_2^2}{2g} + \frac{4f \times L_2 \times V_2^2}{d_2 \times 2g} + \frac{V_2^2}{2g}$$

or $$15 = \frac{0.5\,V_1^2}{2 \times 9.81} + \frac{4 \times 0.008 \times 1000 \times V_1^2}{0.4 \times 2 \times 9.81} + \frac{0.5\,V_2^2}{2 \times 9.81} + \frac{4 \times 0.008 \times 800 \times V_2^2}{0.2 \times 2 \times 9.81} + \frac{V_2^2}{2g}$$

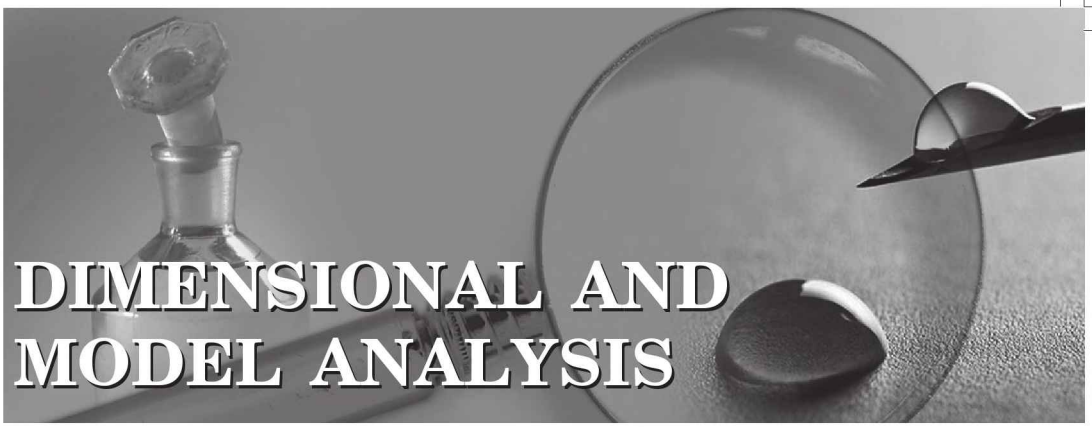
Also $$A_1V_1 = A_2V_2 \text{ or } V_2 = 4V_1$$

$\therefore$ $$15 = 0.02548V_1^2 + 4.0775V_1^2 + 0.02548V_2^2 + 6.524V_2^2 + 0.05097V_2^2$$
$$= 4.103V_1^2 + 6.6V_2^2 = 4.103V_1^2 + 6.6 \times (4V_1)^2$$
$$= 4.103V_1^2 + 105.607V_1^2 = 109.71V_1^2$$

$\therefore$ $$V_1 = \sqrt{\frac{15}{109.71}} = 0.3697 \text{ m/s} \therefore Q = A_1V_1 = \frac{\pi}{4}(.4)^2 \times 0.3697 = 0.0464 \text{ m}^3\text{/s}]$$

31. A pipe of diameter 25 cm and length 2000 m connects two reservoirs, having difference of water level 25 m. Determine the discharge through the pipe. If an additional pipe of diameter 25 cm and length 1000 m is attached to the last 1000 m length of the existing pipe, find the increase in discharge. Take $f = 0.015$. Neglect minor losses. (*Delhi University, December 2002*) [**Ans.** (*i*) 49.62 l/s, (*ii*) 13.14 l/s]

12 CHAPTER



DIMENSIONAL AND MODEL ANALYSIS

▶ 12.1 INTRODUCTION

Dimensional analysis is a method of dimensions. It is a mathematical technique used in research work for design and for conducting model tests. It deals with the dimensions of the physical quantities involved in the phenomenon. All physical quantities are measured by comparison, which is made with respect to an arbitrarily fixed value. Length L, mass M and time T are three fixed dimensions which are of importance in Fluid Mechanics. If in any problem of fluid mechanics, heat is involved then temperature is also taken as fixed dimension. These fixed dimensions are called fundamental dimensions or fundamental quantity.

▶ 12.2 SECONDARY OR DERIVED QUANTITIES

Secondary or derived quantities are those quantities which possess more than one fundamental dimension. For example, velocity is denoted by distance per unit time (L/T), density by mass per unit volume $\left(\frac{M}{L^3}\right)$ and acceleration by distance per second square (L/T^2). Then velocity, density and acceleration become as secondary or derived quantities. The expressions (L/T), $\left(\frac{M}{L^3}\right)$ and $\left(\frac{L}{T^2}\right)$ are called the dimensions of velocity, density and acceleration respectively. The dimensions of mostly used physical quantities in Fluid Mechanics are given in Table 12.1.

Table 12.1

S. No.	*Physical Quantity*	*Symbol*	*Dimensions*
	(*a*) **Fundamental**		
1.	Length	L	L
2.	Mass	M	M
3.	Time	T	T

Contd...

S.No.	Physical Quantity	Symbol	Dimensions
	(*b*) **Geometric**		
4.	Area	A	L^2
5.	Volume	$\forall$	L^3
	(*c*) **Kinematic Quantities**		
6.	Velocity	v	LT^{-1}
7.	Angular Velocity	ω	T^{-1}
8.	Acceleration	a	LT^{-2}
9.	Angular Acceleration	α	T^{-2}
10.	Discharge	Q	L^3T^{-1}
11.	Acceleration due to Gravity	g	LT^{-2}
12.	Kinematic Viscosity	ν	L^2T^{-1}
	(*d*) **Dynamic Quantities**		
13.	Force	F	MLT^{-2}
14.	Weight	W	MLT^{-2}
15.	Density	ρ	ML^{-3}
16.	Specific Weight	w	$ML^{-2}T^{-2}$
17.	Dynamic Viscosity	μ	$ML^{-1}T^{-1}$
18.	Pressure Intensity	p	$ML^{-1}T^{-2}$
19.	Modulus of Elasticity	$\begin{cases} K \\ E \end{cases}$	$ML^{-1}T^{-2}$
20.	Surface Tension	σ	MT^{-2}
21.	Shear Stress	τ	$ML^{-1}T^{-2}$
22.	Work, Energy	W or E	ML^2T^{-2}
23.	Power	P	ML^2T^{-3}
24.	Torque	T	ML^2T^{-2}
25.	Momentum	M	MLT^{-1}

Problem 12.1 *Determine the dimensions of the quantities given below : (i) Angular velocity, (ii) Angular acceleration, (iii) Discharge, (iv) Kinematic viscosity, (v) Force, (vi) Specific weight, and (vii) Dynamic viscosity.*

Solution. (*i*) Angular velocity $= \dfrac{\text{Angle covered in radians}}{\text{Time}} = \dfrac{1}{T} = T^{-1}$.

(*ii*) Angular acceleration $= \text{rad/sec}^2 = \dfrac{\text{rad}}{T^2} = \dfrac{1}{T^2} = T^{-2}$.

(*iii*) Discharge $= \text{Area} \times \text{Velocity} = L^2 \times \dfrac{L}{T} = \dfrac{L^3}{T} = L^3T^{-1}$.

(*iv*) Kinematic viscosity (ν) $= \dfrac{\mu}{\rho}$, where μ is given by $\tau = \mu \dfrac{\partial u}{\partial y}$

$\therefore$ $$\mu = \frac{\tau}{\dfrac{\partial u}{\partial y}} = \frac{\text{Shear Stress}}{\dfrac{L}{T} \times \dfrac{1}{L}} = \frac{\text{Force}}{\dfrac{\text{Area}}{\dfrac{1}{T}}}$$

$$= \frac{\text{Mass} \times \text{Acceleration}}{\text{Area} \times \text{Time}} = \frac{M \times \frac{L}{T^2}}{L^2 \times \frac{1}{T}} = \frac{ML}{L^2T^2 \times \frac{1}{T}} = \frac{M}{LT} = ML^{-1}T^{-1}$$

and $$\rho = \frac{\text{Mass}}{\text{Volume}} = \frac{M}{L^3} = ML^{-3}$$

$\therefore$ Kinematic viscosity $(\nu) = \frac{\mu}{\rho} = \frac{ML^{-1}T^{-1}}{ML^{-3}} = L^2T^{-1}$.

(*v*) Force $= \text{Mass} \times \text{Acceleration} = M \times \frac{\text{Length}}{(\text{Time})^2} = \frac{ML}{T^2} = MLT^{-2}$.

(*vi*) Specific weight $= \frac{\text{Weight}}{\text{Volume}} = \frac{\text{Force}}{\text{Volume}} = \frac{MLT^{-2}}{L^3} = ML^{-2}T^{-2}$.

(*vii*) Dynamic viscosity, μ is derived in (*iv*) as $\mu = ML^{-1}T^{-1}$.

▶ 12.3 DIMENSIONAL HOMOGENEITY

Dimensional homogeneity means the dimensions of each terms in an equation on both sides are equal. Thus if the dimensions of each term on both sides of an equation are the same the equation is known as dimensionally homogeneous equation. The powers of fundamental dimensions (*i.e.*, *L*, *M*, *T*) on both sides of the equation will be identical for a dimensionally homogeneous equation. Such equations are independent of the system of units.

Let us consider the equation, $V = \sqrt{2gH}$

Dimension of L.H.S. $= V = \frac{L}{T} = LT^{-1}$

Dimension of R.H.S. $= \sqrt{2gH} = \sqrt{\frac{L}{T^2} \times L} = \sqrt{\frac{L^2}{T^2}} = \frac{L}{T} = LT^{-1}$

Dimension of L.H.S. = Dimension of R.H.S. $= LT^{-1}$

$\therefore$ Equation $V = \sqrt{2gH}$ is dimensionally homogeneous. So it can be used in any system of units.

▶ 12.4 METHODS OF DIMENSIONAL ANALYSIS

If the number of variable involved in a physical phenomenon are known, then the relation among the variables can be determined by the following two methods :

1. Rayleigh's method, and
2. Buckingham's π-theorem.

12.4.1 Rayleigh's Method. This method is used for determining the expression for a variable which depends upon maximum three or four variables only. If the number of independent variables becomes more than four, then it is very difficult to find the expression for the dependent variable.

Let X is a variable, which depends on X_1, X_2 and X_3 variables. Then according to Rayleigh's method, X is function of X_1, X_2 and X_3 and mathematically it is written as $X = f[X_1, X_2, X_3]$.

This can also be written as $X = KX_1^{\,a} \cdot X_2^{\,b} \cdot X_3^{\,c}$

where K is constant and a, b and c are arbitrary powers.

The values of a, b and c are obtained by comparing the powers of the fundamental dimension on both sides. Thus the expression is obtained for dependent variable.

Problem 12.2 *The time period (t) of a pendulum depends upon the length (L) of the pendulum and acceleration due to gravity (g). Derive an expression for the time period.*

Solution. Time period t is a function of (*i*) L and (*ii*) g

$\therefore$ $t = KL^a \cdot g^b$, where K is a constant ...(*i*)

Substituting the dimensions on both sides $T^1 = KL^a \cdot (LT^{-2})^b$

Equating the powers of M, L and T on both sides, we have

Power of T, $1 = -2b$ $\therefore$ $b = -\dfrac{1}{2}$

Power of L, $0 = a + b$ $\therefore$ $a = -b = -\left(-\dfrac{1}{2}\right) = \dfrac{1}{2}$

Substituting the values of a and b in equation (*i*),

$$t = KL^{1/2} \cdot g^{-1/2} = K\sqrt{\frac{L}{g}}$$

The value of K is determined from experiments which is given as

$$K = 2\pi$$

$\therefore$ $$t = \mathbf{2\pi\sqrt{\frac{L}{g}}}\textbf{. Ans.}$$

Problem 12.3 *Find an expression for the drag force on smooth sphere of diameter D, moving with a uniform velocity V in a fluid of density ρ and dynamic viscosity μ.*

Solution. Drag force F is a function of

(*i*) Diameter, D (*ii*) Velocity, V (*iii*) Density, ρ

(*iv*) Viscocity, μ

$\therefore$ $F = KD^a \cdot V^b \cdot \rho^c \cdot \mu^d$ ...(*i*)

where K is non-dimensional factor.

Substituting the dimensions on both sides,

$$MLT^{-2} = KL^a \cdot (LT^{-1})^b \cdot (ML^{-3})^c \cdot (ML^{-1}T^{-1})^d$$

Equating the powers of M, L and T on both sides,

Power of M, $1 = c + d$

Power of L, $1 = a + b - 3c - d$

Power of T, $-2 = -b - d$.

There are four unknowns (a, b, c, d) but equations are three. Hence it is not possible to find the values of a, b, c and d. But three of them can be expressed in terms of fourth variable which is most important. Here viscosity is having a vital role and hence a, b, c are expressed in terms of d which is the power to viscosity.

$\therefore$ $c = 1 - d$

$b = 2 - d$

$a = 1 - b + 3c + d = 1 - 2 + d + 3(1 - d) + d$

$$= 1 - 2 + d + 3 - 3d + d = 2 - d$$

Substituting these values of a, b and c in (i), we get

$$F = KD^{2-d} \cdot V^{2-d} \cdot \rho^{1-d} \cdot \mu^d$$

$$= KD^2V^2\rho\,(D^{-d} \cdot V^{-d} \cdot \rho^{-d} \cdot \mu^d) = K\rho D^2V^2 \left(\frac{\mu}{\rho VD}\right)^d$$

$$= \mathbf{K\rho D^2V^2\phi\left(\frac{\mu}{\rho VD}\right)}\ \textbf{. Ans.}$$

Problem 12.4 *Find the expression for the power P, developed by a pump when P depends upon the head H, the discharge Q and specific weight w of the fluid.*

Solution. Power P is a function of

(i) Head, H (ii) Discharge, Q

(iii) Specific weight, w

$\therefore$ $$P = KH^a \cdot Q^b \cdot w^c \qquad ...(i)$$

where K = Non-dimensional constant.

Substituting the dimensions on both sides of equation (i)

$$ML^2T^{-3} = KL^a \cdot (L^3T^{-1})^b \cdot (ML^{-2}T^{-2})^c$$

Equating the powers of M, L and T on both sides,

Power of M, $1 = c$, $\therefore$ $c = 1$

Power of L, $2 = a + 3b - 2c$, $a = 2 - 3b + 2c = 2 - 3 + 2 = 1$

Power of T, $-3 = -b - 2c$ $\therefore$ $b = 3 - 2c = 3 - 2 = 1$

Substituting the values of a, b and c in (i)

$$P = KH^1 \cdot Q^1 \cdot w^1 = \mathbf{KHQw.\ Ans.}$$

Problem 12.5 *The efficiency η of a fan depends on the density ρ, the dynamic viscosity μ of the fluid, the angular velocity ω, diameter D of the rotor and the discharge Q. Express η in terms of dimensionless parameters.*

Solution. The efficiency η depends on

(i) density, ρ (ii) viscosity, μ

(iii) angular velocity, ω (iv) diameter, D

(v) discharge, Q

$\therefore$ $$\eta = K\rho^a \cdot \mu^b \cdot \omega^c \cdot D^d \cdot Q^e \qquad ...(i)$$

where K = Non-dimensional constant.

Substituting the dimensions on both sides of equation (i)

$$M^0L^0T^0 = K\,(ML^{-3})^a \cdot (ML^{-1}T^{-1})^b \cdot (T^{-1})^c \cdot (L)^d \cdot (L^3T^{-1})^e$$

Equating powers of M, L, T on both sides,

Power of M, $0 = a + b$

Power of L, $0 = -3a - b + d + 3e$

Power of T, $0 = -b - c - e$.

There are five unknowns but equations are three. Express the three unknowns in terms of the other two unknowns which are more important. Viscosity and discharge are more important in this problem. Hence expressing a, c and d in terms of b and e, we get

$$a = -b$$

$$c = -(b + e)$$

$$d = +3a + b - 3e = -3b + b - 3e = -2b - 3e.$$

Substituting these values in equation (*i*), we get

$$\eta = K\rho^{-b} . \mu^b . \omega^{-(b+e)} . D^{-2b-3e} . Q^e$$
$$= K\rho^{-b} . \mu^b . \omega^{-b} . \omega^{-e} . D^{-2b} . D^{-3e} . Q^e$$
$$= \mathbf{K}\left(\frac{\mu}{\rho\omega D^2}\right)^{b} \cdot \left(\frac{Q}{\omega D^3}\right)^{e} = \phi\left[\left(\frac{\mu}{\rho\omega D^2}\right) \cdot \left(\frac{Q}{\omega D^3}\right)\right]. \textbf{ Ans.}$$

Problem 12.6 *The resisting force R of a supersonic plane during flight can be considered as dependent upon the length of the aircraft l, velocity V, air viscosity μ, air density ρ and bulk modulus of air K. Express the functional relationship between these variables and the resisting force.*

Solution. The resisting force R depends upon

(*i*) density, l, (*ii*) velocity, V,
(*iii*) viscosity, μ, (*iv*) density, ρ,
(*v*) Bulk modulus, K.

$$\therefore \qquad R = Al^a . V^b . \mu^c . \rho^d . K^e \qquad ...(i)$$

where A is the non-dimensional constant.

Substituting the dimensions on both sides of the equation (*i*),

$$MLT^{-2} = AL^a . (LT^{-1})^b . (ML^{-1}T^{-1})^c . (ML^{-3})^d . (ML^{-1}T^{-2})^e$$

Equating the powers of M, L, T on both sides,

Power of M, $1 = c + d + e$
Power of L, $1 = a + b - c - 3d - e$
Power of T, $-2 = -b - c - 2e$.

There are five unknowns but equations are only three. Expressing the three unknowns in terms of two unknowns (μ and K).

$\therefore$ Express the values of a, b and d in terms of c and e.

Solving,
$$d = 1 - c - e$$
$$b = 2 - c - 2e$$
$$a = 1 - b + c + 3d + e = 1 - (2 - c - 2e) + c + 3(1 - c - e) + e$$
$$= 1 - 2 + c + 2e + c + 3 - 3c - 3e + e = 2 - c.$$

Substituting these values in (*i*), we get

$$R = A\, l^{2-c} . V^{2-c-2e} . \mu^c . \rho^{1-c-e} . K^e$$
$$= A\, l^2 . V^2 . \rho(l^{-c} V^{-c} \mu^c \rho^{-c}) . (V^{-2e} . \rho^{-e} . K^e)$$
$$= A\, l^2V^2\rho \left(\frac{\mu}{\rho VL}\right)^{c} \cdot \left(\frac{K}{\rho V^2}\right)^{e}$$
$$= \mathbf{A\rho l^2V^2}\ \phi\left[\left(\frac{\mu}{\rho \mathbf{VL}}\right) \cdot \left(\frac{\mathbf{K}}{\rho \mathbf{V^2}}\right)\right] \textbf{ . Ans.}$$

Problem 12.7 *A partially sub-merged body is towed in water. The resistance R to its motion depends on the density ρ, the viscosity μ of water, length l of the body, velocity v of the body and the acceleration due to gravity g. Show that the resistance to the motion can be expressed in the form*

$$R = \rho L^2V^2\ \phi\left[\left(\frac{\mu}{\rho VL}\right) \cdot \left(\frac{lg}{V^2}\right)\right].$$

Solution. The resistance R depends on

(*i*) density, ρ, (*ii*) viscosity, μ,
(*iii*) length, l, (*iv*) velocity, V,
(*v*) acceleration, g.

$$\therefore \quad R = K\rho^a \cdot \mu^b \cdot l^c \cdot V^d \cdot g^e \qquad ...(i)$$

where K = Non-dimensional constant.

Substituting the dimensions on both sides of the equation (*i*),

$$MLT^{-2} = K(ML^{-3})^a \cdot (ML^{-1}T^{-1})^b \cdot L^c \cdot (LT^{-1})^d \cdot (LT^{-2})^e$$

Equating the powers of M, L, T on both sides

Power of M, $1 = a + b$

Power of L, $1 = -3a - b + c + d + e$

Power of T, $-2 = -b - d - 2e$.

There are five unknowns and equations are only three. Expressing the three unknowns in terms of two unknowns (μ and g). Hence express a, c and d in terms of b and e. Solving, we get

$$a = 1 - b$$
$$d = 2 - b - 2e$$
$$c = 1 + 3a + b - d - e = 1 + 3(1 - b) + b - (2 - b - 2e) - e$$
$$= 1 + 3 - 3b + b - 2 + b + 2e - e = 2 - b + e.$$

Substituting these values in equation (*i*), we get

$$R = K\rho^{1-b} \cdot \mu^b \cdot l^{2-b+e} \cdot V^{2-b-2e} \cdot g^e$$
$$= K\rho l^2 \cdot V^2 \cdot (\rho^{-b} \mu^b l^{-b} V^{-b}) \cdot (l^e \cdot V^{-2e} \cdot g^e)$$
$$= K\rho l^2 V^2 \cdot \left(\frac{\mu}{\rho V l}\right)^b \cdot \left(\frac{lg}{V^2}\right)^e = \mathbf{K\rho l^2 V^2 \phi}\left[\left(\frac{\mu}{\rho \mathbf{V l}}\right) \cdot \left(\frac{\mathbf{lg}}{\mathbf{V^2}}\right)\right]. \quad \textbf{Ans.}$$

12.4.2 Buckingham's π-Theorem. The Rayleigh's method of dimensional analysis becomes more laborious if the variables are more than the number of fundamental dimensions (M, L, T). This difficulty is overcame by using Buckingham's π-theorem, which states, "If there are n variables (independent and dependent variables) in a physical phenomenon and if these variables contain m fundamental dimensions (M, L, T), then the variables are arranged into $(n - m)$ dimensionless terms. Each term is called π-term".

Let X_1, X_2, X_3, ..., X_n are the variables involved in a physical problem. Let X_1 be the dependent variable and X_2, X_3, ..., X_n are the independent variables on which X_1 depends. Then X_1 is a function of X_2, X_3, ..., X_n and mathematically it is expressed as

$$X_1 = f(X_2, X_3, ..., X_n) \qquad ...(12.1)$$

Equation (12.1) can also be written as

$$f_1(X_1, X_2, X_3, ..., X_n) = 0. \qquad ...(12.2)$$

Equation (12.2) is a dimensionally homogeneous equation. It contains n variables. If there are m fundamental dimensions then according to Buckingham's π-theorem, equation (12.2) can be written in terms of number of dimensionless groups or π-terms in which number of π-terms is equal to $(n - m)$. Hence equation (12.2) becomes as

$$f(\pi_1, \pi_2, ..., \pi_{n-m}) = 0. \qquad ...(12.3)$$

Each of π-terms is dimensionless and is independent of the system. Division or multiplication by a constant does not change the character of the π-term. Each π-term contains $m + 1$ variables, where m is the number of fundamental dimensions and is also called repeating variables. Let in the above case X_2, X_3 and X_4 are repeating variables if the fundamental dimension m (M, L, T) = 3. Then each π-term is written as

$$\left.\begin{aligned} \pi_1 &= X_2^{a_1} \cdot X_3^{b_1} \cdot X_4^{c_1} \cdot X_1 \\ \pi_2 &= X_2^{a_2} \cdot X_3^{b_2} \cdot X_4^{c_2} \cdot X_5 \\ &\vdots \\ \pi_{n-m} &= X_2^{a_{n-m}} \cdot X_3^{b_{n-m}} \cdot X_4^{c_{n-m}} \cdot X_n \end{aligned}\right\} \quad ...(12.4)$$

Each equation is solved by the principle of dimensional homogeneity and values of a_1, b_1, c_1 etc., are obtained. These values are substituted in equation (12.4) and values of $\pi_1, \pi_2, ..., \pi_{n-m}$ are obtained. These values of π's are substituted in equation (12.3). The final equation for the phenomenon is obtained by expressing any one of the π-terms as a function of others as

$$\pi_1 = \phi\ [\pi_2, \pi_3, ..., \pi_{n-m}]$$

or $$\pi_2 = \phi_1\ [\pi_1, \pi_3, ..., \pi_{n-m}] \quad ...(12.5)$$

12.4.3 Method of Selecting Repeating Variables. The number of repeating variables are equal to the number of fundamental dimensions of the problem. The choice of repeating variables is governed by the following considerations :

1. As far as possible, the dependent variable should not be selected as repeating variable.
2. The repeating variables should be choosen in such a way that one variable contains geometric property, other variable contains flow property and third variable contains fluid property.

Variables with Geometric Property are

(*i*) Length, l (*ii*) d (*iii*) Height, H etc.

Variables with flow property are

(*i*) Velocity, V (*ii*) Acceleration etc.

Variables with fluid property : (*i*) μ, (*ii*) ρ, (*iii*) ω etc.

3. The repeating variables selected should not form a dimensionless group.
4. The repeating variables together must have the same number of fundamental dimensions.
5. No two repeating variables should have the same dimensions.

In most of fluid mechanics problems, the choice of repeating variables may be (*i*) d, v, ρ (*ii*) l, v, ρ or (*iii*) l, v, μ or (*iv*) d, v, μ.

12.4.4 Procedure for Solving Problems by Buckingham's π-theorem. The procedure for solving problems by Buckingham's π-theorem is explained by considering the problem 12.6 which is also solved by the Rayleigh's method. The problem is :

The resisting force R of a supersonic plane during flight can be considered as dependent upon the length of the aircraft l, velocity V, air viscosity μ, air density ρ and bulk modulus of air K. Express the functional relationship between these variables and the resisting force.

Solution. Step 1. The resisting force R depends upon (*i*) l, (*ii*) V, (*iii*) μ, (*iv*) ρ and (*v*) K. Hence R is a function of l, V, μ, ρ and K. Mathematically,

$$R = f(l, V, \mu, \rho, K) \quad ...(i)$$

or it can be written as $f_1\ (R, l, V, \mu, \rho, K) = 0$ ...(*ii*)

$\therefore$ Total number of variables, $n = 6$.

Number of fundamental dimensions, $m = 3$.

[m is obtained by writing dimensions of each variables as $R = MLT^{-2}$, $V = LT^{-1}$, $\mu = ML^{-1}T^{-1}$, $\rho = ML^{-3}$, $K = ML^{-1}T^{-2}$. Thus as fundamental dimensions in the problem are M, L, T and hence $m = 3$.]

Number of dimensionless π-terms = $n - m = 6 - 3 = 3$.

Thus three π-terms say π_1, π_2 and π_3 are formed. Hence equation (*ii*) is written as

$$f_1\ (\pi_1, \pi_2, \pi_3) = 0. \quad ...(iii)$$

Step 2. Each π term = $m + 1$ variables, where m is equal to 3 and also called repeating variables. Out of six variables R, l, V, μ, ρ and K, three variables are to be selected as repeating variable. R is a dependent variable and should not be selected as a repeating variable. Out of the five remaining

variables, one variable should have geometric property, the second variable should have flow property and third one fluid property. These requirements are fulfilled by selecting l, V and ρ as repeating variables. The repeating variables themselves should not form a dimensionless term and should have themselves fundamental dimensions equal to m, *i.e.*, 3 here. Dimensions of l, V and ρ are L, LT^{-1}, ML^{-3} and hence the three fundamental dimensions exist in l, V and ρ and they themselves do not form dimensionless group.

Step 3. Each π-term is written as according to equation (12.4)

$$\left.\begin{aligned}\pi_1 &= l^{a_1} \cdot V^{b_1} \cdot \rho^{c_1} \cdot R \\ \pi_2 &= l^{a_2} \cdot V^{b_2} \cdot \rho^{c_2} \cdot \mu \\ \pi_3 &= l^{a_3} \cdot V^{b_3} \cdot \rho^{c_3} \cdot K\end{aligned}\right\} \quad ...(iv)$$

Step 4. Each π-term is solved by the principle of dimensional homogeneity. For the first π-term, we have

$$\pi_1 = M^0L^0T^0 = L^{a_1} \cdot (LT^{-1})^{b_1} \cdot (ML^{-3})^{c_1} \cdot MLT^{-2}.$$

Equating the powers of M, L, T on both sides, we get

Power of M, $0 = c_1 + 1 \quad \therefore \quad c_1 = -1$

Power of L, $0 = a_1 + b_1 - 3c_1 + 1,$

$\therefore$ $a_1 = -b_1 + 3c_1 - 1 = 2 - 3 - 1 = -2$

Power of T, $0 = -b_1 - 2 \quad \therefore \quad b_1 = -2$

Substituting the values of a_1, b_1 and c_1 in equation (iv),

$$\pi_1 = l^{-2} \cdot V^{-2} \cdot \rho^{-1} \cdot R$$

or $$\pi_1 = \frac{R}{l^2V^2\rho} = \frac{R}{\rho l^2V^2} \quad ...(v)$$

Similarly for the 2nd π-term, we get $\pi_2 = M^0L^0T^0 = L^{a_2} \cdot (LT^{-1})^{b_2} \cdot (ML^{-3})^{c_2} \cdot ML^{-1}T^{-1}$.

Equating the powers of M, L, T on both sides

Power of M, $0 = c_2 + 1, \quad \therefore \quad c_2 = -1$

Power of L, $0 = a_2 + b_2 - 3c_2 - 1,$

$a_2 = -b_2 + 3c_2 + 1 = 1 - 3 + 1 = -1$

Power of T, $0 = -b_2 - 1, \quad \therefore \quad b_2 = -1$

Substituting the values of a_2, b_2 and c_2 in π_2 of (iv)

$$\pi_2 = l^{-1} \cdot V^{-1} \cdot \rho^{-1} \cdot \mu = \frac{\mu}{lV\rho}.$$

3rd π-term

$$\pi_3 = l^{a_3} \cdot V^{b_3} \cdot \rho^{c_3} \cdot K$$

or $$M^0L^0T^0 = L^{a_3} \cdot (LT^{-3})^{b_3} \cdot (ML^{-3})^{a_3} \cdot ML^{-1}T^{-2}$$

Equating the powers of M, L, T on both sides, we have

Power of M, $0 = c_3 + 1, \quad \therefore \quad c_3 = -1$

Power of L, $0 = a_3 + b_3 - 3c_3 - 1, \quad \therefore \quad a_3 = -b_3 + 3c_3 + 1 = 2 - 3 + 1 = 0$

Power of T, $0 = -b_3 - 2, \quad \therefore \quad b_3 = -2$

Substituting the values of a_3, b_3 and c_3 in π_3 term

$$\pi_3 = l^0 \cdot V^{-2} \cdot \rho^{-1} \cdot K = \frac{K}{V^2\rho}.$$

Step 5. Substituting the values of π_1, π_2 and π_3 in equation (*iii*), we get

$$f_1\left(\frac{R}{\rho l^2V^2}, \frac{\mu}{lV\rho}, \frac{K}{V^2\rho}\right) = 0 \quad \text{or} \quad \frac{R}{\rho l^2V^2} = \phi\left[\frac{\mu}{lV\rho}, \frac{K}{V^2\rho}\right]$$

or
$$R = \rho \mathbf{l^2V^2}\phi\left[\frac{\mu}{\mathbf{lV\rho}}, \frac{\mathbf{K}}{\mathbf{V^2\rho}}\right]. \textbf{ Ans.}$$

Problem 12.8 *(a) State Buckingham's π-theorem.*
(b) The efficiency η of a fan depends on density ρ, dynamic viscosity μ of the fluid, angular velocity ω, diameter D of the rotor and the discharge Q. Express η in terms of dimensionless parameters.

Solution. (*a*) Statement of Buckingham's π-theorem is given in Article 12.4.2.

(*b*) Given : η is a function of ρ, μ, ω, D and Q

$$\therefore \qquad \eta = f(\rho, \mu, \omega, D, Q) \quad \text{or} \quad f_1(\eta, \rho, \mu, \omega, D, Q) = 0 \qquad ...(i)$$

Hence total number of variables, $n = 6$.

The value of m, *i.e.*, number of fundamental dimensions for the problem is obtained by writing dimensions of each variable. Dimensions of each variable are

$$\eta = \text{Dimensionless}, \rho = ML^{-3}, \mu = ML^{-1}T^{-1}, \omega = T^{-1}, D = L \text{ and } Q = L^3T^{-1}$$

$\therefore \quad m = 3$

Number of π-terms $= n - m = 6 - 3 = 3$

Equation (*i*) is written as $f_1(\pi_1, \pi_2, \pi_3) = 0$ ...(*ii*)

Each π-term contains $m + 1$ variables, where m is equal to three and is also repeating variable.

Choosing D, ω and ρ as repeating variables, we have

$$\pi_1 = D^{a_1} \cdot \omega^{b_1} \cdot \rho^{c_1} \cdot \eta$$
$$\pi_2 = D^{a_2} \cdot \omega^{b_2} \cdot \rho^{c_2} \cdot \mu$$
$$\pi_3 = D^{a_3} \cdot \omega^{b_3} \cdot \rho^{c_3} \cdot Q$$

First π-term $\pi_1 = D^{a_1} \cdot \omega^{b_1} \cdot \rho^{c_1} \cdot \eta$

Substituting dimensions on both sides of π_1,

$$M^0L^0T^0 = L^{a_1} \cdot (T^{-1})^{b_1} \cdot (ML^{-3})^{c_1} \cdot M^0L^0T^0$$

Equating the powers of M, L, T on both sides

Power of M, $\quad 0 = c_1 + 0, \quad \therefore \quad c_1 = 0$

Power of L, $\quad 0 = a_1 + 0, \quad \therefore \quad a_1 = 0$

Power of T, $\quad 0 = -b_1 + 0, \quad \therefore \quad b_1 = 0$

Substituting the values of a_1, b_1 and c_1 in π_1, we get

$$\pi_1 = D^0\omega^0\rho^0 \cdot \eta = \eta$$

[If a variable is dimensionless, it itself is a π-term. Here the variable η is a dimensionless and hence η is a π-term. As it exists in first π-term and hence $\pi_1 = \eta$. Then there is no need of equating the powers. Directly the value can be obtained.]

Second π-term $\pi_2 = D^{a_2} \cdot \omega^{b_2} \cdot \rho^{c_2} \cdot \mu$

Substituting the dimensions on both sides

$$M^0L^0T^0 = L^{a_2} \cdot (T^{-1})^{b_2} \cdot (ML^{-3})^{c_2} \cdot ML^{-1}T^{-1}$$

Equating the powers of M, L, T on both sides

Power of M, $0 = c_2 + 1$, $\therefore$ $c_2 = -1$

Power of L, $0 = a_2 - 3c_2 - 1$, $\therefore$ $a_2 = 3c_2 + 1 = -3 + 1 = -2$

Power of T, $0 = -b_2 - 1$, $\therefore$ $b_2 = -1$

Substituting the values of a_2, b_2 and c_2 in π_2,

$$\pi_2 = D^{-2} \cdot \omega^{-1} \cdot \rho^{-1} \cdot \mu = \frac{\mu}{D^2 \omega \rho}$$

Third π-term $\pi_3 = D^{a_3} \cdot \omega^{b_3} \cdot \rho^{c_3} \cdot Q$

Substituting the dimensions on both sides

$$M^0 L^0 T^0 = L^{a_3} \cdot (T^{-1})^{b_3} \cdot (ML^{-3})^{c_3} \cdot L^3 T^{-1}$$

Equating the powers of M, L and T on both sides

Power of M, $0 = c_3$, $\therefore$ $c_3 = 0$

Power of L, $0 = a_3 - 3c_3 + 3$, $\therefore$ $a_3 = 3c_3 - 3 = -3$

Power of T, $0 = -b_{3-1}$, $\therefore$ $b_3 = -1$

Substituting the values of a_3, b_3 and c_3 in π_3,

$$\pi_3 = D^{-3} \cdot \omega^{-1} \cdot \rho^0 \cdot Q = \frac{Q}{D^2 \omega}$$

Substituting the values of π_1, π_2 and π_3 in equation (*ii*)

$$f_1\left(\eta, \frac{\mu}{D^2\omega\rho}, \frac{Q}{D^2\omega}\right) = 0 \text{ or } \eta = \phi\left[\frac{\mu}{\mathbf{D^2\omega\rho}}, \frac{\mathbf{Q}}{\mathbf{D^2\omega}}\right]. \textbf{ Ans.}$$

Problem 12.9 *Using Buckingham's π-theorem, show that the velocity through a circular orifice is given by* $V = \sqrt{2gH}\ \phi\left[\frac{D}{H}, \frac{\mu}{\rho VH}\right]$, *where H is the head causing flow, D is the diameter of the orifice, μ is co-efficient of viscosity, ρ is the mass density and g is the acceleration due to gravity.*

Solution. Given :

V is a function of H, D, μ, ρ and g

$\therefore$ $V = f(H, D, \mu, \rho, g)$ or $f_1(V, H, D, \mu, \rho, g) = 0$

$\therefore$ Total number of variable, $n = 6$ ...(*i*)

Writing dimension of each variable, we have

$$V = LT^{-1},\ H = L,\ D = L,\ \mu = ML^{-1}T^{-1},\ \rho = ML^{-3},\ g = LT^{-2}.$$

Thus number of fundamental dimensions, $m = 3$

$\therefore$ Number of π-terms $= n - m = 6 - 3 = 3$.

Equation (*i*) can be written as $f_1(\pi_1, \pi_2, \pi_3) = 0$ ...(*ii*)

Each π-term contains $m + 1$ variables, where $m = 3$ and is also equal to repeating variables. Here V is a dependent variable and hence should not be selected as repeating variable. Choosing H, g, ρ as repeating variable, we get three π-terms as

$$\pi_1 = H^{a_1} \cdot g^{b_1} \cdot \rho^{c_1} \cdot V$$

$$\pi_2 = H^{a_2} \cdot g^{b_2} \cdot \rho^{c_2} \cdot D$$

$$\pi_3 = H^{a_3} \cdot g^{b_3} \cdot \rho^{c_3} \cdot \mu$$

First π-term $\pi_1 = H^{a_1} \cdot g^{b_1} \cdot \rho^{c_1} \cdot V$

Substituting dimensions on both sides

$$M^0L^0T^0 = L^{a_1} \cdot (LT^{-2})^{b_1} \cdot (MT^{-3})^{c_1} \cdot (LT^{-1})$$

Equating the powers of M, L, T on both sides,

Power of M, $\quad 0 = c, \quad \therefore \quad c_1 = 0$

Power of L, $\quad 0 = a_1 + b_1 - 3c_1 + 1, \quad \therefore \quad a_1 = -b_1 + 3c_1 - 1 = \frac{1}{2} - 1 = -\frac{1}{2}$

Power of T, $\quad 0 = -2b_1 - 1, \quad \therefore \quad b_1 = -\frac{1}{2}$

Substituting the values of a_1, b_1 and c_1 in π_1,

$$\pi_1 = H^{-\frac{1}{2}} \cdot g^{-\frac{1}{2}} \cdot \rho^0 \cdot V = \frac{V}{\sqrt{gH}}.$$

Second π-term $\quad \pi_2 = H^{a_2} \cdot g^{b_2} \cdot \rho^{c_2} \cdot D$

Substituting the dimensions on both sides,

$$M^0L^0T^0 = L^{a_2} \cdot (LT^{-2})^{b_2} \cdot (ML^{-3})^{c_2} \cdot L$$

Equating the powers of M, L, T,

Power of M, $\quad 0 = c_2 \quad \therefore \quad c_2 = 0$

Power of L, $\quad 0 = a_2 + b_2 - 3c_2 + 1, \; a_2 = -b_2 + 3c_2 - 1 = -1$

Power of T, $\quad 0 = -2b_2, \quad \therefore \quad b_2 = 0$

Substituting the values of a_2, b_2, c_2 in π_2,

$$\pi_2 = H^{-1} \cdot g^0\rho^0 \cdot D = \frac{D}{H}.$$

Third π-term $\quad \pi_3 = H^{a_3} \cdot g^{b_3} \cdot \rho^{c_3} \cdot \mu$

Substituting the dimensions on both sides

$$M^0L^0T^0 = L^{a_3} \cdot (LT^{-2})^{b_3} \cdot (ML^{-3})^{c_3} \cdot ML^{-1}T^{-1}$$

Equating the powers of M, L, T on both sides

Power of M, $\quad 0 = c_3 + 1, \quad \therefore \quad c_3 = -1$

Power of L, $\quad 0 = a_3 + b_3 - 3c_3 - 1, \quad \therefore \quad a_3 = -b_3 + 3c_3 + 1 = \frac{1}{2} - 3 + 1 = -\frac{3}{2}$

Power of T, $\quad 0 = -2b_3 - 1, \quad \therefore \quad b_3 = -\frac{1}{2}$

Substituting the values of a_3, b_3 and c_3 in π_3,

$$\pi_3 = H^{-3/2} \cdot g^{-1/2} \cdot \rho^{-1} \cdot \mu = \frac{\mu}{H^{3/2}\rho\sqrt{g}}$$

$$= \frac{\mu}{H\rho\sqrt{gH}} = \frac{\mu V}{H\rho V\sqrt{gH}} \qquad \text{[Multiply and Divide by } V\text{]}$$

$$= \frac{\mu}{H\rho V} \cdot \pi_1 \qquad \left\{\because \quad \frac{V}{\sqrt{gH}} = \pi_1\right\}$$

Substituting the values of π_1, π_2 and π_3 in equation (*ii*),

$$f_1\left(\frac{V}{\sqrt{gH}}, \frac{D}{H}, \pi_1 \frac{\mu}{H\rho V}\right) = 0 \text{ or } \frac{V}{\sqrt{gH}} = \phi\left[\frac{D}{H}, \pi_1 \frac{\mu}{H\rho V}\right]$$

or $$V = \sqrt{2gH}\ \phi\left[\frac{D}{H}, \frac{\mu}{\rho VH}\right]. \textbf{ Ans.}$$

Multiplying by a constant does not change the character of π-terms.

Problem 12.10 *The pressure difference Δp in a pipe of diameter D and length l due to turbulent flow depends on the velocity V, viscosity μ, density ρ and roughness k. Using Buckingham's π-theorem, obtain an expression for Δp.*

Solution. Given :

Δp is a function of D, l, V, μ, ρ, k

$\therefore \quad \Delta p = f(D, l, V, \mu, \rho, k)$ or $f_1(\Delta p, D, l, V, \mu, \rho, k) = 0$ ...(*i*)

$\therefore$ Total number of variables, $n = 7$.

Writing dimensions of each variable,

Dimension of Δp = Dimension of pressure = $ML^{-1}T^{-2}$

$D = L$, $l = L$, $V = LT^{-1}$, $\mu = ML^{-1}T^{-1}$, $\rho = ML^{-3}$, $k = L$

$\therefore$ Number of fundamental dimensions, $m = 3$

Number of π-terms $= n - m = 7 - 3 = 4$.

Now equation (*i*) can be grouped in 4 π-terms as

$$f_1(\pi_1, \pi_2, \pi_3, \pi_4) = 0 \qquad ...(ii)$$

Each π-term contains $m + 1$ or $3 + 1 = 4$ variables. Out of four variables, three are repeating variables. Choosing D, V, ρ as the repeating variables, we have the four π-terms as

$$\pi_1 = D^{a_1} \cdot V^{b_1} \cdot \rho^{c_1} \cdot \Delta p$$

$$\pi_2 = D^{a_2} \cdot V^{b_2} \cdot \rho^{c_2} \cdot l$$

$$\pi_3 = D^{a_3} \cdot V^{b_3} \cdot \rho^{c_3} \cdot \mu$$

$$\pi_4 = D^{a_4} \cdot V^{b_4} \cdot \rho^{c_4} \cdot k$$

First π-term $\pi_1 = D^{a_1} \cdot V^{b_1} \cdot \rho^{c_1} \cdot \Delta p$

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_1} \cdot (LT^{-1})^{b_1} \cdot (ML^{-3})^{c_1} \cdot ML^{-1}T^{-2}$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_1 + 1$ $\quad \therefore \quad c_1 = -1$

Power of L, $0 = a_1 + b_1 - 3c_1 - 1$, $\quad \therefore \quad a_1 = -b_1 + 3c_1 + 1 = 2 - 3 + 1 = 0$

Power of T, $0 = -b_1 - 2$, $\quad \therefore \quad b_1 = -2$

Substituting the values of a_1, b_1 and c_1 in π_1,

$$\pi_1 = D^0 \cdot V^{-2} \cdot \rho^{-1} \cdot \Delta p = \frac{\Delta p}{\rho V^2}$$

Second π-term $\pi_2 = D^{a_2} \cdot V^{b_2} \cdot \rho^{c_2} \cdot l$

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_2} \cdot (LT^{-1})^{b_2} \cdot (ML^{-3})^{c_2} \cdot L$$

Equating powers of M, L, T on both sides,

Power of M, $0 = c_2$, $\quad \therefore \quad c_2 = 0$

Power of L, $0 = a_2 - b_2 - 3c_2 + 1$, $\quad \therefore \quad a_2 = b_2 + 3c_2 - 1 = -1$

Power of T, $0 = -b_2$, $\quad \therefore \quad b_2 = 0$

Substituting the values of a_2, b_2, c_2 in π_2,

$$\pi_2 = D^{-1} \cdot V^0 \cdot \rho^0 \cdot l = \frac{l}{D}$$

Third π-term $\pi_3 = D^{a_3} \cdot V^{b_3} \cdot \rho^{c_3} \cdot \mu$

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_3} \cdot (LT^{-1})^{b_3} \cdot (ML^{-3})^{c_3} \cdot ML^{-1}T^{-1}$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_3 + 1$, $\therefore \quad c_3 = -1$

Power of L, $0 = a_3 + b_3 - 3c_3 - 1$, $\therefore \quad a_3 = -b_3 + 3c_3 + 1 = 1 - 3 + 1 = -1$

Power of T, $0 = -b_3 - 1$, $\therefore \quad b_3 = -1$

Substituting the values of a_3, b_3 and c_3 in π_3,

$$\pi_3 = D^{-1} \cdot V^{-1} \cdot \rho^{-1} \cdot \mu = \frac{\mu}{DV\rho}.$$

Fourth π-term $\pi_4 = D^{a_4} \cdot V^{b_4} \cdot \rho^{c_4} \cdot k$

or $M^0L^0T^0 = L^{a_4} \cdot (LT^{-1})^{b_4} \cdot (ML^{-3})^{c_4} \cdot L$ {Dimension of $k = L$}

Equating the power of M, L, T on both sides,

Power of M, $0 = c_4$, $\therefore \quad c_4 = 0$

Power of L, $0 = a_4 - b_4 - 3c_4 + 1$, $\therefore \quad a_4 = b_4 + 3c_4 - 1 = -1$

Power of T, $0 = -b_4$, $\therefore \quad b_4 = 0$

Substituting the values of a_4, b_4, c_4 in π_4,

$$\pi_4 = D^{-1} \cdot V^0 \cdot \rho^0 \cdot k = \frac{k}{D}$$

Substituting the values of π_1, π_2, π_3 and π_4 in (*ii*),we get

$$f_1\left(\frac{\Delta p}{\rho V^2}, \frac{l}{D}, \frac{\mu}{DV\rho}, \frac{k}{D}\right) = 0 \quad \text{or} \quad \frac{\Delta p}{\rho V^2} = \phi\left[\frac{\boldsymbol{l}}{\mathbf{D}}, \frac{\mu}{\mathbf{DV\rho}}, \frac{\mathbf{k}}{\mathbf{D}}\right]. \textbf{ Ans.}$$

Expression for h_f (Difference of pressure-head). From experiments, it was observed that pressure difference, Δp is a linear function of $\frac{l}{D}$ and hence it is taken out of function

$$\therefore \quad \frac{\Delta p}{\rho V^2} = \frac{l}{D}\phi\left[\frac{\mu}{DV\rho}, \frac{k}{D}\right]$$

$$\therefore \quad \frac{\Delta p}{\rho} = V^2 \cdot \frac{l}{D}\phi\left[\frac{\mu}{DV\rho}, \frac{k}{D}\right]$$

Dividing by g to both sides, we have $\frac{\Delta p}{\rho g} = \frac{V^2 \cdot l}{g \cdot D}\phi\left[\frac{\mu}{DV\rho}, \frac{k}{D}\right]$.

Now $\phi\left[\frac{\mu}{DV\rho}, \frac{k}{D}\right]$ contains two terms. First one is $\frac{\mu}{DV\rho}$ which is $\frac{1}{\text{Reynolds number}}$ or $\frac{1}{R_e}$ and second one is $\frac{k}{D}$ which is called roughness factor. Now $\phi\left[\frac{1}{R_e}, \frac{k}{D}\right]$ is put equal to f, where f is the co-efficient of friction which is a function of Reynolds number and roughness factor.

$$\therefore \quad \frac{\Delta p}{\rho g} = \frac{4f}{2} \cdot \frac{V^2 l}{gD} \qquad \left\{ \because \quad f = \phi\left(\frac{\mu}{DV\rho}, \frac{K}{D}\right) \right\}$$

Multiplying or dividing by any constant does not change the character of π-terms.

$$\therefore \quad \frac{\Delta p}{\rho g} = h_f = \frac{\mathbf{4f \,.\, LV^2}}{\mathbf{D \times 2g}} \,\mathbf{.\ Ans.}$$

Problem 12.11 *The pressure difference Δp in a pipe of diameter D and length l due to viscous flow depends on the velocity V, viscosity μ and density ρ. Using Buckingham's π-theorem, obtain an expression for Δp.*

Solution. This problem is similar to problem 12.10. The only difference is that Δp is to be calculated for viscous flow. Then in the repeating variable instead of ρ, the fluid property μ is to be chosen.

Now Δp is a function of D, l, V, μ, ρ or $\Delta p = f(D, l, V, \mu, \rho)$

or $f_1(\Delta p, D, l, V, \mu, \rho) = 0$ ...(*i*)

Total number of variables, $n = 6$

Number of fundamental dimensions, $m = 3$

Number of π-terms $= n - 3 = 6 - 3 = 3$

Hence equation (*i*) is written as $f_1(\pi_1, \pi_2, \pi_3) = 0$ ...(*ii*)

Each π-term contains $m + 1$ variables, *i.e.*, 3 + 1 = 4 variables. Out of four variables, three are repeating variables.

Choosing D, V, μ as repeating variables, we have π-terms as

$$\pi_1 = D^{a_1} \,.\, V^{b_1} \,.\, \mu^{c_1} \,.\, \Delta p$$

$$\pi_2 = D^{a_2} \,.\, V^{b_2} \,.\, \mu^{c_2} \,.\, l$$

$$\pi_3 = D^{a_3} \,.\, V^{b_3} \,.\, \mu^{c_3} \,.\, \rho$$

First π-term $\pi_1 = D^{a_1} \,.\, V^{b_1} \,.\, \mu^{c_1} \,.\, \Delta p$

Substituting the dimensions on both sides,

$$M^0L^0T^0 = L^{a_1} \,.\, (LT^{-1})^{b_1} \,.\, (ML^{-1}T^{-1})^{c_1} \,.\, ML^{-1}T^{-2}$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_1 + 1$, $\therefore \quad c_1 = -1$

Power of L, $0 = a_1 + b_1 - c_1 - 1$, $\therefore \quad a_1 = -b_1 + c_1 + 1 = 1 - 1 + 1 = 1$

Power of T, $0 = -b_1 - c_1 - 2$, $\therefore \quad b_1 = -c_1 - 2 = 1 - 2 = -1$

Substituting the values of a_1, b_1 and c_1 in π_1,

$$\pi_1 = D^1 \,.\, V^{-1} \,.\, \mu^{-1} \,.\, \Delta p = \frac{D\Delta p}{\mu V}.$$

Second π-term $\pi_2 = D^{a_2} \,.\, V^{b_2} \,.\, \mu^{c_2} \,.\, l$

Substituting the dimensions on both sides,

$$M^0L^0T^0 = L^{a_2} \,.\, (LT^{-1})^{b_2} \,.\, (ML^{-1}T^{-1})^{c_2} \,.\, L.$$

Equating the powers of M, L, T on both sides

Power of M, $0 = c_2$, $\therefore \quad c_2 = 0$

Power of L, $0 = a_2 + b_2 - c_2 + 1$, $\therefore \quad a_2 = -b_2 + c_2 - 1 = -1$

Power of T, $0 = -b_2 - c_2$, $\therefore \quad b_2 = -c_2 = 0$

Substituting the values of a_2, b_2 and c_2 in π_2,

$$\pi_2 = D^{-1} \cdot V^0 \cdot \mu^0 \cdot l = \frac{l}{D}.$$

Third π-term $\pi_3 = D^{a_3} \cdot V^{b_3} \cdot \mu^{c_3} \cdot \rho$

Substituting the dimension on both sides,

$$M^0L^0T^0 = L^{a_3} \cdot (LT^{-1})^{b_3} \cdot (ML^{-1}T^{-1})^{c_3} \cdot ML^{-3}.$$

Equating the powers of M, L, T on both sides

Power of M, $0 = c_3 + 1$, $\therefore \quad c_3 = -1$

Power of L, $0 = a_3 + b_3 - c_3 - 3$, $\therefore \quad a_3 = -b_3 + c_3 + 3 = -1 - 1 + 3 = 1$

Power of T, $0 = -b_3 - c_3$, $\therefore \quad b_3 = -c_3 = -(-1) = 1$

Substituting the values of a_3, b_3 and c_3 in π_3,

$$\pi_3 = D^1 \cdot V^1 \cdot \mu^{-1} \cdot \rho = \frac{\rho DV}{\mu}.$$

Substituting the values of π_1, π_2 and π_3 in equation (*ii*),

$$f_1\left(\frac{D\Delta p}{\mu V}, \frac{l}{D}, \frac{\rho DV}{\mu}\right) = 0 \quad \text{or} \quad \frac{D\Delta p}{\mu V} = \phi\left[\frac{l}{D}, \frac{\rho DV}{\mu}\right] \quad \text{or} \quad \Delta p = \frac{\mu V}{D}\phi\left[\frac{l}{D}, \frac{\rho DV}{\mu}\right]$$

Experiments show that the pressure difference Δp is a linear function $\frac{l}{D}$. Hence $\frac{l}{D}$ can be taken out of the functional as

$$\Delta p = \frac{\mu \mathbf{V}}{\mathbf{D}} \times \frac{\boldsymbol{l}}{\mathbf{D}}\phi\left[\frac{\rho \mathbf{DV}}{\mu}\right]. \textbf{ Ans.}$$

Expression for difference of pressure head for viscous flow

$$h_f = \frac{\Delta p}{\rho g} = \frac{\mu V}{D} \times \frac{l}{D} \times \frac{1}{\rho g}\, \phi\,[R_e] \qquad \left\{\because \quad \frac{\rho DV}{\mu} = R_e\right\}$$

$$= \frac{\mu \mathbf{Vl}}{\mathbf{wD^2}}\phi\,[\mathbf{R_e}]. \textbf{ Ans.}$$

Problem 12.12 *Derive on the basis of dimensional analysis suitable parameters to present the thrust developed by a propeller. Assume that the thrust P depends upon the angular velocity ω, speed of advance V, diameter D, dynamic viscosity μ, mass density ρ, elasticity of the fluid medium which can be denoted by the speed of sound in the medium C.*

Solution. Thrust P is a function of ω, V, D, μ, ρ, C

or $P = f(\omega, V, D, \mu, \rho, C)$

or $f_1 = (P, \omega, V, D, \mu, \rho, C) = 0$ ...(*i*)

$\therefore$ Total number of variables, $n = 7$

Writing dimensions of each variable, we have

$P = MLT^{-2}$, $\omega = T^{-1}$, $V = LT^{-1}$, $D = L$,

$\mu = ML^{-1}T^{-1}$, $\rho = ML^{-3}$, $C = LT^{-1}$

$\therefore$ Number of fundamental dimensions, $m = 3$

$\therefore$ Number of π-terms $= n - m = 7 - 3 = 4$

Hence, equation (*i*) can be written as $f_1(\pi_1, \pi_2, \pi_3, \pi_4) = 0$ ...(*ii*)

Each π-term contains $m + 1$, *i.e.*, $3 + 1 = 4$ variables. Out of four variables, three are repeating variables.

Choosing D, V, ρ as repeating variables, we get π-terms as

$$\pi_1 = D^{a_1} \cdot V^{b_1} \cdot \rho^{c_1} \cdot P$$

$$\pi_2 = D^{a_2} \cdot V^{b_2} \cdot \rho^{c_2} \cdot \omega$$

$$\pi_3 = D^{a_3} \cdot V^{b_3} \cdot \rho^{c_3} \cdot \mu$$

$$\pi_4 = D^{a_4} \cdot V^{b_4} \cdot \rho^{c_4} \cdot C$$

First π-term $\pi_1 = D^{a_1} \cdot V^{b_1} \cdot \rho^{c_1} \cdot P$

Writing dimensions on both sides,

$$M^0L^0T^0 = L^{a_1} \cdot (LT^{-1})^{b_1} \cdot (ML^{-3})^{c_1} \cdot MLT^{-2}.$$

Equating powers of M, L, T on both sides,

Power of M, $0 = c_1 + 1$, $\because \quad c_1 = -1$

Power of L, $0 = a_1 + b_1 - 3c_1 + 1$,

$a_1 = -b_1 + 3c_1 - 1 = 2 - 3 - 1 = -2$

Power of T, $0 = -b_1 - 2$, $\therefore \quad b_1 = -2$

Substituting the values of a_1, b_1 and c_1 in π_1,

$$\pi_1 = D^{-2} \cdot V^{-2} \cdot \rho^{-1} P = \frac{P}{D^2V^2\rho}.$$

Second π-term $\pi_2 = D^{a_2} \cdot V^{b_2} \cdot \Delta^{c_2} \cdot \omega$

Writing dimensions on both sides,

$$M^0L^0T^0 = L^{a_2} \cdot (LT^{-1})^{b_2} \cdot (ML^{-3})^{c_2} \cdot T^{-1}$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_2$, $\therefore \quad c_2 = 0$

Power of L, $0 = a_2 + b_2 - 3c_2$, $\therefore \quad a_2 = -b_2 + 3c_2 = 1 + 0 = 1$

Power of T, $0 = -b_2 - 1$, $\therefore \quad b_2 = -1$

Substituting the values of a_2, b_2, c_2 in π_2,

$$\pi_2 = D^1 \cdot V^{-1} \cdot \rho^0 \cdot \omega = \frac{D\omega}{V}.$$

Third π-term $\pi_3 = D^{a_3} \cdot V^{b_3} \cdot \rho^{c_3} \cdot \mu.$

Writing dimensions on both sides,

$$M^0L^0T^0 = D^{a_3} \cdot (LT^{-1})^{b_3} \cdot (ML^{-3})^{c_3} \cdot ML^{-1}T^{-1}$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_3 + 1$, $\therefore \quad c_3 = -1$

Power of L, $0 = a_3 + b_3 - 3c_3 - 1$, $\therefore \quad a_3 = -b_3 + 3c_3 + 1 = 1 - 3 + 1 = -1$

Power of T, $0 = -b_3 - 1$ $\therefore \quad b_3 = -1$

Substituting the values of a_3, b_3 and c_3 in π_3,

$$\pi_3 = D^{-1} \cdot V^{-1} \cdot \rho^{-1} \cdot \mu = \frac{\mu}{DV\rho}.$$

Fourth π-term $\pi_4 = D^{a_4} \cdot V^{b_4} \cdot \rho^{c_4} C.$

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_4} \cdot (LT^{-1})^{b_4} \cdot (ML^{-3})^{c_4} \cdot LT^{-1}$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_4$, $\therefore \quad c_4 = 0$
Power of L, $0 = a_4 + b_4 - 3c_4 + 1$, $\therefore \quad a_4 = -b_4 + 3c_4 - 1 = 1 + 0 - 1 = 0$
Power of T, $0 = -a_4 - 1$, $\therefore \quad b_4 = -1$

Substituting the values of a_4, b_4 and c_4 in π_4,

$$\pi_4 = D^0 \,.\, V^{-1} .\, \rho^0 \,.\, C = \frac{C}{V}.$$

Substituting the values of π_1, π_2, π_3 and π_4 in equation (*ii*),

$$f_1\left(\frac{P}{D^2V^2\rho}, \frac{D\omega}{V}, \frac{\mu}{DV\rho}, \frac{C}{V}\right) = 0 \quad \text{or} \quad \frac{P}{D^2V^2\rho} = \phi\left(\frac{D\omega}{V}, \frac{\mu}{DV\rho}, \frac{C}{V}\right)$$

or
$$\mathbf{P = D^2V^2\rho\ \phi\left(\frac{D\omega}{V}, \frac{\mu}{DV\rho}, \frac{C}{V}\right). \text{ Ans.}}$$

Problem 12.13 *The frictional torque T of a disc of diameter D rotating at a speed N in a fluid of viscosity μ and density ρ in a turbulent flow is given by* $T = D^5N^2\,\rho\phi\left[\frac{\mu}{D^2N\rho}\right]$.

Prove this by the method of dimensions.

Solution. Given : $T = f(D, N, \mu, \rho)$ or $f_1(T, D, N, \mu, \rho) = 0$ ...(*i*)

$\therefore$ Total number of variables, $n = 5$

Dimensions of each variable are expressed as

$$T = ML^2T^{-2},\ D = L,\ N = T^{-1},\ \mu = ML^{-1}T^{-1},\ \rho = ML^{-3}$$

$\therefore$ Number of fundamental dimensions, $m = 3$

Number of π-terms $= n - m = 5 - 3 = 2$

Hence equation (*i*) can be written as $f_1(\pi_1, \pi_2) = 0$ ...(*ii*)

Each π-term contains $m + 1$ variable, *i.e.*, $3 + 1 = 4$ variables. Three variables are repeating variables.

Choosing D, N, ρ as repeating variables, the π-terms are

$$\pi_1 = D^{a_1} .\, N^{b_1} .\, \rho^{c_1} .\, T$$
$$\pi_2 = D^{a_2} .\, N^{b_2} .\, \rho^{c_2} \,.\, \mu$$

Dimensional Analysis of π_1

$$\pi_1 = D^{a_1} \,.\, N^{b_1} \,.\, \rho^{c_1} \,.\, T$$

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_1} \,.\, (T^{-1})^{b_1} \,.\, (ML^{-3})^{c_1} \,.\, ML^2T^{-2}.$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_1 + 1$, $\therefore \quad c_1 = -1$
Power of L, $0 = a_1 - 3c_1 + 2$, $\therefore \quad a_1 = 3c_1 - 2 = -3 - 2 = -5$
Power of T, $0 = -b_1 - 2$, $\therefore \quad b_1 = -2$

Substituting the values of a_1, b_1, c_1 in π,

$$\pi_1 = D^{-5} \,.\, N^{-2} \,.\, \rho^{-1} \,.\, T = \frac{T}{D^5N^2\rho}.$$

Dimensional Analysis of π_2

$$\pi_2 = D^{a_2} \,.\, N^{b_2} \,.\, \rho^{c_2} \,.\, \mu$$

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_2} \cdot (T^{-1})^{b_2} \cdot (ML^{-3})^{c_2} \cdot ML^{-1}T^{-1}.$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_2 + 1$, $\therefore$ $c_2 = -1$

Power of L, $0 = a_2 - 3c_2 - 1$, $\therefore$ $a_2 = 3c_2 + 1 = -3 + 1 = -2$

Power of T, $0 = -b_2 - 1$, $\therefore$ $b_2 = -1$

Substituting the values of a_2, b_2 and c_2 in π_2,

$$\pi_2 = D^{-2}N^{-1}\rho^{-1} \cdot \mu = \frac{\mu}{D^2N\rho}.$$

Substituting the values of π_1 and π_2 in equation (*ii*),

$$f_1\left(\frac{T}{D^5N^2\rho}, \frac{\mu}{D^2N\rho}\right) = 0 \quad \text{or} \quad \frac{T}{D^5N^2\rho} = \phi\left(\frac{\mu}{D^2N\rho}\right)$$

or

$$T = \mathbf{D^5N^2\,\rho\,\phi\left[\frac{\mu}{D^2N\rho}\right]}. \textbf{ Ans.}$$

Problem 12.14 *Using Buckingham's π-theorem, show that the discharge Q consumed by an oil ring is given by*

$$Q = Nd^3\phi\left[\frac{\mu}{\rho Nd^2}, \frac{\sigma}{\rho N^2d^3}, \frac{w}{\rho N^2d}\right]$$

where d is the internal diameter of the ring, N is rotational speed, ρ is density, μ is viscosity, σ is surface tension and w is the specific weight of oil.

Solution. Given : $Q = f(d, N, \rho, \mu, \sigma, w)$ or $f_1(Q, d, N, \rho, \mu, \sigma, w) = 0$ ...(*i*)

$\therefore$ Total number of variables, $n = 7$

Dimensions of each variables are

$$Q = L^3T^{-1},\ d = L,\ N = T^{-1},\ \rho = ML^{-3},\ \mu = ML^{-1}T^{-1},\ \sigma = MT^{-2}$$

and $w = ML^{-2}T^{-2}$

$\therefore$ Total number of fundamental dimensions, $m = 3$

$\therefore$ Total number of π-terms $= n - m = 7 - 3 = 4$

$\therefore$ Equation (*i*) becomes as $f_1(\pi_1, \pi_2, \pi_3, \pi_4) = 0$ ...(*ii*)

Choosing d, N, ρ as repeating variables, the π-terms are

$$\pi_1 = d^{a_1} \cdot N^{b_1} \cdot \rho^{c_1} \cdot Q$$
$$\pi_2 = d^{a_2} \cdot N^{b_2} \cdot \rho^{c_2} \cdot \mu$$
$$\pi_3 = d^{a_3} \cdot N^{b_3} \cdot \rho^{c_3} \cdot \sigma$$
$$\pi_4 = d^{a_4} \cdot N^{b_4} \cdot \rho^{c_4} \cdot w.$$

First π-term $\pi_1 = d^{a_1} \cdot N^{b_1} \cdot \rho^{c_1} \cdot Q$.

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_1} \cdot (T^{-1})^{b_1} \cdot (ML^{-3})^{c_1} \cdot L^3T^{-1}.$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_1$, $\therefore$ $c_1 = 0$

Power of L, $0 = a_1 - 3c_1 + 3$, $\therefore$ $a_1 = 3c_1 - 3 = 0 - 3 = -3$

Power of T, $0 = -b_1 - 1$, $\therefore$ $b_1 = -1$

Substituting a_1, b_1, c_1 in π_1, $\pi_1 = d^{-3} \cdot N^{-1} \cdot \rho^0 \cdot Q = \dfrac{Q}{d^3N}$

Second π-term $\pi_2 = d^{a_2} \cdot N^{b_2} \cdot \rho^{c_2} \cdot \mu.$

Substituting the dimensions on both sides,

$$M^0L^0T^0 = L^{a_2} \cdot (T^{-1})^{b_2} \cdot (ML^{-3})^{c_2} \cdot ML^{-1}\ T^{-1}$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_2 + 1, \quad \therefore \quad c_2 = -1$

Power of L, $0 = a_2 - 3c_2 - 1,$

$\therefore$ $a_2 = 3c_2 + 1 = -3 + 1 = -2$

Power of T, $0 = -b_2 - 1, \quad \therefore \quad b_2 = -1$

Substituting the values of a_2, b_2, c_2 in π_2,

$$\pi_2 = d^{-2} \cdot N^{-1} \cdot \rho^{-1} \cdot \mu = \frac{\mu}{d^2 N\rho} \quad \text{or} \quad \frac{\mu}{\rho N d^2}.$$

Third π-term $\pi_3 = d^{a_3} \cdot N^{b_3} \cdot \rho^{c_3} \cdot \sigma.$

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_3} \cdot (T^{-1})^{b_3} \cdot (ML^{-3})^{c_3} \cdot MT^{-2}.$$

Equating the powers of M, L, T on the sides,

Power of M, $0 = c_3 + 1, \quad \therefore \quad c_3 = -1$

Power of L, $0 = a_3 - 3c_3, \quad \therefore \quad a_3 = 3c_3 = -3$

Power of T, $0 = -b_3 - 2, \quad \therefore \quad b_3 = -2$

Substituting the values of a_3, b_3, c_3 in π_3,

$$\pi_3 = d^{-3} \cdot N^{-2} \cdot \rho^{-1} \cdot \sigma = \frac{\sigma}{d^3 N^2 \rho}.$$

Fourth π-term $\pi_4 = d^{a_4} \cdot N^{b_4} \cdot \rho^{c_4} \cdot w$

Substituting dimensions on both sides,

$$M^0L^0T^0 = L^{a_4} \cdot (T^{-1})^{b_4} \cdot (ML^{-3})^{c_4} \cdot ML^{-2}T^{-2}.$$

Equating the powers of M, L, T on both sides,

Power of M, $0 = c_4 + 1, \quad \therefore \quad c_4 = -1$

Power of L, $0 = a_4 - 3c_4 - 2, \quad \therefore \quad a_4 = 3c_4 + 2 = -3 + 2 = -1$

Power of T, $0 = -b_4 - 2, \quad \therefore \quad b_4 = -2$

Substituting the values of a_4, b_4 and c_4 in π_4,

$$\pi_4 = d^{-1} \cdot N^{-2} \cdot \rho^{-1} \cdot w = \frac{w}{dN^2\rho}.$$

Now substituting the values of π_1, π_2, π_3, π_4 in (*ii*),

$$f\left(\frac{Q}{d^3N}, \frac{\mu}{\rho N d^2}, \frac{\sigma}{d^3N^2\rho}, \frac{w}{dN^2\rho}\right) = 0 \quad \text{or} \quad \frac{Q}{d^3N} = f_1\left[\frac{\mu}{\rho N d^2}, \frac{\sigma}{d^3N^2\rho}, \frac{w}{dN^2\rho}\right]$$

or $$Q = \mathbf{d^3N\phi}\left[\frac{\mu}{\mathbf{\rho N d^2}}, \frac{\sigma}{\mathbf{d^3N^2d}}, \frac{\mathbf{w}}{\mathbf{dN^2\rho}}\right]. \textbf{ Ans.}$$

▶ 12.5 MODEL ANALYSIS

For predicting the performance of the hydraulic structures (such as dams, spillways etc.) or hydraualic machines (such as turbines, pumps etc.), before actually constructing or manufacturing,

models of the structures or machines are made and tests are performed on them to obtain the desired information.

The **model** is the small scale replica of the actual structure or machine. The actual structure or machine is called **Prototype**. It is not necessary that the models should be smaller than the prototypes (though in most of cases it is), they may be larger than the prototype. The study of models of actual machines is called **Model analysis**. Model analysis is actually an experimental method of finding solutions of complex flow problems. Exact analytical solutions are possible only for a limited number of flow problems. The followings are the advantages of the dimensional and model analysis :

1. The performance of the hydraulic structure or hydraulic machine can be easily predicted, in advance, from its model.
2. With the help of dimensional analysis, a relationship between the variables influencing a flow problem in terms of dimensionless parameters is obtained. This relationship helps in conducting tests on the model.
3. The merits of alternative designs can be predicted with the help of model testing. The most economical and safe design may be, finally, adopted.
4. The tests performed on the models can be utilized for obtaining, in advance, useful information about the performance of the prototypes only if a complete similarity exists between the model and the prototype.

▶ 12.6 SIMILITUDE-TYPES OF SIMILARITIES

Similitude is defined as the similarity between the model and its prototype in every respect, which means that the model and prototype have similar properties or model and prototype are completely similar. Three types of similarities must exist between the model and prototype. They are

1. Geometric Similarity, 2. Kinematic Similarity, and 3. Dynamic Similarity.

1. Geometric Similarity. The geometric similarity is said to exist between the model and the prototype. The ratio of all corresponding linear dimension in the model and prototype are equal.

Let L_m = Length of model, b_m = Breadth of model,
D_m = Diameter of model, A_m = Area of model,
$\forall_m$ = Volume of model,
and $L_P, b_P, D_P, A_P, \forall_P$ = Corresponding values of the prototype.

For geometric similarity between model and prototype, we must have the relation,

$$\frac{L_P}{L_m} = \frac{b_P}{b_m} = \frac{D_P}{D_m} = L_r \qquad ...(12.6)$$

where L_r is called the scale ratio.

For area's ratio and volume's ratio the relation should be as given below :

$$\frac{A_P}{A_m} = \frac{L_P \times b_P}{L_m \times b_m} = L_r \times L_r = L_r^2 \qquad ...(12.7)$$

and

$$\frac{\forall_P}{\forall_m} = \left(\frac{L_P}{L_m}\right)^3 = \left(\frac{b_P}{b_m}\right)^3 = \left(\frac{D_P}{D_m}\right)^3 \qquad ...(12.8)$$

2. Kinematic Similarity. Kinematic similarity means the similarity of motion between model and prototype. Thus kinematic similarity is said to exist between the model and the prototype if the ratios of the velocity and acceleration at the corresponding points in the model and at the corresponding

points in the prototype are the same. Since velocity and acceleration are vector quantities, hence not only the ratio of magnitude of velocity and acceleration at the corresponding points in model and prototype should be same ; but the directions of velocity and accelerations at the corresponding points in the model and prototype also should be parallel.

Let V_{P_1} = Velocity of fluid at point 1 in prototype,
V_{P_2} = Velocity of fluid at point 2 in prototype,
a_{P_1} = Acceleration of fluid at point 1 in prototype,
a_{P_2} = Acceleration of fluid at point 2 in prototype, and
$V_{m_1}, V_{m_2}, a_{m_1}, a_{m_2}$ = Corresponding values at the corresponding points of fluid velocity and acceleration in the model.

For kinematic similarity, we must have

$$\frac{V_{P_1}}{V_{m_1}} = \frac{V_{P_2}}{V_{m_2}} = V_r \qquad ...(12.9)$$

where V_r is the velocity ratio.

For acceleration, we must have $\frac{a_{P_1}}{a_{m_1}} = \frac{a_{P_2}}{a_{m_2}} = a_r$ ...(12.10)

where a_r is the acceleration ratio.

Also the directions of the velocities in the model and prototype should be same.

3. Dynamic Similarity. Dynamic similarity means the similarity of forces between the model and prototype. Thus dynamic similarity is said to exist between the model and the prototype if the ratios of the corresponding forces acting at the corresponding points are equal. Also the directions of the corresponding forces at the corresponding points should be same.

Let $(F_i)_P$ = Inertia force at a point in prototype,
$(F_v)_P$ = Viscous force at the point in prototype,
$(F_g)_P$ = Gravity force at the point in prototype,
and $(F_i)_m$ $(F_v)_m$, $(F_g)_m$ = Corresponding values of forces at the corresponding point in model.

Then for dynamic similarity, we have

$$\frac{(F_i)_P}{(F_i)_m} = \frac{(F_v)_P}{(F_v)_m} = \frac{(F_g)_P}{(F_g)_m} \ldots = F_r, \text{ where } F_r \text{ is the force ratio.}$$

Also the directions of the corresponding forces at the corresponding points in the model and prototype should be same.

▶ 12.7 TYPES OF FORCES ACTING IN MOVING FLUID

For the fluid flow problems, the forces acting on a fluid mass may be any one, or a combination of the several of the following forces :

1. Inertia force, F_i.
2. Viscous force, F_v.
3. Gravity force, F_g.
4. Pressure force, F_p.
5. Surface tension force, F_s.
6. Elastic force, F_e.

1. **Inertia Force** (F_i). It is equal to the product of mass and acceleration of the flowing fluid and acts in the direction opposite to the direction of acceleration. It is always existing in the fluid flow problems.

2. **Viscous Force** (F_v). It is equal to the product of shear stress (τ) due to viscosity and surface area of the flow. It is present in fluid flow problems where viscosity is having an important role to play.

3. **Gravity Force** (F_g). It is equal to the product of mass and acceleration due to gravity of the flowing fluid. It is present in case of open surface flow.

4. **Pressure Force** (F_p). It is equal to the product of pressure intensity and cross-sectional area of the flowing fluid. It is present in case of pipe-flow.

5. **Surface Tension Force** (F_s). It is equal to the product of surface tension and length of surface of the flowing fluid.

6. **Elastic Force** (F_e). It is equal to the product of elastic stress and area of the flowing fluid.

For a flowing fluid, the above-mentioned forces may not always be present. And also the forces, which are present in a fluid flow problem, are not of equal magnitude. There are always one or two forces which dominate the other forces. These dominating forces govern the flow of fluid.

▶ 12.8 DIMENSIONLESS NUMBERS

Dimensionless numbers are those numbers which are obtained by dividing the inertia force by viscous force or gravity force or pressure force or surface tension force or elastic force. As this is a ratio of one force to the other force, it will be a dimensionless number. These dimensionless numbers are also called non-dimensional parameters. The followings are the important dimensionless numbers :

1. Reynold's number,
2. Froude's number,
3. Euler's number,
4. Weber's number,
5. Mach's number.

12.8.1 Reynold's Number (R_e). It is defined as the ratio of inertia force of a flowing fluid and the viscous force of the fluid. The expression for Reynold's number is obtained as

Inertia force (F_i) = Mass × Acceleration of flowing fluid

$$= \rho \times \text{Volume} \times \frac{\text{Velocity}}{\text{Time}} = \rho \times \frac{\text{Volume}}{\text{Time}} \times \text{Velocity}$$

$$= \rho \times AV \times V \quad \{\because \text{ Volume per sec} = \text{Area} \times \text{Velocity} = A \times V\}$$

$$= \rho AV^2 \qquad \ldots(12.11)$$

Viscous force (F_v) = Shear stress × Area $\left\{\because \tau = \mu \frac{du}{dy} \quad \therefore \text{ Force} = \tau \times \text{Area}\right\}$

$$= \tau \times A$$

$$= \left(\mu \frac{du}{dy}\right) \times A = \mu \cdot \frac{V}{L} \times A \qquad \left\{\because \frac{du}{dy} = \frac{V}{L}\right\}$$

By definition, Reynold's number,

$$R_e = \frac{F_i}{F_v} = \frac{\rho AV^2}{\mu \cdot \frac{V}{L} \times A} = \frac{\rho VL}{\mu}$$

$$= \frac{V \times L}{(\mu / \rho)} = \frac{V \times L}{\nu} \qquad \left\{\because \frac{\mu}{\rho} = \nu = \text{Kinematic viscosity}\right\}$$

In case of pipe flow, the linear dimension L is taken as diameter, d. Hence Reynold's number for pipe flow,

$$R_e = \frac{V \times d}{\nu} \quad \text{or} \quad \frac{\rho V d}{\mu}. \qquad ...(12.12)$$

12.8.2 Froude's Number (F_e). The Froude's number is defined as the square root of the ratio of inertia force of a flowing fluid to the gravity force. Mathematically, it is expressed as

$$F_e = \sqrt{\frac{F_i}{F_g}}$$

where F_i from equation (12.11) $= \rho A V^2$

and F_g = Force due to gravity

= Mass × Acceleration due to gravity

= $\rho \times$ Volume $\times g = \rho \times L^3 \times g$ $\{\because$ Volume $= L^3\}$

= $\rho \times L^2 \times L \times g = \rho \times A \times L \times g$ $\{\because$ $L^2 = A$ = Area$\}$

$$\therefore \quad F_e = \sqrt{\frac{F_i}{F_g}} = \sqrt{\frac{\rho A V^2}{\rho A L g}} = \sqrt{\frac{V^2}{Lg}} = \frac{V}{\sqrt{Lg}} \qquad ...(12.13)$$

12.8.3 Euler's Number (E_u). It is defined as the square root of the ratio of the inertia force of a flowing fluid to the pressure force. Mathematically, it is expressed as

$$E_u = \sqrt{\frac{F_i}{F_P}}$$

where F_P = Intensity of pressure × Area = $p \times A$

and $F_i = \rho A V^2$

$$\therefore \quad E_u = \sqrt{\frac{\rho A V^2}{p \times A}} = \sqrt{\frac{V^2}{p/\rho}} = \frac{V}{\sqrt{p/\rho}} \qquad ...(12.14)$$

12.8.4 Weber's Number (W_e). It is defined as the square root of the ratio of the inertia force of a flowing fluid to the surface tension force. Mathematically, it is expressed as

Weber's Number, $$W_e = \sqrt{\frac{F_i}{F_s}}$$

where F_i = Inertia force = $\rho A V^2$

and F_s = Surface tension force

= Surface tension per unit length × Length = $\sigma \times L$

$$\therefore \quad W_e = \sqrt{\frac{\rho A V^2}{\sigma \times L}} = \sqrt{\frac{\rho \times L^2 \times V^2}{\sigma \times L}} \qquad \{\because \quad A = L^2\}$$

$$= \sqrt{\frac{\rho L \times V^2}{\sigma}} = \sqrt{\frac{V^2}{\sigma/\rho L}} = \frac{V}{\sqrt{\sigma/\rho L}}. \qquad ...(12.15)$$

12.8.5 Mach's Number (M). Mach's number is defined as the square root of the ratio of the inertia force of a flowing fluid to the elastic force. Mathematically, it is defined as

$$M = \sqrt{\frac{\text{Inertia force}}{\text{Elastic force}}} = \sqrt{\frac{F_i}{F_e}}$$

where $F_i = \rho A V^2$

and F_e = Elastic force = Elastic stress × Area

$= K \times A = K \times L^2$ {$\because$ K = Elastic stress}

$$\therefore \quad M = \sqrt{\frac{\rho A V^2}{K \times L^2}} = \sqrt{\frac{\rho \times L^2 \times V^2}{K \times L^2}} = \sqrt{\frac{V^2}{K/\rho}} = \frac{V}{\sqrt{K/\rho}}$$

But $\sqrt{\frac{K}{\rho}} = C$ = Velocity of sound in the fluid

$$\therefore \quad M = \frac{V}{C}. \qquad ...(12.16)$$

▶ 12.9 MODEL LAWS OR SIMILARITY LAWS

For the dynamic similarity between the model and the prototype, the ratio of the corresponding forces acting at the corresponding points in the model and prototype should be equal. The ratio of the forces are dimensionless numbers. It means for dynamic similarity between the model and prototype, the dimensionless numbers should be same for model and the prototype. But it is quite difficult to satisfy the condition that all the dimensionless numbers (*i.e.*, R_e, F_e, W_e, E_u and M) are the same for the model and prototype. Hence models are designed on the basis of ratio of the force, which is dominating in the phenomenon. The laws on which the models are designed for dynamic similarity are called model laws or laws of similarity. The followings are the model laws :

1. Reynold's model law,
2. Froude model law,
3. Euler model law,
4. Weber model law,
5. Mach model law.

12.9.1 Reynold's Model Law. Reynold's model law is the law in which models are based on Reynold's number. Models based on Reynold's number includes :

(*i*) Pipe flow

(*ii*) Resistance experienced by sub-marines, airplanes, fully immersed bodies etc.

As defined earlier that Reynold number is the ratio of inertia force and viscous force, and hence fluid flow problems where viscous forces alone are predominent, the models are designed for dynamic similarity on Reynolds law, which states that the Reynold number for the model must be equal to the Reynold number for the prototype.

Let V_m = Velocity of fluid in model,

ρ_m = Density of fluid in model,

L_m = Length or linear dimension of the model,

μ_m = Viscosity or fluid in model,

and V_P, ρ_P, L_P and μ_P are the corresponding values of velocity, density, linear dimension and viscosity of fluid in prototype. Then according to Reynold's model law,

$$[R_e]_m = [R_e]_P \text{ or } \frac{\rho_m V_m L_m}{\mu_m} = \frac{\rho_P V_P L_P}{\mu_P} \qquad ...(12.17)$$

or $$\frac{\rho_P \cdot V_P \cdot L_P}{\rho_m \cdot V_m \cdot L_m} \times \frac{1}{\frac{\mu_P}{\mu_m}} = 1 \quad \text{or} \quad \frac{\rho_r \cdot V_r \cdot L_r}{\mu_r} = 1$$

$$\left\{\text{where} \quad \rho_r = \frac{\rho_P}{\rho_m}, V_r = \frac{V_P}{V_m} \text{ and } L_r = \frac{L_P}{L_m}, \frac{\mu_P}{\mu_m} = \mu_r\right\}$$

And also ρ_r, V_r, L_r and μ_r are called the scale ratios for density, velocity, linear dimension and viscosity.

The scale ratios for time, acceleration, force and discharge for Reynold's model law are obtained as

$$t_r = \text{Time scale ratio} = \frac{L_r}{V_r} \qquad \left\{\because \quad V = \frac{L}{t} \quad \therefore \quad t = \frac{L}{V}\right\}$$

$$a_r = \text{Acceleration scale ratio} = \frac{V_r}{t_r}$$

$$F_r = \text{Force scale ratio} = (\text{Mass} \times \text{Acceleration})_r$$

$$= m_r \times a_r = \rho_r A_r V_r \times a_r \qquad \{A_r = \text{Area ratio}\}$$

$$= \rho_r L_r^2 V_r \times a_r$$

$$Q_r = \text{Discharge scale ratio} = (\rho AV)_r$$

$$= \rho_r A_r V_r = \rho_r \cdot L_r^2 \cdot V_r.$$

Problem 12.15 *A pipe of diameter 1.5 m is required to transport an oil of sp. gr. 0.90 and viscosity 3×10^{-2} poise at the rate of 3000 litre/s. Tests were conducted on a 15 cm diameter pipe using water at 20°C. Find the velocity and rate of flow in the model. Viscosity of water at 20°C = 0.01 poise.*

Solution. Given :

Dia. of prototype, $D_P = 1.5$ m

Viscosity of fluid, $\mu_P = 3 \times 10^{-2}$ poise

Q for prototype, $Q_P = 3000$ lit/s $= 3.0$ m^3/s

Sp. gr. of oil, $S_P = 0.9$

$\therefore$ Density of oil, $\rho_P = S_P \times 1000 = 0.9 \times 1000 = 900$ kg/m^3

Dia. of the model, $D_m = 15$ cm $= 0.15$ m

Viscosity of water at 20°C $= .01$ poise $= 1 \times 10^{-2}$ poise or $\mu_m = 1 \times 10^{-2}$ poise

Density of water or $\rho_m = 1000$ kg/m^3.

For pipe flow, the dynamic similarity will be obtained if the Reynold's number in the model and prototype are equal

Hence using equation (12.17), $\frac{\rho_m V_m D_m}{\mu_m} = \frac{\rho_P V_P D_P}{\mu_P}$ {For pipe, linear dimension is D}

$\therefore$ $$\frac{V_m}{V_P} = \frac{\rho_P}{\rho_m} \cdot \frac{D_P}{D_m} \cdot \frac{\mu_m}{\mu_P}$$

$$= \frac{900}{1000} \times \frac{1.5}{0.15} \times \frac{1 \times 10^{-2}}{3 \times 10^{-2}} = \frac{900}{1000} \times 10 \times \frac{1}{3} = 3.0$$

But $$V_P = \frac{\text{Rate of flow in prototype}}{\text{Area of prototype}} = \frac{3.0}{\frac{\pi}{4}(D_P)^2} = \frac{3.0}{\frac{\pi}{4}(1.5)^2}$$

$$= \frac{3.0 \times 4}{\pi \times 2.25} = 1.697 \text{ m/s}$$

$\therefore$ $V_m = 3.0 \times V_P = 3.0 \times 1.697 =$ **5.091 m/s. Ans.**

Rate of flow through model, $Q_m = A_m \times V_m = \frac{\pi}{4}(D_m)^2 \times V_m = \frac{\pi}{4}(0.15)^2 \times 5.091 \text{ m}^3/\text{s}$

$= 0.0899 \text{ m}^3/\text{s} = 0.0899 \times 1000 \text{ lit/s} =$ **89.9 lit/s. Ans.**

Problem 12.16 *Water is flowing through a pipe of diameter 30 cm at a velocity of 4 m/s. Find the velocity of oil flowing in another pipe of diameter 10 cm, if the condition of dynamic similarity is satisfied between the two pipes. The viscosity of water and oil is given as 0.01 poise and .025 poise. The sp. gr. of oil = 0.8.*

Solution. Given :

Two pipes having different liquids.

Let for pipe 1, Liquid = Water

Dia. of pipe, $d_1 = 30 \text{ cm} = 0.30 \text{ m}$

Velocity of flow, $V_1 = 4 \text{ m/s}$

Viscosity, $\mu_1 = 0.01 \text{ poise} = \frac{0.01}{10}$ (S.I. Units)

Density, $\rho_1 = 1000 \text{ kg/m}^3$

For pipe 2, Liquid = Oil

Dia. of pipe, $d_2 = 10 \text{ cm} = 0.1 \text{ m}$

Velocity of flow, $V_2 = ?$

Viscosity, $\mu_2 = 0.025 \text{ poise} = \frac{0.025}{10}$ (S.I. Units)

Sp. gr. of oil $= 0.8$

$\therefore$ Density, $\rho_2 = 0.8 \times 1000 = 800 \text{ kg/m}^3$

If the pipes are dynamically similar, the Reynold's number for both the pipes should be same.

$\therefore$ $$\frac{\rho_1 V_1 d_1}{\mu_1} = \frac{\rho_2 V_2 d_2}{\mu_2} \quad \text{or} \quad V_2 = \frac{\rho_1}{\rho_2} \cdot \frac{d_1}{d_2} \cdot \frac{\mu_2}{\mu_1} V_1$$

$$= \frac{1000}{800} \times \frac{0.30}{0.10} \times \frac{\frac{.025}{10}}{\frac{.01}{10}} \times 4.0 = \frac{1000}{800} \times 3 \times \frac{.025}{.01} \times 4.0$$

$=$ **37.5 m/s. Ans.**

Problem 12.17 *The ratio of lengths of a sub-marine and its model is 30 : 1. The speed of sub-marine (prototype) is 10 m/s. The model is to be tested in a wind tunnel. Find the speed of air in wind tunnel. Also determine the ratio of the drag (resistance) between the model and its prototype. Take the value of kinematic viscosities for sea water and air as .012 stokes and .016 stokes respectively. The density for sea-water and air is given as 1030 kg/m^3 and 1.24 kg/m^3 respectively.*

Solution. Given :

Prototype (sub-marine) and its model.

For prototype, Speed $V_P = 10$ m/s

Fluid = Sea - water

Kinematic viscosity, $\nu_P = 0.012$ stokes $= .012$ cm^2/s

$= .012 \times 10^{-4}$ m^2/s $\quad \{\because$ Stoke = cm^2/s$\}$

Density, $\rho_P = 1030$ kg/m^3

For model Fluid = Air

Kinematic viscosity, $\nu_m = 0.016$ stokes $= 0.016$ cm^2/s $= .016 \times 10^{-4}$ m^2/s

Density, $\rho_m = 1.24$ kg/m^3

Also $\dfrac{\text{Length of prototype}}{\text{Length of model}} = \dfrac{L_P}{L_m} = 30.0$

Let the velocity of air in model $= V_m$.

For dynamic similarity between model and sub-marine, the viscous resistance is to be overcame and hence for fully submerged sub-marine, the Reynold's number for model and prototype should be same.

$$\therefore \quad \frac{\rho_P V_P D_P}{\mu_P} = \frac{\rho_m V_m D_m}{\mu_m} \quad \text{or} \quad \frac{V_P D_P}{(\mu/\rho)_P} = \frac{V_m D_m}{(\mu/\rho)_m} \; ; \; \frac{V_P D_P}{\nu_P} = \frac{V_m D_m}{\nu_m}$$

$$\therefore \quad V_m = \frac{\nu_m}{\nu_P} \times \frac{D_P}{D_m} \times V_P$$

$$= \frac{0.016 \times 10^{-4}}{.012 \times 10^{-4}} \times 30 \times 10 \text{ m/s} \qquad \left\{\because \quad \frac{D_P}{D_m} = \frac{L_P}{L_m} = 30\right\}$$

$$= \frac{0.016}{.012} \times 30 \times 10 = \textbf{400 m/s. Ans.}$$

Ratio of drag force (resistance) :

Drag force = Mass × Acceleration

$$= \rho L^3 \times \frac{V}{t} = \rho \, . \, L^2 \, . \, \frac{L}{t} \times V = \rho \, L^2 V^2 \qquad \left\{\because \quad \frac{L}{t} = V\right\}$$

Let F_P and F_m denote the drag force for the prototype and for the model respectively then,

$$\frac{F_P}{F_m} = \frac{\rho_P L_P^2 V_P^2}{\rho_m L_m^2 V_m^2} = \frac{\rho_P}{\rho_m} \times \left(\frac{L_P}{L_m}\right)^2 \times \left(\frac{V_P}{V_m}\right)^2$$

$$= \frac{1030}{1.24} \times 30^2 \times \left(\frac{10}{400}\right)^2 = \textbf{467.22. Ans.}$$

Problem 12.18 *A ship 300 m long moves in sea-water, whose density is 1030 kg/m^3, A 1 : 100 model of this ship is to be tested in a wind tunnel. The velocity of air in the wind tunnel around the model is 30 m/s and the resistance of the model is 60 N. Determine the velocity of ship in sea-water and also the resistance of the ship in sea-water. The density of air is given as 1.24 kg/m^3. Take the kinematic viscosity of sea-water and air as 0.012 stokes and 0.018 stokes respectively.*

Solution. Given :

For Prototype,

Length, $L_P = 300$ m

Fluid = Sea-water

Density of water $= 1030$ kg/m^3

Kinematic viscosity, $\nu_P = 0.012$ stokes $= 0.012 \times 10^{-4}$ m^2/s

Let velocity of ship $= V_P$

Resistance $= F_P$

For model

Length, $L_m = \frac{1}{100} \times 300 = 3$ m

Velocity, $V_m = 30$ m/s

Resistance, $F_m = 60$ N

Density of air, $\rho_m = 1.24$ kg/m^3

Kinematic viscosity of air, $\nu_m = 0.018$ stokes $= .018 \times 10^{-4}$ m^2/s.

For dynamic similarity between the prototype and its model, the Reynolds number for both of them should be equal.

$$\therefore \quad \frac{V_P \times L_P}{\nu_P} = \frac{V_m \times L_m}{\nu_m} \quad \text{or} \quad V_P = \frac{\nu_P}{\nu_m} \times \frac{L_m}{L_P} \times V_m$$

$$= \frac{.012 \times 10^{-4}}{.018 \times 10^{-4}} \times \frac{3}{300} \times 30 = \frac{1}{1.5} \times \frac{1}{100} \times 30 = \textbf{0.2 m/s. Ans.}$$

Resistance = Mass × Acceleration

$$= \rho L^3 \times \frac{V}{t} = \rho L^2 \times \frac{V}{1} \times \frac{L}{t} = \rho L^2 V^2$$

Then $$\frac{F_P}{F_m} = \frac{\left(\rho L^2 V^2\right)_P}{\left(\rho L^2 V^2\right)_m} = \frac{\rho_P}{\rho_m} \times \left(\frac{L_P}{L_m}\right)^2 \times \left(\frac{V_P}{V_m}\right)^2$$

But $$\frac{\rho_P}{\rho_m} = \frac{1030}{1.24}$$

$$\therefore \quad \frac{F_P}{F_m} = \frac{1030}{1.24} \times \left(\frac{300}{3}\right)^2 \times \left(\frac{0.2}{30}\right)^2 = 369.17$$

$$\therefore \quad F_P = 369.17 \times F_m = 369.17 \times 60 = \textbf{22150.2 N. Ans.}$$

12.9.2 Froude Model Law. Froude model law is the law in which the models are based on Froude number which means for dynamic similarity between the model and prototype, the Froude number for both of them should be equal. Froude model law is applicable when the gravity force is only predominant force which controls the flow in addition to the force of inertia. Froude model law is applied in the following fluid flow problems :

1. Free surface flows such as flow over spillways, weirs, sluices, channels etc.,
2. Flow of jet from an orifice or nozzle,
3. Where waves are likely to be formed on surface,
4. Where fluids of different densities flow over one another.

Let V_m = Velocity of fluid in model,
L_m = Linear dimension or length of model,
g_m = Acceleration due to gravity at a place where model is tested.

and V_P, L_P and g_P are the corresponding values of the velocity, length and acceleration due to gravity for the prototype. Then according to Froude model law,

$$(F_e)_{model} = (F_e)_{prototype} \text{ or } \frac{V_m}{\sqrt{g_m L_m}} = \frac{V_P}{\sqrt{g_P L_P}} \qquad ...(12.18)$$

If the tests on the model are performed on the same place where prototype is to operate, then $g_m = g_P$ and equation (12.18) becomes as

$$\frac{V_m}{\sqrt{L_m}} = \frac{V_P}{\sqrt{L_P}} \qquad ...(12.19)$$

or $$\frac{V_m}{V_P} \times \frac{1}{\sqrt{\frac{L_m}{L_P}}} = 1$$

$$\frac{V_P}{V_m} = \sqrt{\frac{L_P}{L_m}} = \sqrt{L_r} \qquad \left\{\because \frac{L_P}{L_m} = L_r\right\}$$

where L_r = Scale ratio for length

$\frac{V_P}{V_m} = V_r$ = Scale ratio for velocity.

$\therefore$ $$\frac{V_P}{V_m} = V_r = \sqrt{L_r}\,. \qquad ...(12.20)$$

Scale ratios for various physical quantities based on Froude model law are :

(*a*) **Scale ratio for time**

As $$\text{time} = \frac{\text{Length}}{\text{Velocity}},$$

then ratio of time for prototype and model is

$$T_r = \frac{T_P}{T_m} = \frac{\left(\frac{L}{V}\right)_P}{\left(\frac{L}{V}\right)_m} = \frac{\frac{L_P}{V_P}}{\frac{L_m}{V_m}} = \frac{L_P}{L_m} \times \frac{V_m}{V_P} = L_r \times \frac{1}{\sqrt{L_r}} \qquad \left\{\because \frac{V_P}{V_m} = \sqrt{L_r}\right\}$$

$$= \sqrt{L_r}\,. \qquad ...(12.21)$$

(*b*) **Scale ratio for acceleration**

$$\text{Acceleration} = \frac{V}{T}$$

$\therefore$ $$a_r = \frac{a_P}{a_m} = \frac{\left(\frac{V}{T}\right)_P}{\left(\frac{V}{T}\right)_m} = \frac{V_P}{T_P} \times \frac{T_m}{V_m} = \frac{V_P}{V_m} \times \frac{T_m}{T_P}$$

$$= \sqrt{L_r} \times \frac{1}{\sqrt{L_r}} \qquad \left\{\because \frac{V_P}{V_m} = \sqrt{L_r}, \frac{T_P}{T_m} = \sqrt{L_r}\right\}$$

$$= 1. \qquad ...(12.22)$$

(*c*) **Scale ratio for discharge**

$$Q = A \times V = L^2 \times \frac{L}{T} = \frac{L^3}{T}$$

$$\therefore \quad Q_r = \frac{Q_P}{Q_m} = \frac{\left(\frac{L^3}{T}\right)_P}{\left(\frac{L^3}{T}\right)_m} = \left(\frac{L_P}{L_m}\right)^3 \times \left(\frac{T_m}{T_P}\right) = L_r^{\,3} \times \frac{1}{\sqrt{L_r}} = L_r^{\,2.5} \qquad ...(12.23)$$

(*d*) **Scale ratio for force**

As $\text{Force} = \text{Mass} \times \text{Acceleration} = \rho L^3 \times \frac{V}{T} = \rho L^2 \,.\, \frac{L}{T} \,.\, V = \rho L^2 V^2$

$\therefore$ Ratio for force, $F_r = \frac{F_P}{F_m} = \frac{\rho_P L_P^2 V_P^2}{\rho_m L_m^2 V_m^2} = \frac{\rho_P}{\rho_m} \times \left(\frac{L_P}{L_m}\right)^2 \times \left(\frac{V_P}{V_m}\right)^2.$

If the fluid used in model and prototype is same, then

$$\frac{\rho_P}{\rho_m} = 1 \quad \text{or} \quad \rho_P = \rho_m$$

and hence $F_r = \left(\frac{L_P}{L_m}\right)^2 \times \left(\frac{V_P}{V_m}\right)^2 = L_r^{\,2} \times \left(\sqrt{L_r}\right)^2 = L_r^{\,2} \,.\, L_r = L_r^{\,3}.$ ...(12.24)

(*e*) **Scale ratio for pressure intensity**

As $p = \frac{\text{Force}}{\text{Area}} = \frac{\rho L^2 V^2}{L^2} = \rho V^2$

$\therefore$ Pressure ratio, $p_r = \frac{p_P}{p_m} = \frac{\rho_P V_P^2}{\rho_m V_m^2}$

If fluid is same, then $\rho_P = \rho_m$

$$\therefore \quad p_r = \frac{V_P^2}{V_m^2} = \left(\frac{V_P}{V_m}\right)^2 = L_r. \qquad ...(12.25)$$

(*f*) **Scale ratio for work, energy, torque, moment etc.**

$$\text{Torque} = \text{Force} \times \text{Distance} = F \times L$$

$\therefore$ Torque ratio, $T_r^* = \frac{T_P^*}{T_m^*} = \frac{(F \times L)_P}{(F \times L)_m} = F_r \times L_r = L_r^{\,3} \times L_r = L_r^{\,4}.$ ...(12.26)

(*g*) **Scale ratio for power**

As $\text{Power} = \text{Work per unit time}$

$$= \frac{F \times L}{T}$$

$\therefore$ Power ratio, $$P_r = \frac{P_P}{P_m} = \frac{\dfrac{F_P \times L_P}{T_P}}{\dfrac{F_m \times L_m}{T_m}} = \frac{F_P}{F_m} \times \frac{L_P}{L_m} \times \frac{1}{\dfrac{T_P}{T_m}}$$

$$= F_r \cdot L_r \cdot \frac{1}{T_r} = L_r^3 \cdot L_r \cdot \frac{1}{\sqrt{L_r}} = L^{3.5}. \quad \text{...(12.27)}$$

Problem 12.19 *In 1 in 40 model of a spillway, the velocity and discharge are 2 m/s and 2.5 m^3/s. Find the corresponding velocity and discharge in the prototype.*

Solution. Given :

Scale ratio of length, $L_r = 40$

Velocity in model, $V_m = 2$ m/s

Discharge in model, $Q_m = 2.4$ m^3/s

Let V_P and Q_P are the velocity and discharge in prototype.

Using equation (12.20) for velocity ratio, $\dfrac{V_P}{V_m} = \sqrt{L_r} = \sqrt{40}$

$\therefore$ $V_P = V_m \times \sqrt{40} = 2 \times \sqrt{40}$ = **12.65 m/s. Ans.**

Using equation (12.23) for discharge ratio,

$$\frac{Q_P}{Q_m} = L_r^{2.5} = (40)^{2.5}$$

$\therefore$ $Q_P = Q_m \times 40^{2.5} = 2.5 \times 40^{2.5}$ = **25298.2 m^3/s. Ans.**

Problem 12.20 *A ship model of scale $\dfrac{1}{50}$ is towed through sea water at a speed of 1 m/s. A force of 2 N is required to tow the model. Determine the speed of ship and the propulsive force on the ship, if prototype is subjected to wave resistance only.*

Solution. Given :

Scale ratio of length, $L_r = 50$

Speed of model, $V_m = 1$ m/s

Force required for model, $F_m = 2$ N

Let the speed of ship $= V_P$

and the propulsive force for ship $= F_P$.

As prototype is subjected to wave resistance only for dynamic similarity, the Froude number should be same for model and prototype. Hence for velocity ratio, for Froude model law using equation (12.20), we have

$\therefore$ $$\frac{V_P}{V_m} = \sqrt{L_r} = \sqrt{50}$$

$\therefore$ $V_P = \sqrt{50} \times V_m = \sqrt{50} \times 1$ = **7.071 m/s. Ans.**

Force scale ratio is given by equation (12.24),

$\therefore$ $$F_r = \frac{F_P}{F_m} = L_r^3$$

$\therefore$ $F_P = F_m \times L_r^3 = 2 \times (50)^3$ = **250000 N. Ans.**

Problem 12.21 *In the model test of a spillway the discharge and velocity of flow over the model were 2 m^3/s and 1.5 m/s respectively. Calculate the velocity and discharge over the prototype which is 36 times the model size.*

Solution. Given :

Discharge over model, $Q_m = 2\ m^3/s$

Velocity over model, $V_m = 1.5$ m/s

Linear scale ratio, $L_r = 36$.

For dynamic similarity, Froude model law is used. Using equation (12.20), we have

$$\frac{V_P}{V_m} = \sqrt{L_r} = \sqrt{36} = 6.0$$

$\therefore$ V_P = Velocity over prototype = $V_m \times 6.0 = 1.5 \times 6.0 =$ **9 m/s. Ans.**

For discharge, using equation (12.23), we get

$$\frac{Q_P}{Q_m} = L_r^{2.5} = (36)^{2.5}.$$

$\therefore$ $Q_P = Q_m \times (36)^{2.5} = 2 \times 36^{2.5} =$ **15552 m^3/s. Ans.**

Problem 12.22 *In a geometrically similar model of spillway the discharge per metre length is $\frac{1}{6}$ m^3/s. If the scale of the model is $\frac{1}{36}$, find the discharge per metre length of the prototype.*

Solution. Given :

Discharge per metre length for model, $q_m = \frac{1}{6}\ m^3/s$

Linear scale ratio, $L_r = 36$

Discharge per metre length for prototype, $q_P = ?$

The discharge ratio for spillway is given by equation (12.23), $\frac{Q_P}{Q_m} = L_r^{2.5}$.

But discharge ratio per metre length is given as

$$\frac{q_P}{q_m} = \frac{Q_P / L_P}{Q_m / L_m} = \frac{Q_P}{Q_m} \times \frac{L_m}{L_P} = L_r^{2.5} \times \frac{1}{L_r} = L_r^{1.5}$$

$\therefore$ $$q_P = q_m \times L_r^{1.5} = \frac{1}{6} \times (36)^{1.5} = \frac{1}{6} \times 6^{2 \times 1.5}$$

$$= 6^{3-1} = 6^2 =$$ **36 m^3/s per metre length. Ans.**

Problem 12.23 *A spillway model is to be built to a geometrically similar scale of $\frac{1}{50}$ across a flume of 600 mm width. The prototype is 15 m high and maximum head on it is expected to be 1.5 m. (i) What height of model and what head on the model should be used ? (ii) If the flow over the model at a particular head is 12 litres per second, what flow per metre length of the prototype is expected ? (iii) If the negative pressure in the model is 200 mm, what is the negative pressure in prototype ? Is it practicable ?*

Solution. Given :

Scale ratio for length, $L_r = 50$

Width of model, $B_m = 600$ mm = 0.6 m

Flow over model, $Q_m = 12$ litres/s
Pressure in model, $h_m = -200$ mm of water $= -0.2$ m
Height of prototype, $H_P = 15$ m
Head on prototype, $H_P^* = 1.5$ m
(*i*) Let the height of model $= H_m$
and head on model $= H_m^*$

Linear scale ratio, $$L_r = \frac{H_P}{H_m} = \frac{H_{P^*}}{H_{m^*}} = 50$$

$\therefore$ Height of model, $$H_m = \frac{H_P}{50} = \frac{15}{50} = \textbf{0.3 m. Ans.}$$

And head on model, $$H_m^* = \frac{H_{P^*}}{50} = \frac{1.50}{50} = \textbf{0.03 m. Ans.}$$

Width of prototype, $B_P = L_r \times B_m = 50 \times 0.6 = 30$ m.
(*ii*) Discharge ratio is given by equation (12.23) as

$$\frac{Q_P}{Q_m} = L_r^{2.5} = (50)^{2.5} = 17677.67$$

$\therefore$ $$Q_P = Q_m \times 17677.67 = 12 \times 17677.67 = 212132.04 \text{ lit/s}$$

Discharge per metre length of prototype $= \dfrac{Q_P}{\text{Length of prototype}} = \dfrac{212132.04}{\text{Width of prototype}}$

$$= \frac{212132.04}{30} = \textbf{7071.078 litres/s. Ans.}$$

(*iii*) Negative pressure head in prototype,

$$h_P = L_r \times h_m = 50 \times (-0.2) = \textbf{– 10.0 m. Ans.}$$

This negative pressure is not practicable. Maximum practicable negative pressure head is – 7.50 m of water.

Problem 12.24 *In a 1 in 20 model of stilling basin, the height of the hydraulic jump in the model is observed to be 0.20 metre. What is the height of the hydraulic jump in the prototype ? If the energy dissipated in the model is* $\frac{1}{10}$ *kW, what is the corresponding value in prototype ?*

Solution. Given :
Linear scale ratio, $L_r = 20$
Height of hydraulic jump in model, $h_m = 0.20$ m
Energy dissipated in model, $P_m = \dfrac{1}{10}$ kW
(*i*) Let the height of hydraulic jump in the prototype $= h_P$
Then $$\frac{h_P}{h_m} = L_r = 20$$

$\therefore$ $$h_P = h_m \times 20 = 0.20 \times 20 = \textbf{4 m. Ans.}$$

(*ii*) Let the energy dissipated in prototype $= F_P$
Using equation (12.27) for power ratio, $\dfrac{P_P}{P_m} = L_r^{3.5} = 20^{3.5} = 35777.088$

$\therefore$ $$P_P = P_m \times 35777.088 = \frac{1}{10} \times 35777.088 = \textbf{3577.708 kW. Ans.}$$

Problem 12.25 *The characteristics of the spillway are to be studied by means of a geometrically similar model constructed to the scale ratio of 1 : 10.*

(i) If the maximum rate of flow in the prototype is 28.3 cumecs, what will be the corresponding flow in model ?

(ii) If the measured velocity in the model at a point on the spillway is 2.4 m/s, what will be the corresponding velocity in prototype ?

(iii) If the hydraulic jump at the foot of the model is 50 mm high, what will be the height of jump in prototype ?

(iv) If the energy dissipated per second in the model is 3.5 Nm, what energy is dissipated per second in the prototype ?

Solution. Given :

$$\frac{\text{Linear dimension of model}}{\text{Linear dimension of prototype}} = \frac{1}{10}$$

$\therefore$ Scale ratio, $L_r = 10$.

(*i*) Discharge in prototype, $Q_P = 28.3\ m^3/s$

Let Q_m = Discharge in model

For discharge using equation (12.23), we get

$$\frac{Q_P}{Q_m} = L_r^{2.5}$$

$$\therefore \quad Q_m = \frac{Q_P}{L_r^{2.5}} = \frac{28.3}{10^{2.5}} = \mathbf{0.0895\ m^3/s.\ Ans.}$$

(*ii*) Velocity in the model, $V_m = 2.4$ m/s

Let V_P = Velocity in the prototype

For velocity using equation (12.20), we get

$$\frac{V_P}{V_m} = \sqrt{L_r}$$

$$\therefore \quad V_P = V_m \times \sqrt{L_r} = 2.4 \times \sqrt{10} = \mathbf{7.589\ m/s.\ Ans.}$$

(*iii*) Hydraulic jump in model, $H_m = 50$ mm

Let H_P = Hydraulic jump in prototype

Now scale ratio $= \dfrac{H_P}{H_m}$

$$\therefore \quad H_P = H_m \times \text{Scale ratio} = 50 \times 10 = \mathbf{500\ mm.\ Ans.}$$

(*iv*) Energy dissipated/s in model, $E_m = 3.5$ N m/s

Let E_P = Energy dissipated/s in prototype

Now using equation (12.27), we get $\dfrac{E_P}{E_m} = L_r^{3.5}$

$$\therefore \quad E_P = E_m \times L_r^{3.5} = 3.5 \times 10^{3.5} = \mathbf{11067.9\ N\ m/s.\ Ans.}$$

Problem 12.26 *A 1 : 64 model is constructed of an open channel in concrete which has Manning's N = 0.014. Find the value of N for the model.*

Solution. Given :

Linear scale ratio, $L_r = 64$

Value of N for prototype, $N_P = 0.014$

Let N_m = Value of N for model.

The Manning's formula* is given by, $V = \frac{1}{N} m^{3/2} \cdot i^{1/2}$

in which m = Hydraulic mean depth in m

i = Slope of the bed of the channel

Now for the model, the Manning's formula becomes as

$$V_m = \frac{1}{N_m} \cdot (m_m)^{2/3} \cdot (i_m)^{1/2} \qquad ...(i)$$

and for the prototype, the Manning's formula is written as

$$V_P = \frac{1}{N_P} \cdot (m_p)^{2/3} \cdot (i_p)^{1/2} \qquad ...(ii)$$

Dividing equation (*ii*) by equation (*i*), we get

$$\frac{V_P}{V_m} = \frac{\frac{1}{N_P} \cdot (m_P)^{2/3} \cdot (i_P)^{1/2}}{\frac{1}{N_m} \cdot (m_m)^{2/3} \cdot (i_m)^{1/2}} = \frac{N_m}{N_P} \cdot \left(\frac{m_P}{m_m}\right)^{2/3} \cdot \left(\frac{i_P}{i_m}\right)^{1/2} \qquad ...(iii)$$

For dynamic similarity, Froude model law is used. Using equation (12.20), we have

$$\frac{V_P}{V_m} = \sqrt{L_r} = \sqrt{64} = 8$$

But $\frac{m_p}{m_m} = L_r$ and $\frac{i_p}{i_m} = 1$ as i_p and i_m are dimensionless.

Substituting these values in equation (*iii*), we get

$$8 = \frac{N_m}{N_P} \times (L_r)^{2/3} \times 1 = \frac{N_m}{0.014} \times (64)^{2/3} \qquad (\because N_P = 0.014)$$

$$\therefore \qquad N_m = \frac{8 \times 0.014}{64^{2/3}} = \frac{8 \times 0.014}{16} = \mathbf{0.007.\ Ans.}$$

Problem 12.27 *A 7.2 m height and 15 m long spillway discharges 94 m^3/s discharge under a head of 2.0 m. If a 1 : 9 scale model of this spillway is to be constructed, determine model dimensions, head over spillway model and the model discharge. If model experiences a force of 7500 N (764.53 kgf), determine force on the prototype.*

Solution. Given :

For prototype : Height $h_P = 7.2$ m

Length, $L_P = 15$ m

Discharge, $Q_P = 94\ m^3/s$

*See chapter 16 where Manning's formula for velocity through an open channel flow is given.

Head, $H_P = 2.0$ m

Size of model = $\frac{1}{9}$ of the size of prototype.

$\therefore$ Linear scale ratio, $L_r = 9$

Force experienced by model, $F_P = 7500$ N

Find : (*i*) Model dimensions *i.e.*, height and length of model (h_m and L_m)

(*ii*) Head over model *i.e.*, H_m

(*iii*) Discharge through model *i.e.*, Q_m

(*iv*) Force on prototype (*i.e.*, F_p)

(*i*) *Model dimensions (h_m and L_m)*

$$\frac{h_P}{h_m} = \frac{L_P}{L_m} = L_r = 9$$

$\therefore$ $$h_m = \frac{h_P}{9} = \frac{7.2}{9} = \textbf{0.8 m. Ans.}$$

And $$L_m = \frac{L_P}{9} = \frac{15}{9} = \textbf{1.67 m. Ans.}$$

(*ii*) *Head over model (H_m)*

$$\frac{H_P}{H_m} = L_r = 9$$

$\therefore$ $$H_m = \frac{H_P}{9} = \frac{2}{9} = \textbf{0.222 m. Ans.}$$

(*iii*) *Discharge through model (Q_m)*

Using equation (12.23), we get $\frac{Q_P}{Q_m} = L_r^{2.5}$

$\therefore$ $$Q_m = \frac{Q_P}{L_r^{2.5}} = \frac{94}{9^{2.5}} = \frac{94}{243} = \textbf{0.387 m}^3\textbf{/s. Ans.}$$

(*iv*) *Force on the Prototype (F_P)*

Using equation (12.24), we get $F_r = \frac{F_P}{F_m} = L_r^3$

$\therefore$ $$F_P = F_m \times L_r^3 = 7500 \times 9^3 = \textbf{5467500 N. Ans.}$$

12.9.3 Euler's Model Law. Euler's model law is the law in which the models are designed on Euler's number which means for dynamic similarity between the model and prototype, the Euler number for model and prototype should be equal. Euler's model law is applicable when the pressure forces are alone predominant in addition to the inertia force. According to this law :

$$(E_u)_{\text{model}} = (E_u)_{\text{prototype}} \qquad ...(12.28)$$

If V_m = Velocity of fluid in model,

p_m = Pressure of fluid in model,

ρ_m = Density of fluid in model,

and V_P, p_P, ρ_P = Corresponding values in prototype, then

Substituting these values in equation (12.28), we get

$$\frac{V_m}{\sqrt{p_m/\rho_m}} = \frac{V_P}{\sqrt{p_P/\rho_P}} \qquad ...(12.29)$$

If fluid is same in model and prototype, then equation (12.29) becomes as

$$\frac{V_m}{\sqrt{p_m}} = \frac{V_P}{\sqrt{p_P}} \qquad ...(12.30)$$

Euler's model law is applied for fluid flow problems where flow is taking place in a closed pipe in which case turbulence is fully developed so that viscous forces are negligible and gravity force and surface tension force is absent. This law is also used where the phenomenon of cavitation takes place.

12.9.4 Weber Model Law. Weber model law is the law in which models are based on Weber's number, which is the ratio of the square root of inertia force to surface tension force. Hence where surface tension effects predominate in addition to inertia force, the dynamic similarity between the model and prototype is obtained by equating the Weber number of the model and its prototype. Hence according to this law :

$$(W_e)_{\text{model}} = (W_e)_{\text{prototype,}} \qquad \text{where } W_e \text{ is Weber number and} = \frac{V}{\sqrt{\sigma/\rho L}}$$

If V_m = Velocity of fluid in model,
σ_m = Surface tensile force in model,
ρ_m = Density of fluid in model,
L_m = Length of surface in model,
and V_P, σ_P, ρ_P, L_P = Corresponding values of fluid in prototype.

Then according to Weber law, we have

$$\frac{V_m}{\sqrt{\sigma_m/\rho_m L_m}} = \frac{V}{\sqrt{\sigma_P/\rho_P L_P}} \qquad ...(12.31)$$

Weber model law is applied in following cases :

1. Capillary rise in narrow passages,
2. Capillary movement of water in soil,
3. Capillary waves in channels,
4. Flow over weirs for small heads.

12.9.5 Mach Model Law. Mach model law is the law in which models are designed on Mach number, which is the ratio of the square root of inertia force to elastic force of a fluid. Hence where the forces due to elastic compression predominate in addition to inertia force, the dynamic similarity between the model and its prototype is obtained by equating the Mach number of the model and its prototype. Hence according to this law :

$$(M)_{\text{model}} = (M)_{\text{prototype}}$$

where M = Mach number $= \frac{V}{\sqrt{K/\rho}}$

If V_m = Velocity of fluid in model,
K_m = Elastic stress for model,
ρ_m = Density of fluid in model,
and V_P, K_P and ρ_P = Corresponding values for prototype. Then according to Mach law,

$$= \frac{V_m}{\sqrt{K_m/\rho_m}} = \frac{V}{\sqrt{K_P/\rho_P}} \qquad ...(12.32)$$

Mach model law is applied in the following cases :

1. Flow of aeroplane and projectile through air at supersonic speed, *i.e.*, at a velocity more than the velocity of sound,
2. Aerodynamic testing,
3. Under water testing of torpedoes,
4. Water-hammer problems.

Problem 12.28 *The pressure drop in an aeroplane model of size* $\frac{1}{10}$ *of its prototype is 80 N/cm*^{2}. *The model is tested in water. Find the corresponding pressure drop in the prototype. Take density of air = 1.24 kg/m*^{3}. *The viscosity of water is 0.01 poise while the viscosity of air is 0.00018 poise.*

Solution. Given :

Pressure drop in model, $p_m = 80 \text{ N/cm}^2 = 80 \times 10^4 \text{ N/m}^2$

Linear scale ratio, $L_r = 40$

Fluid in model = Water, while in prototype = Air

Viscosity of water, $\mu_m = 0.01$ poise

Density of water, $\rho_m = 1000 \text{ kg/m}^3$

Viscosity of air, $\mu_P = .00018$ poise

Density of air, $\rho_P = 1.24 \text{ kg/m}^3$

Let the corresponding pressure drop in prototype = p_P.

As the problem involves pressure force and viscous force and hence for dynamic similarity between the model and prototype, Euler's number and Reynold's number should be considered. Making first of all, Reynold's number equal, we get from equation (12.17)

$$\frac{\rho_m V_m L_m}{\mu_m} = \frac{\rho_P V_P L_P}{\mu_P} \quad \text{or} \quad \frac{V_m}{V_P} = \frac{\rho_P}{\rho_m} \times \frac{L_P}{L_m} \times \frac{\mu_m}{\mu_P}$$

But $$\frac{\rho_P}{\rho_m} = \frac{1.24}{1000}$$

$$\frac{L_P}{L_m} = L_r = 40, \quad \frac{\mu_m}{\mu_P} = \frac{0.01}{.00018}$$

$\therefore$ $$\frac{V_m}{V_P} = \frac{1.24}{1000} \times 40 \times \frac{.01}{.00018} = 2.755.$$

Now making Euler's number equal, we get from equation (12.29) as

$$\frac{V_m}{\sqrt{\frac{p_m}{\rho_m}}} = \frac{V_P}{\sqrt{\frac{p_P}{\rho_P}}} \quad \text{or} \quad \frac{V_m}{V_P} = \frac{\sqrt{p_m/\rho_m}}{\sqrt{p_P/\rho_P}} = \sqrt{\frac{p_m}{p_P}} \times \sqrt{\frac{\rho_P}{\rho_m}}$$

But $$\frac{V_m}{V_P} = 2.755 \text{ and } \frac{\rho_P}{\rho_m} = \frac{1.24}{1000}$$

$\therefore$ $$2.755 = \sqrt{\frac{p_m}{p_P}} \times \sqrt{\frac{1.24}{1000}} = \sqrt{\frac{p_m}{p_P}} \times .0352$$

$\therefore$ $$\sqrt{\frac{p_m}{p_P}} = \frac{2.755}{.0352} = 78.267$$

$\therefore$ $$\frac{p_m}{p_P} = (78.267)^2 \text{ or } p_P = \frac{p_m}{(78.267)^2} = \frac{80}{(78.267)^2}$$

$$= \mathbf{0.01306 \text{ N/cm}^2}. \textbf{ Ans.}$$

▶ 12.10 MODEL TESTING OF PARTIALLY SUB-MERGED BODIES

Let us consider the testing of a ship model (ship is a partially sub-merged body) in a water-tunnel in order to find the drag force F or resistance experienced by a ship. The drag experienced by a ship consists of :

1. The wave resistance, which is the resistance offered by the waves on the free sea-surface, and
2. The frictional or viscous resistance, which is offered by the water on the surface of contact of the ship with water.

Thus in this case three forces namely inertia, gravity and viscous forces are present. Then for dynamic similarity between the model and its prototype, the Reynold's number (which is ratio of inertia force to viscous force) and the Froude number (which is the ratio of inertia force to gravity force) should be taken into account. This means that in this case, the Reynold model law and Froude model law should be applied.

But for Reynold model law, the condition is

Reynold number of model= Reynold number of prototype

or
$$\frac{\rho_m V_m L_m}{\mu_m} = \frac{\rho_P V_P L_P}{\mu_P}$$

If fluid is same for the model and prototype, then $\rho_m = \rho_P$ and $\mu_m = \mu_P$

$\therefore$
$$V_m L_m = V_P L_P$$

$$V_m = \frac{V_P L_P}{L_m} = L_r V_P \qquad \left\{\because \frac{L_P}{L_m} = L_r\right\} \quad ...(12.33)$$

For Froude model law, have from equation (12.18) as $\frac{V_m}{\sqrt{g_m L_m}} = \frac{V_P}{\sqrt{g_P L_P}}$

If fluid is same for model and prototype and test is conducted at the same place where prototype is to operate, then $g_m = g_P$

$\therefore$
$$\frac{V_m}{\sqrt{L_m}} = \frac{V_P}{\sqrt{L_P}}$$

$\therefore$
$$V_m = \sqrt{\frac{L_m}{L_P}} \times V_P = V_P \times \frac{1}{\sqrt{\frac{L_P}{L_m}}} = V_P \times \frac{1}{\sqrt{L_r}} \quad \left\{\because \frac{L_P}{L_m} = L_r\right\} \quad ...(12.34)$$

From equations (12.33) and (12.34), we observe that the velocity of fluid in model for Reynold model law and Froude model law is different. Thus it is quite impossible to satisfy both the laws together, which means the dynamic similarity between the model and its prototype will not exist. To overcome this difficulty, the method suggested by William Froude is adopted for testing the ship model (or partially sub-merged bodies) as :

Step 1. The total resistance experienced by a ship is equal to the wave resistance plus frictional or viscous resistance.

Let
$(R)_P$ = Total resistance experienced by prototype,
$(R_w)_P$ = Wave resistance experienced by prototype,
$(R_f)_P$ = Frictional resistance experienced by prototype, and

$(R)_m$, $(R_w)_m$, $(R_f)_m$ = Corresponding values for model.

Then, we have for prototype, $(R)_P = (R_w)_P + (R_f)_P$ ...(12.35)

and for model, $(R)_m = (R_w)_m + (R_f)_m$ ...(12.36)

Step 2. The frictional resistances for the model and the ship [*i.e.*, $(R_f)_m$ and $(R_f)_P$] are calculated from the expressions given below :

$$(R_f)_P = f_P A_P V_P^n \quad ...(12.37)$$

and $$(R_f)_m = f_m A_m V_m^n \quad ...(12.38)$$

where f_P = Frictional resistance per unit area per unit velocity of prototype,

A_P = Wetted surface area of the prototype,

V_P = Velocity of prototype,

n = Constant, and

f_m, A_m, V_m = Corresponding values of frictional resistance, wetted area and velocity of model.

The values of f_P and f_m are determined from experiments.

Step 3. The model is tested by towing it in water contained in a towing tank such that the dynamic similarity for Froude number is satisfied *i.e.*, $(F_e)_m = (F_e)_P$. The total resistance of the model (R_m) is measured for this condition.

Step 4. The total resistance (R_m) for the model is known from step 3 and frictional resistance of the model $(R_f)_m$ is calculated from equation (12.37). Then the wave resistance for the model is known from equation (12.36) as

$$(R_w)_m = R_m - (R_f)_m \quad ...(12.39)$$

Step 5. The resistance experienced by a ship of length L, flowing with velocity V in fluid of viscosity μ, density ρ depends upon g, the acceleration due to gravity. By dimensional analysis, the expression for resistance is given by

$$\frac{R}{\rho L^2 V^2} = \phi\left[\frac{\rho VL}{\mu}, \frac{V^2}{gL}\right] = \phi\,[R_e, F_e^2]$$

Thus resistance is a function of Reynold number (R_e) and Froude number (F_e).

For dynamic similarity for model and prototype for wave resistance only, we have

$$\frac{(R_w)_P}{\rho_P L_P^2 V_P^2} = \frac{(R_w)_m}{\rho_m L_m^2 V_m^2}$$

or wave resistance for prototype is given as

$$(R_w)_P = \frac{\rho_P}{\rho_m} \times \frac{L_P^2}{L_m^2} \times \frac{V_P^2}{V_m^2} \times (R_w)_m \quad ...(12.40)$$

But from Step 3, $(F_e)_m = (F_e)_P$ or $\dfrac{V_m}{\sqrt{L_m g_m}} = \dfrac{V_P}{\sqrt{L_P g_P}}$

If the model and ship are at the same place, $g_m = g_P$

$\therefore$ $$\frac{V_m}{\sqrt{L_m}} = \frac{V_P}{\sqrt{L_P}} \quad \text{or} \quad V_m = \sqrt{\frac{L_m}{L_P}} \cdot V_P$$

Substituting the value of V_m in equation (12.40), we have

$$(R_w)_P = \frac{\rho_P}{\rho_m} \times \frac{L_P^2}{L_m^2} \times \frac{V_P^2}{V_P^2 \times \dfrac{L_m}{L_P}} \times (R_w)_m$$

$$= \frac{\rho_P}{\rho_m} \times \frac{L_P^3}{L_m^3} \times (R_w)_m. \qquad ...(12.41)$$

Step 6. The total resistance of the ship is given by adding $(R_w)_P$ from equation (12.41) to $(R_f)_P$ given by equation (12.37) as

$$R_P = \frac{\rho_P}{\rho_m} \times \left(\frac{L_P}{L_m}\right)^3 \times (R_w)_m + f_P A_P V_P^2. \qquad ...(12.42)$$

Problem 12.29 *A 1 in 20 model of a naval ship having a sub-merged surface area of 5 m^2 and length 8 m has a total drag of 20 N when towed through water at a velocity of 1.5 m/s. Calculate the total drag on the prototype when moving at the corresponding speed. Use the relation* $F_f = \frac{1}{2} C_f \rho A V^2$ *for calculating the skin (frictional) resistance. The value of* C_f *is given by* $C_f = \frac{0.0735}{(R_e)^{1.5}}$.

Take kinematic viscosity of water (or sea-water) as 0.01 stoke and density of water (or sea-water) as 1000 kg/m^3.

Solution. Given :

Linear scale ratio, $L_r = 20$

Sub-merged area of model, $A_m = 5.0\ m^2$

Length of model, $L_m = 8.0$ m

Total drag of model, $R_m = 20$ N

Velocity of model, $V_m = 1.5$ m/s

Let A_P, L_P, R_P, V_P = Corresponding values for prototype.

Fluid in model is the same as in prototype and is sea-water.

Kinematic viscosity of sea-water, $\nu_m = \nu_P = 0.01$ stokes $= .01\ cm^2/s = .01 \times 10^{-4}\ m^2/s$

Density of water, $\rho_m = 1000\ kg/m^3$

The skin (frictional) resistance of model is given by

$$(F_f)_m = \frac{1}{2} C_{f_m} \rho_m A_m V_m^2 \qquad ...(i)$$

where $$C_{f_m} = \frac{0.0735}{\left[(R_e)_m\right]^{1/5}} \qquad ...(ii)$$

where $(R_e)_m$ = Reynold's number for model

$$= \frac{\rho_m V_m L_m}{\mu_m} \text{ or } \frac{V_m L_m}{\nu_m} \qquad \left\{\because \nu = \frac{\mu}{\rho}\right\}$$

$$= \frac{1.5 \times 8.0}{.01 \times 10^{-4}} = 1.2 \times 10^7.$$

Substituting this value in equation (*ii*), we get

$$C_{f_m} = \frac{0.0735}{(1.2 \times 10^7)^{1/5}} = \frac{.0735}{26.0517} = 2.82 \times 10^{-3} \qquad ...(iii)$$

Substituting the value of C_{f_m} in equation (*i*), we get

$$(F_f)_m = \frac{1}{2} \times 2.82 \times 10^{-3} \times 1000 \times 5.0 \times (1.5)^2 = 15.8617 = 15.862 \text{ N}$$

Using equation (12.36), we get $R_m = (R_w)_m + (R_f)_m$

where $(R_f)_m = (F_f)_m = 15.862$ or $20 = (R_w)_m + 15.862$

$\therefore$ Wave resistance for model, $(R_w)_m = 20 - 15.862 = 4.138$ N ...(*iv*)

The wave resistance experienced by the ship is given by equation (12.41) as

$$(R_w)_P = \frac{\rho_P}{\rho_m} \times \left(\frac{L_P}{L_m}\right)^3 \times (R_w)_m$$

$$= 1 \times L_r^3 \times 4.138 \text{ N} \qquad \left\{\because \frac{\rho_P}{\rho_m} = 1 \text{ for same fluid}\right\}$$

$$= 1 \times 20^3 \times 4.138 = 33104 \text{ N}$$

and skin (frictional) resistance of prototype is given by

$$(R_f)_P = (F_f)_P = \frac{1}{2} C_{f_P} \times \rho_P \times A_P \times V_P^2 \qquad ...(v)$$

where V_P is the velocity of prototype and is given by Froude model law,

i.e., $$(F_e)_m = (F_e)_P \quad \text{or} \quad \frac{V_m}{\sqrt{L_m g}} = \frac{V}{\sqrt{L_P g}} \quad \text{or} \quad \frac{V_m}{\sqrt{L_m}} = \frac{V_P}{\sqrt{L_P}}$$

$\therefore$ $$V_P = \sqrt{\frac{L_P}{L_m}} \times V_m = \sqrt{L_r} \times V_m \qquad \left\{\because \frac{L_P}{L_m} = L_r\right\}$$

$$= \sqrt{20} \times 1.5 = 6.708 \text{ m/s}$$

Now $$\frac{A_P}{A_m} = L_r^2 = 20^2$$

$\therefore$ $$A_P = A_m \times 20^2 = 5 \times 400 = 2000 \text{ m}^2$$

and $$L_P = L_m \times L_r = 8 \times 20 = 160 \text{ m}$$

In equation (*v*), the value of C_{f_P} is given by $C_{f_P} = \dfrac{0.0735}{\left[(R_e)_P\right]^{1/5}}$

where $(R_e)_P$ = Reynolds number for prototype

$$= \frac{V_P \times L_P}{\nu_P} = \frac{6.708 \times 160}{.01 \times 10^{-4}} = 1.073 \times 10^9$$

$\therefore$ $$C_{f_P} = \frac{0.0735}{(1.073 \times 10^9)^{1/5}} = \frac{0.0735}{63.99} = 1.1486 \times 10^{-3}$$

Substituting this value of C_{f_P} in equation (*v*), we get

$$(R_f)_P = (F_f)_P = \frac{1}{2} \times 1.1486 \times 10^{-3} \times 1000 \times 2000 \times (6.708)^2 = 51683.8 \text{ N}$$

$\therefore$ Total drag on prototype is obtained by using equation (12.35).

$\therefore$ $$R_P = (R_w)_P + (R_f)_P = 33104 + 51683.8 = \mathbf{84787.8\ N.\ Ans.}$$

Problem 12.30 *A 1 : 15 model of a flying boat is towed through water. The prototype is moving in sea-water of density 1024 kg/m^3 at a velocity of 20 m/s. Find the corresponding speed of the model. Also determine the resistance due to waves on model if the resistance due to waves of prototype is 600 N.*

Solution. Given :

Linear scale ratio, $L_r = 15$

Velocity of prototype, $V_p = 20$ m/s

Fluid in prototype is sea-water while in model it is water

Density of sea-water, $\rho_P = 1024$ kg/m^3

Density of water, $\rho_m = 1000$ kg/m^3

Resistance due to waves for prototype is, $(R_w)_P = 600$ N.

Find V_m and $(R_w)_m$.

(*i*) The velocity, V_m from model is given by Froude model law,

$\therefore$ $$\frac{V_m}{\sqrt{L_m g}} = \frac{V_P}{\sqrt{L_P \times g}}$$

$\therefore$ $$V_m = \sqrt{\frac{L_m}{L_P}} \times V_P = \frac{V_P}{\sqrt{L_P / L_m}} = \frac{20}{\sqrt{15}} \qquad \left\{\because \frac{L_P}{L_m} = L_r = 15\right\}$$

$$= \frac{20}{3.872} = \mathbf{5.165\ m/s.\ Ans.}$$

(*ii*) For dynamic similarity between model and its prototype for wave resistance only, we have equation (12.41) as

$$(R_w)_P = \frac{\rho_P}{\rho_m} \times \left(\frac{L_P}{L_m}\right)^3 \times (R_w)_m$$

Substituting the known values, $600 = \frac{1024}{1000} \times L_r^{\,3} \times (R_w)_m = \frac{1024}{1000} \times 15^3 \times (R_w)_m$

$\therefore$ $$(R_w)_m = \frac{600 \times 1000}{1024 \times 15^3} = \mathbf{0.1736\ N.\ Ans.}$$

Problem 12.31 *A 1 : 40 model of an ocean tanker is dragged through fresh water at 2 m/s with a total measured drag of 12 N. The skin (frictional) drag co-efficient 'f' for model and prototype are 0.03 and 0.002 respectively in the equation $R_f = f \,.\, AV^2$. The wetted surface area of the model is 25 m^2. Determine the total drag on the prototype and the power required to drive the prototype.*

Take $\rho_P = 1030$ kg/m^3 and $\rho_m = 1000$ kg/m^3.

Solution. Given :

Linear scale ratio, $L_r = 40$

Velocity of model, $V_m = 2$ m/s
Total drag of model, $R_m = 12$ N
Wetted area of model, $A_m = 25$ m^2
Co-efficient of friction for model, $f_m = .03$
for prototype, $f_P = .002$.
Let the total drag on prototype $= R_P$
And power required to drive the prototype $= P$
Frictional drag on model, $(R_f)_m = f_m A_m V_m^2 = .03 \times 25 \times 2^2 = 3$ N
$\therefore$ Wave drag on model, $(R_w)_m = R_m - (R_f)_m = 12 - 3 = 9$ N.
The waves drag on prototype is obtained from equation (12.41) as

$$(R_w)_P = \frac{\rho_P}{\rho_m} \times \left(\frac{L_P}{L_m}\right)^3 \times (R_w)_m = \frac{1030}{1000} \times L_r^3 \times 9 \qquad \left\{\because \frac{L_P}{L_m} = L_r = 40\right\}$$

$$= \frac{1030}{1000} \times 40^3 \times 9 = 593291.8 \text{ N} \qquad ...(i)$$

The frictional drag on prototype is given by

$$(R_f)_P = f_P \times A_P \times V_P^2 \qquad ...(ii)$$

where the velocity of prototype V_P is obtained from Froude model law as

$$\frac{V_m}{\sqrt{L_m \times g}} = \frac{V_P}{\sqrt{L_P \times g}} \quad \text{or} \quad \frac{V_m}{\sqrt{L_m}} = \frac{V_P}{\sqrt{L_P}}$$

$$\therefore \qquad V_p = \sqrt{\frac{L_P}{L_m}} \times V_m = \sqrt{L_r} \times V_m = \sqrt{40} \times 2 = 12.65 \text{ m/s}$$

and $$\frac{A_P}{A_m} = L_r^2 = 40 \times 40 \text{ or } A_P = 40 \times 40 \times A_m$$

$$= 40 \times 40 \times 25 = 40000 \text{ m}^2.$$

Substituting these values in (*ii*), we get

$$(R_f)_P = .002 \times 40000 \times (12.65)^2 = 12801.8 \text{ N} \qquad ...(iii)$$

Total drag on the prototype is obtained by adding equations (*i*) and (*ii*) as

$$R_P = (R_w)_P + (R_f)_P$$
$$= 593291.8 + 12801.8 = \textbf{606093.6 N. Ans.}$$

Power required to drive the prototype,

$$P = \frac{(\text{Total drag on prototype}) \times \text{Velocity of prototype}}{1000}$$

$$= \frac{606093.6 \times 12.65}{1000} = \textbf{7667 kW. Ans.}$$

Problem 12.32 *Resistance R, to the motion of a completely sub-merged body is given by*

$$R = \rho V^2 l^2 \, \phi\left(\frac{Vl}{\nu}\right),$$

where ρ and ν are density and kinematic viscosity of the fluid while l is the length of the body and V is the velocity of flow. If the resistance of a one-eight scale air-ship model when tested in water at

12 m/s is 22 N, what will be the resistance in air of the air-ship at the corresponding speed ? Kinematic viscosity of air is 13 times that of water and density of water is 810 times of air.

Solution. Given :

Linear scale ratio, $L_r = 8$

Velocity of model, $V_m = 12$ m/s

Resistance to model, $R_m = 22$ N

The fluid for model is water and for prototype the fluid is air.

Kinematic viscosity of air = 13 × Kinematic viscosity of water

$\therefore \quad \nu_P = 13 \times \nu_m$

Density of water = 810 × Density of air

$\therefore \quad \rho_m = 810 \times \rho_P$

Let V_P = Velocity of the air-ship (Prototype)

R_P = Resistance of the air-ship

The resistance, R, is given by $R = \rho V^2 l^2 \, \phi\left(\frac{Vl}{\nu}\right)$

$\therefore$ The non-dimensional terms $\frac{R}{\rho V^2 l^2}$ and $\frac{Vl}{\nu}$ should be same for the prototype and its model.

$$\therefore \quad \left(\frac{Vl}{\nu}\right)_{\text{prototype}} = \left(\frac{Vl}{\nu}\right)_{\text{model}} \quad \text{or} \quad \frac{V_P l_P}{\nu_P} = \frac{V_m l_m}{\nu_m}$$

$$\therefore \quad V_P = V_m \frac{l_m}{l_P} \times \frac{\nu_P}{\nu_m} = 12 \times \frac{1}{L_r} \times 13 \qquad \left\{\because \ \frac{l_P}{l_m} = L_r\right\}$$

$$= 12 \times \frac{1}{8} \times 13 = 19.5 \text{ m/s}$$

Also $$\left(\frac{R}{\rho V^2 l^2}\right)_{\text{prototype}} = \left(\frac{R}{\rho V^2 l^2}\right)_{\text{model}} \quad \text{or} \quad \frac{R_P}{\rho_P V_P^2 l_P^2} = \frac{R_m}{\rho_m V_m^2 l_m^2}$$

$$\therefore \quad R_p = R_m \times \frac{\rho_P}{\rho_m} \times \frac{V_P^2}{V_m^2} \times \frac{l_P^2}{l_m^2}$$

$$= 22 \times \frac{1}{810} \times \frac{(19.5)^2}{12^2} \times 8^2 \qquad \left(\because \ \frac{l_P}{l_m} = l_r = 8.0\right)$$

$$= \mathbf{4.59 \text{ N. Ans.}}$$

▶ 12.11 CLASSIFICATION OF MODELS

The hydraulic models are classified as :

1. Undistorted models, and 2. Distorted models.

12.11.1 Undistorted Models. Undistorted models are those models which are geometrically similar to their prototypes or in other words if the scale ratio for the linear dimensions of the model and its prototype is same, the model is called undistorted model. The behaviour of the prototype can be easily predicted from the results of undistorted model.

12.11.2 Distorted Models. A model is said to be distorted if it is not geometrically similar to its prototype. For a distorted model different scale ratios for the linear dimensions are adopted. For example, in case of rivers, harbours, reservoirs etc., two different scale ratios, one for horizontal dimensions and other for vertical dimensions are taken. Thus the models of rivers, harbours and reservoirs will become as distorted models. If for the river, the horizontal and vertical scale ratios are taken to be same so that the model is undistorted, then the depth of water in the model of the river will be very-very small which may not be measured accurately. The following are the advantage of distorted models :

1. The vertical dimensions of the model can be measured accurately.
2. The cost of the model can be reduced.
3. Turbulent flow in the model can be maintained.

Though there are some advantages of the distorted model, yet the results of the distorted model cannot be directly transferred to its prototype. But sometimes from the distorted models very useful information can be obtained.

12.11.3 Scale Ratios for Distorted Models. As mentioned above, two different scale ratios, one for horizontal dimensions and other for vertical dimensions, are taken for distorted models.

Let $(L_r)_H$ = Scale ratio for horizontal dimension

$$= \frac{L_P}{L_m} = \frac{B_P}{B_m} = \frac{\text{Linear horizontal dimension of prototype}}{\text{Linear horizontal dimension of model}}$$

$(L_r)_V$ = Scale ratio for vertical dimension

$$= \frac{\text{Linear vertical dimension of prototype}}{\text{Linear vertical dimension of model}} = \frac{h_P}{h_m}$$

Then the scale ratios of velocity, area of flow, discharge etc., in terms of $(L_r)_H$ and $(L_r)_V$ can be obtained for distorted models as given below :

1. Scale ratio for velocity

Let V_P = Velocity in prototype

V_m = Velocity in model.

Then $$\frac{V_P}{V_m} = \frac{\sqrt{2gh_P}}{\sqrt{2gh_m}} = \sqrt{\frac{h_P}{h_m}} = \sqrt{(L_r)_V} \qquad \left(\because \frac{h_P}{h_m} = (L_r)_V\right)$$

2. Scale ratio for area of flow

Let A_P = Area of flow in prototype = $B_P \times h_P$

A_m = Area of flow in model = $B_m \times h_m$

$$\therefore \quad \frac{A_P}{A_m} = \frac{B_P \times h_P}{B_m \times h_m} = \frac{B_P}{B_m} \times \frac{h_P}{h_m} = (L_r)_H \times (L_r)_V$$

3. Scale ratio for discharge

Let Q_P = Discharge through prototype = $A_P \times V_P$

Q_m = Discharge through model = $A_m \times V_m$

$$\therefore \quad \frac{Q_P}{Q_m} = \frac{A_P \times V_P}{A_m \times V_m} = (L_r)_H \times (L_r)_V \times \sqrt{(L_r)_V} = (L_r)_H \times [(L_r)_V]^{3/2} \quad \text{...(12.43)}$$

Problem 12.33 *The discharge through a weir is 1.5 m^3/s. Find the discharge through the model of the weir if the horizontal dimension of the model = $\frac{1}{50}$ the horizontal dimension of the prototype and vertical dimension of the model = $\frac{1}{10}$ the vertical dimension of the prototype.*

Solution. Given :

Discharge through weir (prototype), $Q_P = 1.5\ m^3/s$

$$\text{Horizontal dimension of model} = \frac{1}{50} \times \text{Horizontal dimension of prototype}$$

$$\therefore \quad \frac{\text{Horizontal dimension of prototype}}{\text{Horizontal dimension of model}} = 50 \text{ or } (L_r)_H = 50$$

$$\text{Vertical dimension of model} = \frac{1}{10} \times \text{Vertical dimension of prototype}$$

$$\therefore \quad \frac{\text{Vertical dimension of prototype}}{\text{Vertical dimension of model}} = 10$$

$$\therefore \quad (L_r)_V = 10.$$

Using equation (12.43), we get $$\frac{Q_P}{Q_m} = (L_r)_H \times [(L_r)_V]^{3/2} = 50 \times 10^{3/2} = 1581.14$$

$$Q_m = \frac{Q_P}{1581.14} = \frac{1.50}{1581.14} = .000948\ m^3/s$$

$$= \mathbf{0.948\ litres/s.\ Ans.}$$

HIGHLIGHTS

1. Dimensional analysis is the method of dimensions, in which fundamental dimensions are M, L and T.
2. Dimensional analysis is performed by two methods namely Rayleigh's Method and Buckingham's π-theorem.
3. Rayleigh's method is used for finding an expression for a variable which depends on maximum three or four variables while there is no restriction on the number of variables for Buckingham's π-theorem.
4. Model analysis is an experimental method of finding solutions of complex flow problems. A model is a small scale replica of the actual machine or structure. The actual machine or structure is called prototype.
5. Three types of similarities must exist between the model and prototype. They are : (*i*) Geometric Similarity, (*ii*) Kinematic Similarity, and (*iii*) Dynamic Similarity.
6. For geometric similarity, the ratio of all linear dimensions of the model and of the prototype should be equal.
7. Kinematic similarity means the similarity of motion between model and prototype.
8. Dynamic similarity means the similarity of forces between the model and prototype.
9. Reynold's number is defined as the ratio of inertia force and viscous force of a flowing fluid. It is given by,

$$R_e = \frac{\rho VL}{\mu} = \frac{VL}{\nu} = \frac{V \times d}{\nu} \text{ for pipe flow}$$

where V = Velocity of flow, d = Diameter of pipe and
ν = Kinematic viscosity of fluid.

10. Froude's Number is the ratio of the square root of inertia force and gravity force and is given by

$$F_e = \sqrt{\frac{F_i}{F_g}} = \frac{V}{\sqrt{Lg}}.$$

11. Euler's number is the ratio of the square root of inertia force and pressure force and is given by,

$$E_u = \sqrt{\frac{F_i}{F_P}} = \frac{V}{\sqrt{p/\rho}}.$$

12. Mach number is the ratio of the square root of inertia force and elastic force and is given by

$$M = \sqrt{\frac{F_i}{F_e}} = \frac{V}{\sqrt{K/\rho}} = \frac{V}{C}.$$

13. The laws on which the models are designed for dynamic similarity are called model laws or laws of similarity. The model laws are (*i*) Reynold's Model Law, (*ii*) Froude Model Law, (*iii*) Euler Model Law, (*iv*) Weber Model Law, (*v*) Mach Model Law.

14. The drag experienced by a ship model (or partially sub-merged body) is obtained by Froude's method.

15. Hydraulic models are classified as (*i*) undistorted models and (*ii*) distorted models.

16. If the models are geometrically similar to its prototype, the models are known as undistorted model. And if the models are having different scale ratio for horizontal and vertical dimensions, the models are known as distorted model.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define the terms dimensional analysis and model analysis.

2. What do you mean by fundamental units and derived units ? Give examples.

3. Explain the term, 'dimensionally homogeneous equation'.

4. What are the methods of dimensional analysis ? Describe the Rayleigh's method for dimensional analysis.

5. State Buckingham's π-theorem. Why this theorem is considered superior over the Rayleigh's method for dimensional analysis ?

6. What do you mean by repeating variables ? How are the repeating variables selected for dimensional analysis ?

7. Define the terms : model, prototype, model analysis, hydraulic similitude.

8. Explain the different types of hydraulic similarities that must exist between a prototype and its model.

9. What do you mean by dimensionless numbers ? Name any four dimensionless numbers.
Define and explain Reynold's number, Froude's number's and Mach number. Derive expressions for any above two numbers.

10. What is meant by geometric, kinematic and dynamic similarities ? Are these similarities truly attainable ? If not why ?

11. Define the following non-dimensional numbers : Reynold's number, Froude's number and Mach's number. What are their significances for fluid flow problems ?

12. What are the different laws on which models are designed for dynamic similarity ? Where are they used ?

13. How will you determine the total drag of a ship or partially sub-merged bodies ?

14. Explain the terms : distorted models and undistorted models. What is the use of distorted models ?

15. Prove that the scale ratio for discharge for a distorted model is given as

$$\frac{Q_P}{Q_m} = (L_r)_H \times (L_r)_V^{3/2}$$

where Q_P = Discharge through prototype
Q_m = Discharge through model
$(L_r)_H$ = Horizontal scale ratio
$(L_r)_V$ = Vertical scale ratio.

16. Show that ratio of inertia force to viscous force gives the Reynolds number.

17. State Buckingham's π-theorem. What do you mean by repeating variables ? How are the repeating variables selected in dimensional analysis ?

18. What is the significance of the non-dimensional numbers : Reynolds number, Froude number and Mach number in the theory of similarity ? What is the dimensional analysis ? How is this analysis related to the theory of similarity ? *(Delhi University, December 2001)*

19. Define and explain : (*i*) Froude's number, (*ii*) Mach number, (*iii*) Hydraulic similarities (*iv*) Distorted and undistorted models. *(Delhi University, December 2002)*

(B) NUMERICAL PROBLEMS

1. Give the dimensions of : (*i*) Force (*ii*) Viscosity (*iii*) Power and (*iv*) Kinematic viscosity.
[**Ans.** MLT^{-2}, $ML^{-1}T^{-1}$, $ML^{+2}T^{-3}$, L^2T^{-1}]

2. The variables controlling the motion of a floating vessel through water are the drag force F, the speed V, the length L, the density ρ and dynamic viscosity μ of water and acceleration due to gravity g. Derive an expression for F by dimensional analysis. $\left[\textbf{Ans. } F = \rho L^2 V^2 \phi\left[\frac{\mu}{\rho VL}, \frac{Lg}{V^2}\right]\right]$

3. The resistance R, to the motion of a completely sub-merged body depends upon the length of the body L, velocity of flow V, mass density of fluid ρ and kinematic viscosity of fluid ν. By dimensional analysis prove that

$$R = \rho V^2 L^2 \phi\left(\frac{VL}{\nu}\right).$$

4. A pipe of diameter 1.8 m is required to transport an oil of sp. gr. 0.8 and viscosity .04 poise at the rate of 4 m^3/s. Tests were conducted on a 20 cm diameter pipe using water at 20°C. Find the velocity and rate of flow in the model. Viscosity of water at 20°C = .01 poise. [**Ans.** 2.829 m/s, 88.8 litres/s]

5. A model of a sub-marine of scale $\frac{1}{40}$ is tested in a wind tunnel. Find the speed of air in wind tunnel if the speed of sub-marine in sea-water is 15 m/s. Also find the ratio of the resistance between the model and its prototype. Take the values of kinematic viscosities for sea-water and air as .012 stokes and 0.016 stokes respectively. The density of sea-water and of air are given as 1030 kg/m^3 and 1.24 kg/m^3 respectively.
$\left[\textbf{Ans}.\ 800 \text{ m/s}, \frac{F_m}{F_P} = 0.00214\right]$

6. A ship 250 m long moves in sea-water, whose density is 1030 kg/m^3. A 1 : 125 model of this ship is to be tested in wind tunnel. The velocity of air in the wind tunnel around the model is 20 m/s and the resistance of the model is 50 N. Determine the velocity of ship in sea-water and also the resistance of the ship in sea-water. The density of air is given as 1.24 kg/m^3. Take the kinematic viscosity of sea-water and air as 0.012 stokes and 0.018 stokes respectively. [**Ans.** 0.106 m/s, 18228.7 N]

7. In 1 : 30 model of a spillway, the velocity and discharge are 1.5 m/s and 2.0 m^3/s. Find the corresponding velocity and discharge in the prototype. [**Ans.** 8.216 m/s, 9859 m^3/s]

8. A ship-model of scale $\frac{1}{60}$ is towed through sea-water at a speed of 0.5 m/s. A force of 1.5 N is required to tow the model. Determine the speed of the ship and propulsive force on the ship, if prototype is subjected to wave resistance only. [**Ans.** 3.873 m/s, 324000 N]

9. A spillway model is to be built to a geometrically similar scale of $\frac{1}{40}$ across a flume, of 50 cm width. The prototype is 20 m high and maximum head on it is expected to be 2 m. (*i*) What height of model and what head on the model should be used ? (*ii*) If the flow over the model at a particular head is 10 litres/s, what flow per metre length of the prototype is expected ? (*iii*) If the negative pressure in the model is 150 mm, what is the negative pressure in the prototype ? Is it practicable ?

[**Ans.** (*i*) 0.5 m, 0.05 m, (*ii*) 5059.64 litres/s (*iii*) 6 m. Yes]

10. The pressure drop in an aeroplane model of size $\frac{1}{50}$ of its prototype is 4 N/cm^2. The model is tested in water. Find the corresponding pressure drop in the prototype. Take density of air = 1.24 kg/m^3. The viscosity of water is 0.01 poise while the viscosity of air is 0.00018 poise. [**Ans.** .00042 N/cm^2]

11. A 1 : 20 model of a flying boat is towed through water. The prototype is moving in sea-water of density 1024 kg/m^3 at a velocity of 15 m/s. Find the corresponding speed of the model. Also determine the resistance due to waves on model, if the resistance due to waves of prototype is 500 N.

[**Ans.** 3.354 m/s, .061 N]

12. A 1 : 50 model of an ocean tanker is dragged through fresh water at 1.5 m/s with a total measured drag of 10 N. The frictional drag co-efficient '*f*' for model and prototype are 0.03 and 0.02 respectively in the equation,

$$R_f = f \,.\, A \,.\, V^2 \;;$$

the wetted surface area of the model is 20 m^2. Determine the total drag on the prototype and the power required to derive the prototype. Take density of sea-water and of fresh water as 1024 kg/m^3 and 1000 kg/m^2 respectively. [**Ans.** 1118436 N, 158072 h.p.]

13. If model prototype ratio is 1 : 75, show that the ratio of discharges per unit width of spillway is given by $\left(\frac{1}{75}\right)^{3/2}$.

14. A fluid of density ρ and viscosity μ, flows at an average velocity *V* through a circular pipe of diameter *D*. Show by dimensional analysis, that the shear stress at the pipe wall is given as

$$\tau_0 = \rho V^2 \phi \left[\frac{\rho VD}{\mu}\right].$$

15. The drag force exerted by a flowing fluid on a solid body depends upon the length of the body, *L*, velocity of flow *V*, density of fluid ρ, and viscosity μ. Find an expression for drag force using Buckingham's theorem.

16. The efficiency η of geometrically similar fans depends upon the mass density of air ρ, its viscosity μ, speed of fan *N* (revolutions per sec), diameter of blades *D* and discharge *Q*. Perform dimensional analysis.

[**Hint.** Take *D*, *N*, ρ as repeating variables. Then three π-terms will be

$$\pi_1 = D^{a_1} \,.\, N^{b_1} \,.\, \rho^{c_1} \,.\, \eta = \eta \;;$$

$$\pi_2 = D^{a_2} \,.\, N^{b_2} \,.\, \rho^{c_2} \,.\, \mu = \frac{\mu}{D^2 NP} \text{ and } \pi_3 = D^{a_3} \,.\, N^{b_3} \,.\, \rho^{c_3} \,.\, Q = \frac{Q}{D^3 N}$$

$$\therefore \qquad \eta = \phi\,(\pi_2, \pi_3) = \phi\left(\frac{\mu}{D^2 NP}, \frac{Q}{D^3 N}\right) \;\Big]$$

17. The discharge through an orifice depends on the diameter D of the orifice, head H over the orifice, density ρ of liquid, viscosity μ of the liquid and acceleration g due to gravity. Using dimensional analysis, find an expression for the discharge. Hence find the dimensionless parameters on which the discharge co-efficient of an orifice meter depend.

[**Hint.** $Q = f(D, H, \rho, \mu, g)$. Hence $N = 6$, $m = 3$ and number of π-terms = 3. Take ρ, D, g as repeating variables. Then

$$\pi_1 = \rho^{a_1} D^{b_1} g^{c_1} Q, \pi_2 = \rho^{a_2}. D^{b_2} . g^{c_2} . \mu \text{ and } \pi_3 = \rho^{a_3} . D^{b_3} . g^{c_3} . H.$$

Find π_1, π_2 and π_3. They will be $\pi_1 = \dfrac{Q}{D^{2.5} \times g^{1/2}}$, $\pi_2 = \dfrac{\mu}{\rho . D^{3/2} . g^{1/2}}$ and $\pi_3 = \dfrac{H}{D}$.

Hence $$Q = D^{2.5} . g^{1/2} . \left(\frac{H}{D}, \frac{\mu}{\rho . D^{3/2} . g^{1/2}} \right) = D^2 . g^{1/2} . D^{1/2} \left(\frac{H}{D}, \frac{\mu}{\rho . D^{3/2} . g^{1/2}} \right).$$

Also $$Q = C_d \times A \times \sqrt{2gh} .$$

Comparing the two values of Q.

We get $C_d = \phi \left(\dfrac{H}{D}, \dfrac{\mu}{\rho . D^{3/2} . g^{1/2}} \right)$. Hence C_d depends upon $\dfrac{H}{D}$ and $\dfrac{\mu}{\rho . D^{3/2} . g^{1/2}}$]

18. The force exerted by a flowing fluid on a stationary body depends upon the length (L) of the body, velocity (V) of the fluid, density (ρ) of fluid, viscosity (μ) of the fluid and acceleration (g) due to gravity. Find an expression for the force using dimensional analysis.

$$\left[\textbf{Ans.} \; F = \rho L^2 V^2 \phi \left(\frac{\mu}{\rho VL}, \frac{L \times g}{V^2} \right) \right]$$

19. The pressure difference Δp in a pipe of diameter D and length L due to turbulent flow depends upon the velocity V, viscosity μ, density ρ and roughness k. Using Buckingham's π-theorem or otherwise obtain an expression for Δp. *(Delhi University, December 2002)*

13 CHAPTER

BOUNDARY LAYER FLOW

▶ 13.1 INTRODUCTION

When a real fluid flows past a solid body or a solid wall, the fluid particles adhere to the boundary and condition of no slip occurs. This means that the velocity of fluid close to the boundary will be same as that of the boundary. If the boundary is stationary, the velocity of fluid at the boundary will be zero. Farther away from the boundary, the velocity will be higher and as a result of this variation of velocity, the velocity gradient $\frac{du}{dy}$ will exist. The velocity of fluid increases from zero velocity on the stationary boundary to free-stream velocity (U) of the fluid in the direction normal to the boundary. This variation of velocity from zero to free-stream velocity in the direction normal to the boundary takes place in a narrow region in the vicinity of solid boundary. This narrow region of the fluid is called boundary layer. The theory dealing with boundary layer flows is called boundary layer theory.

According to boundary layer theory, the flow of fluid in the neighbourhood of the solid boundary may be divided into two regions as shown in Fig. 13.1.

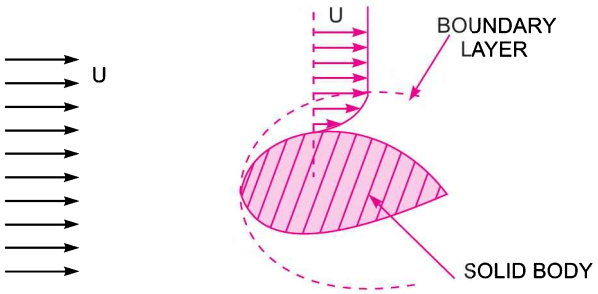


Fig. 13.1 *Flow over solid body.*

1. A very thin layer of the fluid, called the boundary layer, in the immediate neighbourhood of the solid boundary, where the variation of velocity from zero at the solid boundary to free-stream velocity in the direction normal to the boundary takes place. In this region, the velocity gradient $\frac{du}{dy}$ exists and hence the fluid exerts a shear stress on the wall in the direction of motion. The value of shear stress is given by

$$\tau = \mu \frac{du}{dy}.$$

2. The remaining fluid, which is outside the boundary layer. The velocity outside the boundary layer is constant and equal to free-stream velocity. As there is no variation of velocity in this region, the velocity gradient $\frac{du}{dy}$ becomes zero. As a result of this the shear stress is zero.

▶ 13.2 DEFINITIONS

13.2.1 Laminar Boundary Layer. For defining the boundary layer (*i.e.*, laminar boundary layer or turbulent boundary layer) consider the flow of a fluid, having free-stream velocity (U), over a smooth thin plate which is flat and placed parallel to the direction for free stream of fluid as shown in Fig. 13.2. Let us consider the flow with zero pressure gradient on one side of the plate, which is stationary.

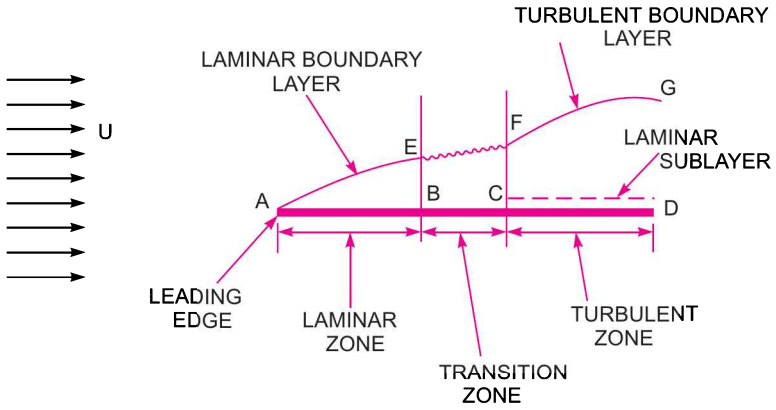


Fig. 13.2 *Flow over a plate.*

The velocity of fluid on the surface of the plate should be equal to the velocity of the plate. But plate is stationary and hence velocity of fluid on the surface of the plate is zero. But at a distance away from the plate, the fluid is having certain velocity. Thus a velocity gradient is set up in the fluid near the surface of the plate. This velocity gradient develops shear resistance, which retards the fluid. Thus the fluid with a uniform free stream velocity (U) is retarded in the vicinity of the solid surface of the plate and the boundary layer region begins at the sharp leading edge. At subsequent points downstream the leading edge, the boundary layer region increases because the retarded fluid is further retarded. This is also referred as the growth of boundary layer. Near the leading edge of the surface of the plate, where the thickness is small, the flow in the boundary layer is laminar though the main flow is turbulent. This layer of the fluid is said to be laminar boundary layer. This is shown by AE in Fig. 13.2. The length of the plate from the leading edge, upto which laminar boundary layer exists, is called laminar zone. This is shown by distance AB. The distance of B from leading edge is obtained from Reynold number equal to 5×10^5 for a plate. Because upto this Reynold number the boundary layer is laminar. The Reynold number is given by $(R_e)_x = \frac{U \times x}{\nu}$

where x = Distance from leading edge,
U = Free-stream velocity of fluid,
ν = Kinematic viscosity of fluid,

Hence for laminar boundary layer, we have $5 \times 10^5 = \dfrac{U \times x}{\nu}$ ...(13.1)

If the values of U and ν are known, x or the distance from the leading edge upto which laminar boundary layer exists can be calculated.

13.2.2 Turbulent Boundary Layer. If the length of the plate is more than the distance x, calculated from equation (13.1), the thickness of boundary layer will go on increasing in the down-stream direction. Then the laminar boundary layer becomes unstable and motion of fluid within it, is disturbed and irregular which leads to a transition from laminar to turbulent boundary layer. This short length over which the boundary layer flow changes from laminar to turbulent is called transition zone. This is shown by distance BC in Fig. 13.2. Further downstream the transition zone, the boundary layer is turbulent and continues to grow in thickness. This layer of boundary is called turbulent boundary layer, which is shown by the portion FG in Fig. 13.2.

13.2.3 Laminar Sub-layer. This is the region in the turbulent boundary layer zone, adjacent to the solid surface of the plate as shown in Fig. 13.2. In this zone, the velocity variation is influenced only by viscous effects. Though the velocity distribution would be a parabolic curve in the laminar sub-layer zone, but in view of the very small thickness we can reasonably assume that velocity variation is linear and so the velocity gradient can be considered constant. Therefore, the shear stress in the laminar sub-layer would be constant and equal to the boundary shear stress τ_0. Thus the shear stress in the sub-layer is

$$\tau_0 = \mu \left(\frac{\partial u}{\partial y}\right)_{y=0} = \mu \frac{u}{y} \qquad \left\{\because \text{ For linear variation, } \frac{\partial u}{\partial y} = \frac{u}{y}\right\}$$

13.2.4 Boundary Layer Thickness (δ). It is defined as the distance from the boundary of the solid body measured in the y-direction to the point, where the velocity of the fluid is approximately equal to 0.99 times the free stream velocity (U) of the fluid. It is denoted by the symbol δ. For laminar and turbulent zone it is denoted as :

1. δ_{lam} = Thickness of laminar boundary layer,
2. δ_{tur} = Thickness of turbulent boundary layer, and
3. δ' = Thickness of laminar sub-layer.

13.2.5 Displacement Thickness (δ^*). It is defined as the distance, measured perpendicular to the boundary of the solid body, by which the boundary should be displaced to compensate for the reduction in flow rate on account of boundary layer formation. It is denoted by δ^*. It is also defined as :

"The distance perpendicular to the boundary, by which the free-stream is displaced due to the formation of boundary layer".

Expression for δ^*.

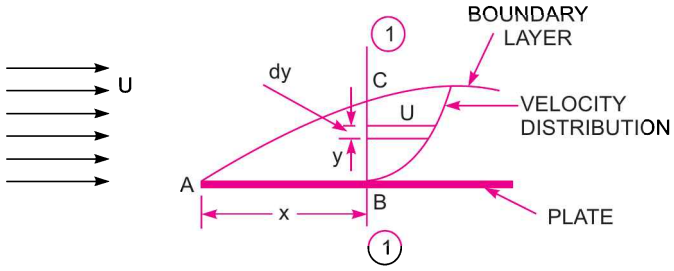


Fig. 13.3 *Displacement thickness.*

Consider the flow of a fluid having free-stream velocity equal to U over a thin smooth plate as shown in Fig. 13.3. At a distance x from the leading edge consider a section 1-1. The velocity of fluid at B is zero and at C, which lies on the boundary layer, is U. Thus velocity varies from zero at B to U at C, where BC is equal to the thickness of boundary layer *i.e.*,

Distance $BC = \delta$

At the section 1-1, consider an elemental strip.

Let y = distance of elemental strip from the plate,

dy = thickness of the elemental strip,

u = velocity of fluid at the elemental strip,

b = width of plate.

Then area of elemental strip, $dA = b \times dy$

Mass of fluid per second flowing through elemental strip

$$= \rho \times \text{Velocity} \times \text{Area of elemental strip}$$

$$= \rho u \times dA = \rho u \times b \times dy \qquad ...(i)$$

If there had been no plate, then the fluid would have been flowing with a constant velocity equal to free-stream velocity (U) at the section 1-1. Then mass of fluid per second flowing through elemental strip would have been

$$= \rho \times \text{Velocity} \times \text{Area} = \rho \times U \times b \times dy \qquad ...(ii)$$

As U is more than u, hence due to the presence of the plate and consequently due to the formation of the boundary layer, there will be a reduction in mass flowing per second through the elemental strip.

This reduction in mass/sec flowing through elemental strip

$$= \text{mass/sec given by equation } (ii) - \text{mass/sec given by equation } (i)$$

$$= \rho U b dy - \rho u b dy = \rho b(U - u)dy$$

$\therefore$ Total reduction in mass of fluid/s flowing through BC due to plate

$$= \int_0^{\delta} \rho b(U - u)dy = \rho b \int_0^{\delta} (U - u)dy \qquad ...(iii)$$

{if fluid is incompressible}

Let the plate is displaced by a distance δ^* and velocity of flow for the distance δ^* is equal to the free-stream velocity (*i.e.*, U). Loss of the mass of the fluid/sec flowing through the distance δ^*

$$= \rho \times \text{Velocity} \times \text{Area}$$

$$= \rho \times U \times \delta^* \times b \qquad \{\because \text{Area} = \delta^* \times b\} \ ...(iv)$$

Equating equation (iii) and (iv), we get

$$\rho b \int_0^{\delta} (U - u)dy = \rho \times U \times \delta^* b$$

Cancelling ρb from both sides, we have

$$\int_0^{\delta} (U - u)dy = U \times \delta^*$$

or
$$\delta^* = \frac{1}{U}\int_0^{\delta} (U - u)dy = \int_0^{\delta} \frac{(U - u)dy}{U} \qquad \left\{\begin{array}{l}\because\ U \text{ is constant and can} \\ \text{be taken inside the integral}\end{array}\right\}$$

$\therefore$
$$\delta^* = \int_0^{\delta} \left(1 - \frac{u}{U}\right) dy. \qquad ...(13.2)$$

13.2.6 Momentum Thickness (θ). Momentum thickness is defined as the distance, measured perpendicular to the boundary of the solid body, by which the boundary should be displaced to compensate for the reduction in **momentum** of the flowing fluid on account of boundary layer formation. It is denoted by θ.

Consider the flow over a plate as shown in Fig. 13.3. Consider the section 1-1 at a distance x from leading edge. Take an elemental strip at a distance y from the plate having thickness (dy). The mass of fluid flowing per second through this elemental strip is given by equation (i) and is equal to $\rho ubdy$.

Momentum of this fluid = Mass × Velocity = $(\rho ubdy)u$

Momentum of this fluid in the absence of boundary layer = $(\rho ubdy)U$

$\therefore$ Loss of momentum through elemental strip = $(\rho ubdy)U - (\rho ubdy) \times u = \rho bu(U - u)dy$

$\therefore$ Total loss of momentum/sec through $BC = \int_0^\delta \rho bu(U - u)dy$ ...(13.3)

Let θ = distance by which plate is displaced when the fluid is flowing with a constant velocity U

$\therefore$ Loss of momentum/sec of fluid flowing through distance θ with a velocity U

= Mass of fluid through θ × velocity

= (ρ × area × velocity) × velocity

$= [\rho \times \theta \times b \times U] \times U$ {$\because$ Area = θ × b}

$= \rho\theta bU^2$ ...(13.4)

Equating equations (13.4) and (13.3), we have

$$\rho\theta bU^2 = \int_0^\delta \rho bu(U - u)dy = \rho b \int_0^\delta u(U - u)dy \quad \{\text{If fluid is assumed incompressible}\}$$

or $$\theta U^2 = \int_0^\delta u(U - u)dy \quad \{\text{cancelling } \rho b \text{ from both sides}\}$$

or $$\theta = \frac{1}{U^2}\int_0^\delta u(U - u)dy = \int_0^\delta \frac{u(U - u)dy}{U^2}$$

$\therefore$ $$\theta = \int_0^\delta \frac{u}{U}\left[1 - \frac{u}{U}\right] dy. \quad ...(13.5)$$

13.2.7 Energy Thickness (δ).** It is defined as the distance measured perpendicular to the boundary of the solid body, by which the boundary should be displaced to compensate for the reduction in kinetic energy of the flowing fluid on account of boundary layer formation. It is denoted by δ**.

Consider the flow over the plate as shown in Fig. 13.3 having section 1-1 at a distance x from leading edge. The mass of fluid flowing per second through the elemental strip of thickness 'dy' at a distance y from the plate as given by equation (i) = $\rho ubdy$

Kinetic energy of this fluid $= \frac{1}{2} m \times \text{velocity}^2 = \frac{1}{2} (\rho ubdy)\, u^2$

Kinetic energy of this fluid in the absence of boundary layer

$$= \frac{1}{2} (\rho ubdy)U^2$$

$\therefore$ Loss of K.E. through elemental strip

$$= \frac{1}{2} (\rho ubdy)U^2 - \frac{1}{2} (\rho ubdy)\, u^2 = \frac{1}{2} \rho ub\, [U^2 - u^2]\, dy$$

$\therefore$ Total loss of K.E. of fluid passing through BC

$$= \int_0^\delta \frac{1}{2}\, \rho u b\, [U^2 - u^2]\, dy = \frac{1}{2}\rho b \int_0^\delta u\,(U^2 - u^2)\, dy$$

{If fluid is considered incompressible}

Let δ^{**} = distance by which the plate is displaced to compensate for the reduction in K.E.

$\therefore$ Loss of K.E. through δ^{**} of fluid flowing with velocity U

$$= \frac{1}{2}\,(\text{mass}) \times \text{velocity}^2 = \frac{1}{2}\,(\rho \times \text{area} \times \text{velocity}) \times \text{velocity}^2$$

$$= \frac{1}{2}\,(\rho \times b \times \delta^{**} \times U)U^2 \qquad \{\because \quad \text{Area} = b \times \delta^{**}\}$$

$$= \frac{1}{2}\,\rho b \delta^{**} U^3$$

Equating the two losses of K.E., we get

$$\frac{1}{2}\,\rho b \delta^{**} U^3 = \frac{1}{2}\rho b \int_0^\delta u\,(U^2 - u^2)\, dy$$

or
$$\delta^{**} = \frac{1}{U^3}\int_0^\delta u\,(U^2 - u^2)\, dy$$

$\therefore$
$$\delta^{**} = \int_0^\delta \frac{u}{U}\left[1 - \frac{u^2}{U^2}\right] dy. \qquad \ldots(13.6)$$

Problem 13.1 *Find the displacement thickness, the momentum thickness and energy thickness for the velocity distribution in the boundary layer given by* $\frac{u}{U} = \frac{y}{\delta}$, *where u is the velocity at a distance y from the plate and* $u = U$ *at* $y = \delta$, *where* δ = *boundary layer thickness. Also calculate the value of* δ^*/θ.

Solution. Given :

Velocity distribution $\quad \frac{u}{U} = \frac{y}{\delta}$

(*i*) Displacement thickness δ^* is given by equation (13.2),

$$\delta^* = \int_0^\delta \left(1 - \frac{u}{U}\right) dy = \int_0^\delta \left(1 - \frac{y}{\delta}\right) dy \qquad \left\{\because \quad \frac{u}{U} = \frac{y}{\delta}\right\}$$

$$= \left[y - \frac{y^2}{2\delta}\right]_0^\delta \qquad \{\delta \text{ is constant across a section}\}$$

$$= \delta - \frac{\delta^2}{2\delta} = \delta - \frac{\delta}{2} = \frac{\delta}{\mathbf{2}}. \quad \textbf{Ans.}$$

(*ii*) Momentum thickness, θ is given by equation (13.5),

$$\theta = \int_0^\delta \frac{u}{U}\left(1 - \frac{u}{U}\right) dy$$

Substituting the value of $\frac{u}{U} = \frac{y}{\delta}$,

$$\theta = \int_0^{\delta} \frac{y}{\delta}\left(1 - \frac{y}{\delta}\right) dy = \int_0^{\delta} \left(\frac{y}{\delta} - \frac{y^2}{\delta^2}\right) dy$$

$$= \left[\frac{y^2}{2\delta} - \frac{y^3}{3\delta^2}\right]_0^{\delta} = \frac{\delta^2}{2\delta} - \frac{\delta^3}{3\delta^2} = \frac{\delta}{2} - \frac{\delta}{3} = \frac{3\delta - 2\delta}{6} = \frac{\delta}{\mathbf{6}}. \textbf{ Ans.}$$

(*iii*) Energy thickness δ^{**} is given by equation (13.6), as

$$\delta^{**} = \int_0^{\delta} \frac{u}{U}\left[1 - \frac{u^2}{U^2}\right] dy = \int_0^{\delta} \frac{y}{\delta}\left[1 - \frac{y^2}{\delta^2}\right] dy \qquad \left\{\because \quad \frac{u}{U} = \frac{y}{\delta}\right\}$$

$$= \int_0^{\delta} \left[\frac{y}{\delta} - \frac{y^3}{\delta^3}\right] dy = \left[\frac{y^2}{2\delta} - \frac{y^4}{4\delta^3}\right]_0^{\delta} = \frac{\delta^2}{2\delta} - \frac{\delta^4}{4\delta^3}$$

$$= \frac{\delta}{2} - \frac{\delta}{4} = \frac{2\delta - \delta}{4} = \frac{\delta}{\mathbf{4}}. \textbf{ Ans.}$$

(*iv*) $$\frac{\delta^*}{\theta} = \frac{\left(\frac{\delta}{2}\right)}{\left(\frac{\delta}{6}\right)} = \frac{\delta}{2} \times \frac{6}{\delta} = \mathbf{3}. \textbf{ Ans.}$$

Problem 13.2 *Find the displacement thickness, the momentum thickness and energy thickness for the velocity distribution in the boundary layer given by* $\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$.

Solution. Given :

Velocity distribution $$\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$$

(*i*) Displacement thickness δ^* is given by equation (13.2),

$$\delta^* = \int_0^{\delta} \left(1 - \frac{u}{U}\right) dy$$

Substituting the value of $\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$, we have

$$\delta^* = \int_0^{\delta} \left\{1 - \left[2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2\right]\right\} dy$$

$$= \int_0^{\delta} \left\{1 - 2\left(\frac{y}{\delta}\right) + \left(\frac{y}{\delta}\right)^2\right\} dy = \left[y - \frac{2y^2}{2\delta} + \frac{y^3}{3\delta^2}\right]_0^{\delta}$$

$$= \delta - \frac{\delta^2}{\delta} + \frac{\delta^3}{3\delta^2} = \delta - \delta + \frac{\delta}{3} = \frac{\delta}{\mathbf{3}}. \quad \textbf{Ans.}$$

(*ii*) Momentum thickness θ, is given by equation (13.5),

$$\theta = \int_0^\delta \frac{u}{U}\left\{1-\frac{u}{U}\right\}dy = \int_0^\delta \left(\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right)\left[1-\left(\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right)\right]dy$$

$$= \int_0^\delta \left[\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right]\left[1-\frac{2y}{\delta}+\frac{y^2}{\delta^2}\right]dy$$

$$= \int_0^\delta \left[\frac{2y}{\delta}-\frac{4y^2}{\delta^2}+\frac{2y^3}{\delta^3}-\frac{y^2}{\delta^2}+\frac{2y^3}{\delta^3}-\frac{y^4}{\delta^4}\right]dy$$

$$= \int_0^\delta \left[\frac{2y}{\delta}-\frac{5y^2}{\delta^2}+\frac{4y^3}{\delta^3}-\frac{y^4}{\delta^4}\right]dy = \left[\frac{2y^2}{2\delta}-\frac{5y^3}{3\delta^2}+\frac{4y^4}{4\delta^3}-\frac{y^5}{5\delta^4}\right]_0^\delta$$

$$= \left[\frac{\delta^2}{\delta}-\frac{5\delta^3}{3\delta^2}+\frac{\delta^4}{\delta^3}-\frac{\delta^5}{5\delta^4}\right] = \delta-\frac{5\delta}{3}+\delta-\frac{\delta}{5}$$

$$= \frac{15\delta-25\delta+15\delta-3\delta}{15} = \frac{30\delta-28\delta}{15} = \mathbf{\frac{2\delta}{15}}. \quad \textbf{Ans.}$$

(*iii*) Energy thickness δ** is given by equation (13.6),

$$\delta^{**} = \int_0^\delta \frac{u}{U}\left[1-\frac{u^2}{U^2}\right]dy = \int_0^\delta \left(\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right)\left(1-\left[\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right]^2\right)dy$$

$$= \int_0^\delta \left(\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right)\left(1-\left[\frac{4y^2}{\delta^2}+\frac{y^4}{\delta^4}-\frac{4y^3}{\delta^3}\right]\right)dy$$

$$= \int_0^\delta \left(\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right)\left(1-\frac{4y^2}{\delta^2}-\frac{y^4}{\delta^4}+\frac{4y^3}{\delta^3}\right)dy$$

$$= \int_0^\delta \left(\frac{2y}{\delta}-\frac{8y^3}{\delta^3}-\frac{2y^5}{\delta^5}+\frac{8y^4}{\delta^4}-\frac{y^2}{\delta^2}+\frac{4y^4}{\delta^4}+\frac{y^6}{\delta^6}-\frac{4y^5}{\delta^5}\right)dy$$

$$= \int_0^\delta \left[\frac{2y}{\delta}-\frac{y^2}{\delta^2}-\frac{8y^3}{\delta^3}+\frac{12y^4}{\delta^4}-\frac{6y^5}{\delta^5}+\frac{y^6}{\delta^6}\right]dy$$

$$= \left[\frac{2y^2}{2\delta}-\frac{y^3}{3\delta^2}-\frac{8y^4}{4\delta^3}+\frac{12y^5}{5\delta^4}-\frac{6y^6}{6\delta^5}+\frac{y^7}{7\delta^6}\right]_0^\delta$$

$$= \frac{\delta^2}{\delta}-\frac{\delta^3}{3\delta^2}-\frac{2\delta^4}{\delta^3}+\frac{12\delta^5}{5\delta^4}-\frac{\delta^6}{\delta^5}+\frac{\delta^7}{7\delta^6} = \delta-\frac{\delta}{3}-2\delta+\frac{12}{5}\delta-\delta+\frac{\delta}{7}$$

$$= -2\delta-\frac{\delta}{3}+\frac{12}{5}\delta+\frac{\delta}{7} = \frac{-210\delta-35\delta+252\delta+15\delta}{105}$$

$$= \frac{-245\delta+267\delta}{105} = \mathbf{\frac{22\delta}{105}}. \quad \textbf{Ans.}$$

▶ 13.3 DRAG FORCE ON A FLAT PLATE DUE TO BOUNDARY LAYER

Consider the flow of a fluid having free-stream velocity equal to U, over a thin plate as shown in Fig. 13.4. The drag force on the plate can be determined if the velocity profile near the plate is known. Consider a small length Δx of the plate at a distance of x from the leading edge as shown in Fig. 13.4 (a). The enlarged view of the small length of the plate is shown in Fig. 13.4 (b).

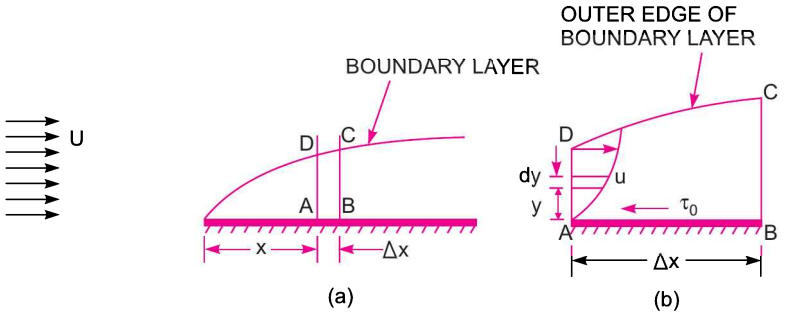


Fig. 13.4 *Drag force on a plate due to boundary layer.*

The shear stress τ_0 is given by $\tau_0 = \mu \left(\frac{du}{dy}\right)_{y=0}$, where $\left(\frac{du}{dy}\right)_{y=0}$ is the velocity distribution near the plate at $y = 0$.

Then drag force or shear force on a small distance Δx is given by

$$\Delta F_D = \text{shear stress} \times \text{area}$$
$$= \tau_0 \times \Delta x \times b \qquad \text{...(13.7)} \quad \{\text{Taking width of plate} = b\}$$

where ΔF_D = drag force on distance Δx

The drag force ΔF_D must also be equal to the rate of change of momentum over the distance Δx.

Consider the flow over the small distance Δx. Let $ABCD$ is the control volume of the fluid over the distance Δx as shown in Fig. 13.4 (b). The edge DC represents the outer edge of the boundary layer.

Let u = velocity at any point within the boundary layer

b = width of plate

Then mass rate of flow entering through the side AD

$$= \int_0^\delta \rho \times \text{velocity} \times \text{area of strip of thickness } dy$$

$$= \int_0^\delta \rho \times u \times b \times dy \qquad \{\because \quad \text{Area of strip} = b \times dy\}$$

$$= \int_0^\delta \rho u b dy$$

Mass rate of flow leaving the side BC

$$= \text{mass through } AD + \frac{\partial}{\partial x}(\text{mass through } AD) \times \Delta x$$

$$= \int_0^\delta \rho u b dy \frac{\partial}{\partial x}\left[\int_0^\delta (\rho u b dy)\right] \times \Delta x$$

From continuity equation for a steady incompressible fluid flow, we have

Mass rate of flow entering AD + mass rate of flow entering DC
= mass rate of flow leaving BC

$\because$ Mass rate of flow entering DC = mass rate of flow through BC – mass rate of flow through AD

$$= \int_0^\delta \rho u b dy + \frac{\partial}{\partial x}\left[\int_0^\delta \rho u b dy\right] \times \Delta x - \int_0^\delta \rho u b dy$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \rho u b dy\right] \times \Delta x$$

The fluid is entering through side DC with a uniform velocity U.

Now let us calculate momentum flux through control volume.

Momentum flux entering through AD

$$= \int_0^\delta \text{momentum flux through strip of thickness } dy$$

$$= \int_0^\delta \text{mass through strip} \times \text{velocity} = \int_0^\delta (\rho u b dy) \times u = \int_0^\delta \rho u^2 b dy$$

Momentum flux leaving the side $BC = \int_0^\delta \rho u^2 b dy + \frac{\partial}{\partial x}\left[\int_0^\delta \rho u^2 b dy\right] \times \Delta x$

Momentum flux entering the side DC = mass rate through DC × velocity

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \rho u b dy\right] \times \Delta x \times U \qquad (\because \text{ Velocity} = U\}$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \rho u U b dy\right] \times \Delta x$$

As U is constant and so it can be taken inside the differential and integral.

$\therefore$ Rate of change of momentum of the control volume
= Momentum flux through BC – Momentum flux through AD
– momentum flux through DC

$$= \int_0^\delta \rho u^2 b dy + \frac{\partial}{\partial x}\left[\int_0^\delta \rho u^2 b dy\right] \times \Delta x - \int_0^\delta \rho u^2 b dy - \frac{\partial}{\partial x}\left[\int_0^\delta \rho u U b dy\right] \times \Delta x$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \rho u^2 b dy\right] \times \Delta x - \frac{\partial}{\partial x}\left[\int_0^\delta \rho u U b dy\right] \times \Delta x$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \rho u^2 b dy - \int_0^\delta \rho u U b dy\right] \times \Delta x$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \left(\rho u^2 b - \rho u U b\right) dy\right] \times \Delta x$$

$$= \frac{\partial}{\partial x}\left[\rho b \int_0^\delta \left(u^2 - uU\right) dy\right] \times \Delta x$$

{For incompressible fluid ρ is constant}

$$= \rho b \frac{\partial}{\partial x}\left[\int_0^{\delta} \left(u^2 - uU\right)dy\right] \times \Delta x \qquad ...(13.8)$$

Now the rate of change of momentum on the control volume *ABCD* must be equal to the total force on the control volume in the same direction according to the momentum principle. But for a flat plate $\frac{\partial p}{\partial x} = 0$, which means there is no external pressure force on the control volume. Also the force on the side *DC* is negligible as the velocity is constant and velocity gradient is zero approximately. The only external force acting on the control volume is the shear force acting on the side *AB* in the direction from *B* to *A* as shown in Fig. 13.4 (*b*). The value of this force is given by equation (13.7) as

$$\Delta F_D = \tau_0 \times \Delta x \times b$$

$\therefore$ Total external force in the direction of rate of change of momentum

$$= -\tau_0 \times \Delta x \times b \qquad ...(13.9)$$

According to momentum principle, the two values given by equations (13.9) and (13.8) should be the same.

$$\therefore \qquad -\tau_0 \times \Delta x \times b = \rho b \frac{\partial}{\partial x}\left[\int_0^{\delta} \left(u^2 - uU\right)dy\right] \times \Delta x$$

Cancelling $\Delta x \times b$, to both sides, we have

$$-\tau_0 = \rho \frac{\partial}{\partial x}\left[\int_0^{\delta} \left(u^2 - uU\right)dy\right]$$

or

$$\tau_0 = -\rho \frac{\partial}{\partial x}\left[\int_0^{\delta} \left(u^2 - uU\right)dy\right] = \rho \frac{\partial}{\partial x}\left[\int_0^{\delta} \left(uU - u^2\right)dy\right]$$

$$= \rho \frac{\partial}{\partial x}\left[\int_0^{\delta} U^2\left(\frac{u}{U} - \frac{u^2}{U^2}\right)dy\right] = \rho U^2 \frac{\partial}{\partial x}\left[\int_0^{\delta} \frac{u}{U}\left[1 - \frac{u}{U}\right]dy\right]$$

or

$$\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\int_0^{\delta} \frac{u}{U}\left[1 - \frac{u}{U}\right]dy\right] \qquad ...(13.10)$$

In equation (13.10), the expression $\int_0^{\delta} \frac{u}{U}\left[1 - \frac{u}{U}\right]dy$ is equal to momentum thickness θ. Hence equation (13.10) is also written as

$$\frac{\tau_0}{\rho U^2} = \frac{\partial \theta}{\partial x} \qquad ...(13.11)$$

Equation (13.11) is known as **Von Karman momentum integral equation** for boundary layer flows.

This is applied to :

1. Laminar boundary layers,
2. Transition boundary layers, and
3. Turbulent boundary layer flows.

For a given velocity profile in laminar zone, transition zone or turbulent zone of a boundary layer, the shear stress τ_0 is obtained from equation (13.10) or (13.11). Then drag force on a small distance Δx of the plate is obtained from equation (13.7) as

$$\Delta F_D = \tau_0 \times \Delta x \times b$$

Then total drag on the plate of length L on one side is

$$F_D = \int \Delta F_D = \int_0^L \tau_0 \times b \times dx \quad \{\text{change } \Delta x = dx\}. \qquad ...(13.12)$$

13.3.1 Local Co-efficient of Drag [C_D*]. It is defined as the ratio of the shear stress τ_0 to the quantity $\frac{1}{2}\rho U^2$. It is denoted by C_D*

Hence
$$C_D^* = \frac{\tau_0}{\frac{1}{2}\rho U^2}. \qquad ...(13.13)$$

13.3.2 Average Co-efficient of Drag [C_D]. It is defined as the ratio of the total drag force to the quantity $\frac{1}{2}\rho A U^2$. It is also called co-efficient of drag and is denoted by C_D.

Hence
$$C_D = \frac{F_D}{\frac{1}{2}\rho A U^2} \qquad ...(13.14)$$

where A = Area of the surface (or plate)
U = Free-stream velocity
ρ = Mass density of fluid.

13.3.3 Boundary Conditions for the Velocity Profiles. The followings are the boundary conditions which must be satisfied by any velocity profile, whether it is in laminar boundary layer zone, or in turbulent boundary layer zone :

1. At $y = 0$, $u = 0$ and $\frac{du}{dy}$ has some finite value
2. At $y = \delta$, $u = U$
3. At $y = \delta$, $\frac{du}{dy} = 0$.

Problem 13.3 *For the velocity profile for laminar boundary layer flows given as*

$$\frac{u}{U} = 2(y/\delta) - (y/\delta)^2$$

find an expression for boundary layer thickness (δ), shear stress (τ_0) and co-efficient of drag (C_D) in terms of Reynold number.

Solution. Given :

(*i*) The velocity distribution $\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$ ...(*i*)

Substituting this value of $\frac{u}{U}$ in equation (13.10), we get

$$\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\int_0^\delta \frac{u}{U}\left(1-\frac{u}{U}\right)dy\right] = \frac{\partial}{\partial x}\left[\int_0^\delta \left[\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right]\left[1-\left(\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right)\right]dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \left[\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right]\left[1-\frac{2y}{\delta}+\frac{y^2}{\delta^2}\right]dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \left[\frac{2y}{\delta}-\frac{4y^2}{\delta^2}+\frac{2y^3}{\delta^3}-\frac{y^2}{\delta^2}+\frac{2y^3}{\delta^3}-\frac{y^4}{\delta^4}\right]dy\right]$$

$$= \frac{\partial}{\partial x}\int_0^\delta \left[\frac{2y}{\delta}-\frac{5y^2}{\delta^2}+\frac{4y^3}{\delta^3}-\frac{y^4}{\delta^4}\right]dy = \frac{\partial}{\partial x}\left[\frac{2y^2}{2\delta}-\frac{5\times y^3}{3\delta^2}+\frac{4y^4}{4\delta^3}-\frac{y^5}{5\delta^4}\right]_0^\delta$$

$$= \frac{\partial}{\partial x}\left[\frac{\delta^2}{\delta}-\frac{5}{3}\frac{\delta^3}{\delta^2}+\frac{\delta^4}{\delta^3}-\frac{\delta^5}{5\delta^4}\right] = \frac{\partial}{\partial x}\left[\delta-\frac{5}{3}\delta+\delta-\frac{\delta}{5}\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{15\delta-25\delta+15\delta-3\delta}{15}\right] = \frac{\partial}{\partial x}\left[\frac{30\delta-28\delta}{15}\right] = \frac{\partial}{\delta x}\left[\frac{2\delta}{15}\right] = \frac{2}{15}\frac{\partial}{\partial x}\ [\delta]$$

$$\therefore \quad \tau_0 = \rho U^2 \times \frac{2}{15}\frac{\partial}{\partial x}[\delta] = \frac{2}{15}\rho U^2\frac{\partial[\delta]}{\partial x} \qquad ...(13.15)$$

The shear stress at the boundary in laminar flow is also given by Newton's law of viscosity as

$$\tau_0 = \mu\left(\frac{du}{dy}\right)_{y=0} \qquad ...(ii)$$

But from equation (*i*), $u = U\left[\frac{2y}{\delta}-\frac{y^2}{\delta^2}\right]$

$$\therefore \quad \frac{du}{dy} = U\left[\frac{2}{\delta}-\frac{2y}{\delta^2}\right] \qquad \{\because \ U \text{ is constant}\}$$

$$\therefore \quad \left(\frac{du}{dy}\right)_{y=0} = U\left[\frac{2}{\delta}-\frac{2\times(0)}{\delta^2}\right] = \frac{2U}{\delta}$$

Substituting this value in (*ii*), we get

$$\tau_0 = \mu\times\frac{2U}{\delta} = \frac{2\mu U}{\delta} \qquad ...(iii)$$

Equating the two values of τ_0 given by equation (13.15) and (*iii*)

$$\frac{2}{15}\rho U^2\frac{\partial}{\partial x}[\delta] = \frac{2\mu U}{\delta}$$

or $$\frac{\delta\partial}{\partial x}[\delta] = \frac{15\mu U}{\rho U^2} = \frac{15\mu}{\rho U} \quad \text{or} \quad \delta\partial[\delta] = \frac{15\mu}{\rho U}\partial x$$

As the boundary layer thickness (δ) is a function of x only.
Hence partial derivative can be changed to total derivative

$$\therefore \qquad \delta d[\delta] = \frac{15\mu}{\rho U}\, dx$$

On integration, we get $\quad \dfrac{\delta^2}{2} = \dfrac{15\mu}{\rho U}\, x + C \qquad \left\{\dfrac{\mu}{\rho U} \text{ is constant}\right\}$

$x = 0$, $\delta = 0$ and hence $C = 0$

$$\therefore \qquad \frac{\delta^2}{2} = \frac{15\mu x}{\rho U}$$

$$\therefore \qquad \delta = \sqrt{\frac{2 \times 15\mu x}{\rho U}} = \sqrt{\frac{30\mu x}{\rho U}} = 5.48\sqrt{\frac{\mu x}{\rho U}} \qquad \ldots(13.16)$$

$$= 5.48\sqrt{\frac{\mu x \times x}{\rho U \times x}} = 5.48\sqrt{\frac{x^2}{R_{e_x}}} \qquad \left\{\because \quad R_{e_x} = \frac{\rho U x}{\mu}\right\}$$

$$= 5.48\,\frac{x}{\sqrt{R_{e_x}}} \qquad \ldots(13.17)$$

In equation (13.16), μ, ρ and U are constant and hence it is clear from this equation that thickness of laminar boundary layer is proportional to the square root of the distance from the leading edge. Equation (13.17) gives the thickness of laminar boundary layer in terms of Reynolds number.

(*ii*) **Shear stress (τ_0) in terms of Reynolds number**

From equation (*iii*), we have $\tau_0 = \dfrac{2\mu U}{\delta}$

Substituting the value of δ from equation (13.17), in the above equation, we get

$$\tau_0 = \frac{2\mu U}{5.48\dfrac{x}{\sqrt{R_{e_x}}}} = \frac{2\mu U\sqrt{R_{e_x}}}{5.48x} = 0.365\,\frac{\mu U}{x}\sqrt{R_{e_x}}$$

(*iii*) **Co-efficient of Drag (C_D)**

From equation (13.14), we have $C_D = \dfrac{F_D}{\frac{1}{2}\rho A U^2}$

where F_D is given by equation (13.12) as

$$F_D = \int_0^L \tau_0 \times b \times dx = \int_0^L 0.365\frac{\mu U}{x}\sqrt{R_{e_x}} \times b \times dx$$

$$= 0.365\int_0^L \frac{\mu U}{x}\sqrt{\frac{\rho U x}{\mu}} \times b \times dx \qquad \left\{\because \quad R_{e_x} = \frac{\rho U x}{\mu}\right\}$$

$$= 0.365\int_0^L \mu U\sqrt{\frac{\rho U}{\mu}} \times \frac{1}{\sqrt{x}} \times b \times dx$$

$$= 0.365\ \mu U \sqrt{\frac{\rho U}{\mu}} \times b \int_0^L x^{-1/2}\, dx = 0.365\ \mu U \sqrt{\frac{\rho U}{\mu}} \times b \times \left[\frac{x^{1/2}}{\frac{1}{2}}\right]_0^L$$

$$= 0.365 \times 2\mu U \sqrt{\frac{\rho U}{\mu}} \times b \times \sqrt{L}$$

$$= 0.73\ b\mu U \sqrt{\frac{\rho UL}{\mu}} \qquad \ldots(13.18)$$

$$\therefore \qquad C_D = \frac{0.73\ b\mu U \sqrt{\frac{\rho UL}{\mu}}}{\frac{1}{2}\rho A U^2}$$

where A = Area of plate = Length of plate × width = $L \times b$

$$\therefore \qquad C_D = \frac{0.73\ b\mu U}{\frac{1}{2}\rho \times L \times b \times U^2} \sqrt{\frac{\rho UL}{\mu}} = \frac{1.46\ \mu}{\rho LU} \sqrt{\frac{\rho UL}{\mu}}$$

$$= \frac{1.46\sqrt{\mu}}{\sqrt{\rho UL}} = 1.46 \sqrt{\frac{\mu}{\rho UL}} = \frac{1.46}{\sqrt{R_{e_L}}} \qquad \ldots(13.19) \qquad \left\{\because \sqrt{\frac{\mu}{\rho UL}} = \frac{1}{\sqrt{R_{e_L}}}\right\}$$

Problem 13.4 *For the velocity profile given in problem 13.3, find the thickness of boundary layer at the end of the plate and the drag force on one side of a plate 1 m long and 0.8 m wide when placed in water flowing with a velocity of 150 mm per second. Calculate the value of co-efficient of drag also. Take* μ *for water = 0.01 poise.*

Solution. Given :

Length of plate, $L = 1$ m

Width of plate, $b = 0.8$ m

Velocity of fluid (water), $U = 150$ mm/s $= 0.15$ m/s

μ for water $= 0.01$ poise $= \frac{0.01}{10} \frac{\text{Ns}}{\text{m}^2} = 0.001 \frac{\text{Ns}}{\text{m}^2}$

Reynold number at the end of the plate *i.e.,* at a distance of 1 m from leading edge is given by

$$R_{e_L} = \frac{\rho UL}{\mu} = 1000 \times \frac{0.15 \times 1.0}{.001} \qquad (\because \quad \rho = 1000)$$

$$= \frac{1000 \times .15 \times 1.0}{0.001} = 150000$$

(*i*) As laminar boundary layer exists upto Reynold number = 2×10^5. Hence this is the case of laminar boundary layer. Thickness of boundary layer at $x = 1.0$ m is given by equation (13.17) as

$$\delta = 5.48 \frac{x}{\sqrt{R_{e_x}}} = \frac{5.48 \times 1.0}{\sqrt{150000}} = 0.01415 \text{ m} = \textbf{14.15 mm. Ans.}$$

(*ii*) Drag force on one side of the plate is given by equation (13.18)

$$F_D = 0.73\, b\mu U \sqrt{\frac{\rho UL}{\mu}}$$

$$= 0.73 \times 0.8 \times 0.001 \times 0.15 \times \sqrt{150000} \qquad \left\{\because \quad \frac{\rho UL}{\mu} = R_{e_L}\right\}$$

$$= \mathbf{0.0338\ N.\ Ans.}$$

(*iii*) Co-efficient of drag, C_D is given by equation (13.19) as

$$C_D = \frac{1.46}{\sqrt{R_{e_L}}} = \frac{1.46}{\sqrt{150000}} = \mathbf{.00376.\ Ans.}$$

Problem 13.5 *For the velocity profile for laminar boundary layer* $\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3$.

Determine the boundary layer thickness, shear stress, drag force and co-efficient of drag in terms of Reynold number.

Solution. Given :

Velocity distribution, $\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3$

Using equation (13.10), we have $\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\int_0^\delta \frac{u}{U}\left(1 - \frac{u}{U}\right) dy\right]$

Substituting the value of $\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3$ in the above equation

$$\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\int_0^\delta \left[\frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3\right]\left[1 - \left\{\frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3\right\}\right] dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \left(\frac{3y}{2\delta} - \frac{y^3}{2\delta^3}\right)\left(1 - \frac{3y}{2\delta} + \frac{y^3}{2\delta^3}\right) dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \left(\frac{3y}{2\delta} - \frac{9y^2}{4\delta^2} + \frac{3y^4}{4\delta^4} - \frac{y^3}{2\delta^3} + \frac{3y^4}{4\delta^4} - \frac{y^6}{4\delta^6}\right) dy\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{3y^2}{2\times 2\delta} - \frac{9y^3}{3\times 4\delta^2} + \frac{3y^5}{5\times 4\delta^4} - \frac{y^4}{4\times 2\delta^3} + \frac{3y^5}{5\times 4\delta^4} - \frac{y^7}{7\times 4\delta^6}\right]_0^\delta$$

$$= \frac{\partial}{\partial x}\left[\frac{3\delta^2}{4\delta} - \frac{3\delta^3}{4\delta^2} + \frac{3}{20}\frac{\delta^5}{\delta^4} - \frac{1}{8}\frac{\delta^4}{\delta^3} + \frac{3}{20}\frac{\delta^5}{\delta^4} - \frac{1}{28}\frac{\delta^7}{\delta^6}\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{3}{4}\delta - \frac{3}{4}\delta + \frac{3}{20}\delta - \frac{1}{8}\delta + \frac{3}{20}\delta - \frac{1}{28}\delta\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{6}{20}\delta - \frac{1}{8}\delta - \frac{1}{28}\delta\right] = \frac{\partial\delta}{\partial x}\left[\frac{84-35-10}{280}\right] = \frac{39}{280}\frac{\partial\delta}{\partial x}$$

$$\tau_0 = \rho U^2 \times \frac{39}{280}\frac{\partial\delta}{\partial x} = \frac{39}{280}\rho U^2 \frac{\partial\delta}{\partial x} \qquad ...(13.20)$$

Also the shear stress τ_0 is given by $\tau_0 = \mu\left(\frac{du}{dy}\right)_{y=0}$, where $u = U\left[\frac{3}{2}\frac{y}{\delta} - \frac{y^3}{2\delta^3}\right]$

$$\therefore \qquad \frac{du}{dy} = U\left[\frac{3}{2\delta} - \frac{3y^2}{2\delta^3}\right]$$

Hence $$\left(\frac{du}{dy}\right)_{y=0} = U\left[\frac{3}{2\delta} - \frac{3}{2\delta^3}\times 0\right] = \frac{3U}{2\delta}$$

$$\therefore \qquad \tau_0 = \mu\left(\frac{du}{dy}\right)_{y=0} = \mu\frac{3U}{2\delta} = \frac{3}{2}\frac{\mu U}{\delta} \qquad ...(13.21)$$

Equating the two values of τ_0 given by equations (13.20) and (13.21)

$$\frac{39}{280}\rho U^2\frac{\partial\delta}{\partial x} = \frac{3}{2}\frac{\mu U}{\delta}$$

$$\therefore \qquad \delta\partial\delta = \frac{3}{2}\mu U \times \frac{280}{39}\times\frac{1}{\rho U^2}\partial x = \frac{420}{39}\frac{\mu}{\rho U}\partial x$$

Integrating, we get $$\frac{\delta^2}{2} = \frac{420}{39}\frac{\mu}{\rho U}x + C$$

where $x = 0, \delta = 0, \therefore C = 0$

$$\therefore \qquad \frac{\delta^2}{2} = \frac{420}{39}.\frac{\mu}{\rho U}x$$

or $$\delta = \sqrt{\frac{420\times 2}{39}\frac{\mu}{\rho U}x} = 4.64\sqrt{\frac{\mu x}{\rho U}} = 4.64\sqrt{\frac{\mu x\times x}{\rho U x}}$$

$$= 4.64\sqrt{\frac{\mu}{\rho U x}}x = \frac{4.64\,x}{\sqrt{R_{e_x}}} \qquad ...(13.22)$$

(*i*) **Shear Stress τ_0.** Substituting the value of δ from equation (13.22) into equation (13.21), we get

$$\tau_0 = \frac{3}{2}\frac{\mu U}{\frac{4.64\,x}{\sqrt{R_{e_x}}}} = \frac{3}{9.28}\frac{\mu U\sqrt{R_{e_x}}}{x} = 0.323\frac{\mu U}{x}\sqrt{R_{e_x}}$$

(*ii*) **Drag force (F_D)**

Using equation (13.12), we get the drag force as

$$F_D = \int_0^L \tau_0 \times b \times dx = \int_0^L 0.323\frac{\mu U}{x}\sqrt{R_{e_x}} \times b \times dx$$

$$= 0.323 \int_0^L \frac{\mu U}{x}\sqrt{\frac{\rho U x}{\mu}} \times b \times dx = 0.323\ \mu U \sqrt{\frac{\rho U}{\mu}} \times b \int_0^L \frac{1}{\sqrt{x}} dx$$

$$= 0.323\ \mu U \sqrt{\frac{\rho U}{\mu}} \times b \int_0^L x^{-1/2}\, dx$$

$$= 0.323\ \mu U \sqrt{\frac{\rho U}{\mu}} \times b \left[\frac{x^{1/2}}{\frac{1}{2}}\right]_0^L = 0.323 \times 2\ \mu U \sqrt{\frac{\rho U}{\mu}} \times b \left[\sqrt{L}\right]$$

$$= 0.646\ \mu U \sqrt{\frac{\rho U L}{\mu}} \times b \qquad \ldots(13.23)$$

(*iii*) **Drag Co-efficient (C_D).** Using equation (13.14), we get the value of C_D as

$$C_D = \frac{F_D}{\frac{1}{2}\rho A U^2}, \text{ where } A = b \times L$$

$$= \frac{0.646\ \mu U \sqrt{\frac{\rho U L}{\mu}} \times b}{\frac{1}{2}\rho \times b \times L \times U^2} = 0.646 \times 2 \times \frac{\mu}{\rho U L} \times \sqrt{\frac{\rho U L}{\mu}} = \frac{1.292}{\sqrt{\frac{\rho U L}{\mu}}}$$

$$= \frac{1.292}{\sqrt{R_{e_L}}}. \qquad \left\{\because \sqrt{\frac{\rho U L}{\mu}} = \sqrt{R_{e_L}}\right\} \qquad \ldots(13.24)$$

Problem 13.6 *For the velocity profile for laminar boundary layer*

$$\frac{u}{U} = 2\,(y/\delta) - 2(y/\delta)^3 + (y/\delta)^4$$

obtain an expression for boundary layer thickness, shear stress, drag force on one side of the plate and co-efficient of drag in term of Reynold number.

Solution. Given :

(*i*) The velocity profile, $\frac{u}{U} = \frac{2y}{\delta} - \frac{2y^3}{\delta^3} + \frac{y^4}{\delta^4}$

Using equation (13.10), we have

$$\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\int_0^\delta \frac{u}{U}\left(1 - \frac{u}{U}\right) dy\right]$$

Substituting the given velocity profile in the above equation

$$\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\int_0^\delta \left(\frac{2y}{\delta} - \frac{2y^3}{\delta^3} + \frac{y^4}{\delta^4}\right)\left(1 - \left\{\frac{2y}{\delta} - \frac{2y^3}{\delta^3} + \frac{y^4}{\delta^4}\right\}\right) dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \left(\frac{2y}{\delta} - \frac{2y^3}{\delta^3} + \frac{y^4}{\delta^4}\right)\left(1 - \frac{2y}{\delta} + \frac{2y^3}{\delta^3} - \frac{y^4}{\delta^4}\right) dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^{\delta}\left(\frac{2y}{\delta}-\frac{4y^2}{\delta^2}+\frac{4y^4}{\delta^4}-\frac{2y^5}{\delta^5}-\frac{2y^3}{\delta^3}+\frac{4y^4}{\delta^4}-\frac{4y^6}{\delta^6}+\frac{2y^7}{\delta^7}+\frac{y^4}{\delta^4}-\frac{2y^5}{\delta^5}+\frac{2y^7}{\delta^7}-\frac{y^8}{\delta^8}\right)dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^{\delta}\left(\frac{2y}{\delta}-\frac{4y^2}{\delta^2}-\frac{2y^3}{\delta^3}+\frac{9y^4}{\delta^4}-\frac{4y^5}{\delta^5}-\frac{4y^6}{\delta^6}+\frac{4y^7}{\delta^7}-\frac{y^8}{\delta^8}\right)dy\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{2y^2}{2\delta}-\frac{4y^3}{3\delta^2}-\frac{2y^4}{4\delta^3}+\frac{9y^5}{5\delta^4}-\frac{4y^6}{6\delta^5}-\frac{4y^7}{7\delta^6}+\frac{4y^8}{8\delta^7}-\frac{y^9}{9\delta^8}\right]_0^{\delta}$$

$$= \frac{\partial}{\partial x}\left[\delta-\frac{4}{3}\delta-\frac{1}{2}\delta+\frac{9}{5}\delta-\frac{2}{3}\delta-\frac{4}{7}\delta+\frac{1}{2}\delta-\frac{1}{9}\delta\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{315-420+63\times 9-210-45\times 4-35}{315}\right]\delta$$

$$= \frac{\partial}{\partial x}\left[\frac{315-420+567-210-180-35}{315}\right]\delta$$

$$= \frac{\partial}{\partial x}\left[\frac{882-845}{815}\right]\delta = \frac{\partial}{\partial x}\left[\frac{37}{315}\right]\delta = \frac{37}{315}\frac{\partial\delta}{\partial x}$$

$$\therefore \quad \tau_0 = \frac{37}{315}\rho U^2\frac{\partial\delta}{\partial x} \qquad \text{...(13.25)}$$

Also shear stress is given by Newton's law of viscosity as

$$\tau_0 = \mu\left(\frac{\partial u}{\partial y}\right)_{y=0}$$

where $u = U\left[\frac{2y}{\delta}-\frac{2y^2}{\delta^2}+\frac{y^4}{\delta^4}\right]$

$$\therefore \qquad \left(\frac{du}{dy}\right) = U\left[\frac{2}{\delta}-\frac{4y}{\delta^2}-\frac{4y^3}{\delta^4}\right]$$

$$\therefore \qquad \left(\frac{\partial u}{\partial y}\right)_{y=0} = U\left[\frac{2}{\delta}-\frac{4}{\delta^2}(0)-\frac{4}{\delta^4}(0)\right] = \frac{2U}{\delta}$$

$$\therefore \qquad \tau_0 = \mu\left(\frac{\partial u}{\partial y}\right)_{y=0} = \mu\times\frac{2U}{\delta} = \frac{2U\mu}{\delta} \qquad \text{...(13.26)}$$

Equating the two values of τ_0 given by equations (13.25) and (13.26)

$$\frac{37}{315}\rho U^2\frac{\partial\delta}{\partial x} = \frac{2U\mu}{\delta} \quad \text{or} \quad \delta\partial\delta = \frac{315}{37}\times\frac{2U\mu}{\rho U^2}\partial x = \frac{630}{37}\frac{\mu}{\rho U}\partial x$$

On integration, we get $\frac{\delta^2}{2} = \frac{630}{37}\frac{\mu}{\rho U}x + C$, where C = Constant of integration

At $x = 0$, $\delta = 0$ and hence $C = 0$

$$\therefore \quad \frac{\delta^2}{2} = \frac{630}{37}\frac{\mu}{\rho U}x$$

$$\therefore \quad \delta = \sqrt{\frac{630 \times 2}{37}\frac{\mu}{\rho U}}\, x = 5.84\sqrt{\frac{\mu x}{\rho U}}$$

$$= 5.84\sqrt{\frac{\mu x \times x}{\rho U x}} = 5.84\sqrt{\frac{\mu}{\rho U x}} \times x = \frac{5.84x}{\sqrt{R_{e_x}}} \qquad ...(13.27)$$

(*ii*) **Shear Stress (τ_0).** Substituting the value of δ from (13.27) into (13.26)

$$\tau_0 = \frac{2U\mu}{\delta} = \frac{2U\mu}{\dfrac{5.84x}{\sqrt{R_{e_x}}}} = \frac{2U\mu}{5.84x}\sqrt{R_{e_x}} = 0.34\,\frac{U\mu}{x}\sqrt{R_{e_x}}\,.$$

(*iii*) **Drag Force (F_D)** on one side of the plate :
Using equation (13.12), we get

$$F_D = \int_0^L \tau_0 \times b \times dx = \int_0^L 0.34\,\frac{U\mu}{x}\sqrt{R_{e_x}} \times b \times dx = \int_0^L 0.34\,\frac{U\mu}{\mu}\frac{\sqrt{\rho U\mu}}{\mu}\,b\,dx$$

$$= 0.34\,U\mu\sqrt{\frac{\rho U}{\mu}} \times b\int_0^L x^{-1/2}\,dx = 0.34\,U\mu\sqrt{\frac{\rho U}{\mu}} \times b \times \left[\frac{x^{1/2}}{1/2}\right]_0^L$$

$$= 0.34 \times 2U\mu\sqrt{\frac{\rho U}{\mu}}\, b\sqrt{L} = 0.68\, b\mu U\sqrt{\frac{\rho UL}{\mu}} \qquad ...(13.28)$$

(*iv*) **Drag Co-efficient (C_D)**

Using equation (13.14), $C_D = \dfrac{F_D}{\frac{1}{2}\rho A U^2}$, where $A = b \times L$

$$= \frac{0.68 \times b \times \mu U \times \sqrt{\dfrac{\rho UL}{\mu}}}{\frac{1}{2}\rho \times b \times L \times U^2} = 0.68 \times 2\,\frac{\mu}{\rho UL} \times \sqrt{\frac{\rho UL}{\mu}} = 1.36 \times \frac{1}{\sqrt{\dfrac{\rho UL}{\mu}}}$$

$$= 1.36 \times \frac{1}{\sqrt{R_{e_L}}}. \qquad ...(13.29)$$

Problem 13.7 *For the velocity profile for laminar boundary flow* $\dfrac{u}{U} = \sin\left(\dfrac{\pi}{2}\dfrac{y}{\delta}\right)$.

Obtain an expression for boundary layer thickness, shear stress, drag force on one side of the plate and co-efficient of drag in terms of Reynold number.

Solution. (*i*) The velocity profile is $\dfrac{u}{U} = \sin\left(\dfrac{\pi}{2}\dfrac{y}{\delta}\right)$.

Substituting this value in equation (13.10), we have

$$\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\int_0^{\delta} \frac{u}{U}\left(1-\frac{u}{U}\right)dy\right] = \frac{\partial}{\partial x}\left[\int_0^{\delta} \sin\left(\frac{\pi}{2}\frac{y}{\delta}\right)\left[1-\sin\left(\frac{\pi}{2}\cdot\frac{y}{\delta}\right)\right]dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^{\delta}\left[\sin\left(\frac{\pi}{2}\frac{y}{\delta}\right)-\sin^2\left(\frac{\pi}{2}\frac{y}{\delta}\right)\right]dy\right]$$

$$= \frac{\partial}{\partial x}\left[\left[\frac{-\cos\frac{\pi y}{2\delta}}{\frac{\pi}{2\delta}}\right]-\left[\frac{\frac{\pi y}{2\delta}\times\frac{1}{2}}{\frac{\pi}{2\delta}}-\frac{\sin 2\left(\frac{\pi}{2}\frac{y}{\delta}\right)}{4\times\frac{\pi}{2\delta}}\right]\right]_0^{\delta}$$

$$\left\{\because \int \sin^2\left(\frac{\pi}{2}\frac{y}{\delta}\right)dy = \frac{\frac{\pi y}{2\delta}\times\frac{1}{2}}{\frac{\pi}{2\delta}}-\frac{\sin 2\left(\frac{\pi}{2}\frac{y}{\delta}\right)}{4\times\frac{\pi}{2\delta}}\right\}$$

$\because$ $$\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\left(\frac{-\cos\frac{\pi}{2}\frac{\delta}{\delta}+\cos\frac{\pi}{2}\times\frac{0}{\delta}}{\frac{\pi}{2\delta}}\right)-\left[\frac{\frac{\pi}{2}\frac{\delta}{\delta}\times\frac{1}{2}}{\frac{\pi}{2\delta}}-0\right]\right]$$

$$= \frac{\partial}{\partial x}\left[\left(0+\frac{1}{\frac{\pi}{2\delta}}\right)-\frac{\left(\frac{\pi}{4}\right)}{\frac{\pi}{2\delta}}\right] = \frac{\partial}{\partial x}\left[\frac{2\delta}{\pi}-\frac{\pi}{4}\times\frac{2\delta}{\pi}\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{2\delta}{\pi}-\frac{\delta}{2}\right] = \frac{\partial}{\partial x}\left[\frac{4-\pi}{2\pi}\right]\delta = \left(\frac{4-\pi}{2\pi}\right)\frac{\partial\delta}{\partial x}$$

$\therefore$ $$\tau_0 = \left(\frac{4-\pi}{2\pi}\right)\rho U^2\frac{\partial\delta}{\partial x} \qquad ...(13.30)$$

τ_0 is also equal = $\mu\left(\frac{du}{dy}\right)_{\text{at } y=0}$

But $$u = U\sin\left(\frac{\pi}{2}\frac{y}{\delta}\right)$$

$\therefore$ $$\left(\frac{du}{dy}\right) = U\cos\left(\frac{\pi}{2}\frac{y}{\delta}\right)\times\frac{\pi}{2\delta}$$

$$\left(\frac{du}{dy}\right)_{y=0} = U\times\frac{\pi}{2\delta}\cos\left(\frac{\pi}{2}\times\frac{0}{\delta}\right) = \frac{U\pi}{2\delta}$$

$$\therefore \qquad \tau_0 = \mu \left(\frac{\partial u}{\partial y}\right)_{y=0} = \frac{\mu U \pi}{2\delta} \qquad \ldots(13.31)$$

Equating the two values τ_0 given by equations (13.30) and (13.31)

$$\left(\frac{4-\pi}{2\pi}\right)\rho U^2 \frac{\partial \delta}{\partial x} = \frac{\mu U \pi}{2\delta} \quad \text{or} \quad \delta \partial \delta = \frac{\mu U \pi}{2} \times \frac{2\pi}{4-\pi} \times \frac{1}{\rho U^2}\, \partial x$$

$$\therefore \qquad \delta \partial \delta = \frac{\pi^2}{(4-\pi)} \frac{\mu U}{\rho U^2} \,.\, \partial x = 11.4975 \frac{\mu}{\rho U} \partial x$$

Integrating, we get $\qquad \frac{\delta^2}{2} = 11.4975 \frac{\mu}{\rho U} x + C$

At $x = 0$, $\delta = 0$ and hence $C = 0$

$$\therefore \qquad \frac{\delta^2}{2} = 11.4975 \frac{\mu}{\rho U} x$$

$$\therefore \qquad \delta = \sqrt{2 \times 11.4975 \frac{\mu}{\rho U} x} = 4.795 \sqrt{\frac{\mu}{\rho U}}\, x$$

$$= 4.795 \sqrt{\frac{\mu}{\rho U x}} = 4.795 \sqrt{\frac{\mu}{\rho U x}} \times x$$

$$= \frac{4.795\, x}{\sqrt{R_{e_x}}} \qquad \ldots(13.32)$$

(*ii*) **Shear Stress (τ_0)**

From equation (13.31), $\qquad \tau_0 = \frac{\mu U \pi}{2\delta} = \frac{\mu U \pi}{\frac{2 \times 4.795\, x}{\sqrt{R_{e_x}}}} = \frac{\mu U \pi \sqrt{R_{e_x}}}{2 \times 4.795\, x}$

$$= \frac{\pi}{2 \times 4.795} \frac{\mu U}{x} \sqrt{R_{e_x}} = 0.327 \frac{\mu U}{x} \sqrt{R_{e_x}}\,.$$

(*iii*) **Drag force (F_D)** on one side of the plate is given by equation (13.12)

$$F_D = \int_0^L \tau_0 \times b \times dx = \int_0^L 0.327 \frac{\mu U}{x} \sqrt{R_{e_x}} \times b \times dx = 0.327\, \mu U \times b \int_0^L \frac{1}{x} \sqrt{\frac{\rho U x}{\mu}}\, dx$$

$$= 0.327\, \mu U \times b \times \sqrt{\frac{\rho U}{\mu}} \int_0^L x^{-1/2}\, dx = 0.327\, \mu U \times b \times \sqrt{\frac{\rho U}{\mu}} \left[\frac{x^{1/2}}{\frac{1}{2}}\right]_0^L$$

$$= 0.327 \times 2 \times \mu U \times b \sqrt{\frac{\rho U}{\mu}} \times \sqrt{L}$$

$$= 0.655 \times \mu U \times b \times \sqrt{\frac{\rho U L}{\mu}} \qquad \ldots(13.33)$$

(*iv*) **Co-efficient of drag, C_D** is given by equation (13.14),

$$C_D = \frac{F_D}{\frac{1}{2}\rho A U^2}, \text{ where } A = b \times L$$

$$\therefore \quad C_D = \frac{0.655 \times \mu U \times b \times \sqrt{\frac{\rho UL}{\mu}}}{\frac{1}{2}\rho U^2 \times b \times L} = 0.655 \times 2 \times \frac{\mu}{\rho UL} \times \sqrt{\frac{\rho UL}{\mu}}$$

$$= 1.31 \times \frac{1}{\sqrt{\frac{\rho UL}{\mu}}} = \frac{1.31}{\sqrt{R_{e_L}}} \qquad \ldots(13.34)$$

Note. $\int \sin^2 x ds = \left(\frac{x}{2} - \frac{\sin 2x}{4}\right)$ is used.

Table 13.1 shows the values of boundary layer thickness and co-efficients of drag in terms of Reynold number for various velocity distributions

Table 13.1

	Velocity Distribution	δ	C_D
1.	$\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$	$5.48\, x/\sqrt{R_{e_x}}$	$1.46/\sqrt{R_{e_L}}$
2.	$\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3$	$4.64\, x/\sqrt{R_{e_x}}$	$1.292/\sqrt{R_{e_L}}$
3.	$\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - 2\left(\frac{y}{\delta}\right)^3 + \left(\frac{y}{\delta}\right)^4$	$5.84\, x/\sqrt{R_{e_x}}$	$1.36/\sqrt{R_{e_L}}$
4.	$\frac{u}{U} = \sin\left(\frac{\pi}{2}\frac{y}{\delta}\right)$	$4.79\, x/\sqrt{R_{e_x}}$	$1.31/\sqrt{R_{e_L}}$
5.	Blasius's Solution	$4.91\, x/\sqrt{R_{e_x}}$	$1.328/\sqrt{R_{e_L}}$

Problem 13.8 *For the velocity profile in laminar boundary layer as,*

$$\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3$$

find the thickness of the boundary layer and the shear stress 1.5 m from the leading edge of a plate. The plate is 2 m long and 1.4 m wide and is placed in water which is moving with a velocity of 200 mm per second. Find the total drag force on the plate if μ for water = .01 poise.

Solution. Given :

Velocity profile is $\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3$

Distance of x from leading edge, x = 1.5 m
Length of plate, L = 2 m
Width of plate, b = 1.4 m
Velocity of plate, U = 200 mm/s = 0.2 m/s
Viscosity of water, $\mu = 0.01 \text{ poise} = \frac{0.01}{10} = 0.001 \text{ Ns/m}^2$

For the given velocity profile, thickness of boundary layer is given by equation (13.22) as

$$\delta = \frac{4.64\, x}{\sqrt{R_{e_x}}}$$

$$\left[\text{Here } R_{e_x} = \frac{\rho U x}{\mu} = 1000 \times \frac{0.2 \times 1.5}{0.001} = 300000\right]$$

$$\delta = \frac{4.64 \times 1.5}{\sqrt{300000}} = 0.0127 \text{ m} = \textbf{12.7 mm. Ans.}$$

Shear stress (τ_0) is given by $\tau_0 = 0.323 \frac{\mu U}{x}\sqrt{R_{e_x}}$

$$= 0.323 \times 0.001 \times \frac{0.2}{1.5} \times \sqrt{300000} = \textbf{0.0235 N/m}^2\textbf{. Ans.}$$

Drag Force (F_D) on one side of the plate is given by (13.23) as

$$F_D = 0.646\, \mu U \sqrt{\frac{\rho UL}{\mu}} \times b$$

$$= 0.646 \times 0.001 \times 0.2 \times \sqrt{1000 \times \frac{0.2 \times 2.0}{0.001}} \times 1.4$$

$$= .646 \times 0.001 \times 0.2 \times \sqrt{400000} \times 1.4 = 0.1138 \text{ N}$$

$\therefore$ Total drag force = Drag force on both sides of the plate
= 2 × 0.1138 = **0.2276 N. Ans.**

Problem 13.9 *Air is flowing over a smooth plate with a velocity of 10 m/s. The length of the plate is 1.2 m and width 0.8 m. If laminar boundary layer exists up to a value of $R_e = 2 \times 10^5$, find the maximum distance from the leading edge upto which laminar boundary layer exists. Find the maximum thickness of laminar boundary layer if the velocity profile is given by*

$$\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$$

Take kinematic viscosity for air = 0.15 stokes.

Solution. Given :
Velocity of air, U = 10 m/s
Length of plate, L = 1.2 m
Width of plate, b = 0.8 m
Reynold number upto which laminar boundary exists = 2×10^5
ν for air = 0.15 stokes = 0.15×10^{-4} m^2/s

Reynold number $R_{e_x} = \frac{\rho U x}{\mu} = \frac{U x}{\nu}$

If $R_{e_x} = 2 \times 10^5$, then x denotes the distance from leading edge upto which laminar boundary layer exists

$\therefore \quad 2 \times 10^5 = \frac{10 \times x}{0.15 \times 10^{-4}}$

$\therefore \quad x = \frac{2 \times 10^5 \times 0.15 \times 10^{-4}}{10} = 0.30 \text{ m} = \mathbf{300 \text{ mm. Ans.}}$

Maximum thickness of the laminar boundary for the velocity profile, $\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$ is given by equation (13.17) as

$$\delta = \frac{5.48 \times x}{\sqrt{R_{e_x}}} = \frac{5.48 \times 0.30}{\sqrt{2 \times 10^5}} = 0.00367 \text{ m} = \mathbf{3.67 \text{ mm. Ans.}}$$

Problem 13.10 *Air is flowing over a flat plate 500 mm long and 600 mm wide with a velocity of 4 m/s. The kinematic viscosity of air is given as 0.15 $\times 10^{-4}$ m^2/s. Find (i) the boundary layer thickness at the end of the plate, (ii) Shear stress at 200 mm from the leading edge and (iii) drag force on one side of the plate. Take the velocity profile over the plate as* $\frac{u}{U} = \sin\left(\frac{\pi}{2} \cdot \frac{y}{\delta}\right)$ *and density of air 1.24 kg/m^3.*

Solution. Given :

Length of plate, $L = 500 \text{ mm} = 0.5 \text{ m}$
Width of plate, $b = 600 \text{ mm} = 0.6 \text{ m}$
Velocity of air, $U = 4 \text{ m/s}$
Kinematic viscosity, $\nu = 0.15 \times 10^{-4} \text{ m}^2/\text{s}$
$\therefore$ Mass density, $\rho = 1.24 \text{ kg/m}^3$

For the velocity profile $\frac{u}{U} = \sin\left(\frac{\pi}{2}\frac{y}{\delta}\right)$, we have

(*i*) Boundary layer thickness at the end of the plate means value of δ at $x = 0.5$ m. First find Reynold number.

$$R_{e_x} = \frac{\rho U x}{\mu} = \frac{U x}{\nu} = \frac{4 \times 0.5}{0.15 \times 10^{-4}} = 1.33 \times 10^5.$$

Hence boundary layer is laminar over the entire length of the plate as Reynold number at the end of the plate is 1.33×10^5.

$\therefore$ δ at $x = 0.5$ m for the given velocity profile is given by equation (13.32) as

$$\delta = \frac{4.795x}{\sqrt{R_{e_x}}} = \frac{4.795 \times 0.5}{\sqrt{1.33 \times 10^5}} = 0.00656 \text{ m} = \mathbf{6.56 \text{ mm.}} \quad \textbf{Ans.}$$

(*ii*) Shear stress at any distance from leading edge is given by $\tau_0 = 0.327 \frac{\mu U}{x}\sqrt{R_{e_x}}$

At x = 200 mm = 0.2 m, $R_{e_x} = \dfrac{U \times x}{\nu} = \dfrac{4 \times 0.2}{0.15 \times 10^{-4}} = 53333$

$$\therefore \quad \tau_0 = \frac{0.327 \times \mu \times 4 \times \sqrt{53333}}{0.2}$$

But $$\mu = \nu \times \rho \qquad \left\{\because \quad \nu = \frac{u}{\rho}, \quad \therefore \quad \mu = \nu \times \rho\right\}$$

$$= 0.15 \times 10^{-4} \times 1.24 = 0.186 \times 10^{-4}$$

$$\therefore \quad \tau_0 = \frac{0.327 \times 0.186 \times 10^{-4} \times 4 \times \sqrt{53333}}{0.2} = \mathbf{0.02805\ N/m^2.\ Ans.}$$

(*iii*) Drag force on one side of the plate is given by equation (13.33)

$$F_D = 0.655 \times \mu U \times b \times \sqrt{\frac{\rho UL}{\mu}}$$

$$= 0.655 \times 0.186 \times 10^{-4} \times 4.0 \times 0.6 \times \sqrt{\frac{UL}{\nu}} \qquad \left\{\because \quad \nu = \frac{\mu}{\rho}\right\}$$

$$= 0.29234 \times 10^{-4} \times \sqrt{\frac{4 \times 0.5}{.15 \times 10^{-4}}} = \mathbf{0.01086\ N.\ Ans.}$$

Problem 13.11 *A thin plate is moving in still atmospheric air at a velocity of 5 m/s. The length of the plate is 0.6 m and width 0.5 m. Calculate (i) the thickness of the boundary layer at the end of the plate, and (ii) drag force on one side of the plate. Take density of air as 1.24 kg/m^3 and kinematic viscosity 0.15 stokes.*

Solution. Given :

Velocity of plate, U = 5 m/s
Length of plate, L = 0.6 m
Width of plate, b = 0.5 m
Density of air, ρ = 1.24 kg/m^3
Kinematic viscosity, ν = 0.15 stokes = 0.15×10^{-4} m^2/s

Reynold number, $R_e = \dfrac{UL}{\nu} = \dfrac{5 \times 0.6}{0.15 \times 10^{-4}} = 200000.$

As R_e is less than 5×10^5, hence boundary layer is laminar over the entire length of the plate.

(*i*) Thickness of boundary layer at the end of the plate by Blasius's solution is

$$\delta = \frac{4.91x}{\sqrt{R_{e_x}}} = \frac{4.91\,L}{\sqrt{R_{e_1}}} = \frac{4.91 \times 0.6}{\sqrt{200000}} = .00658 \text{ m} = \mathbf{6.58\ mm.\ Ans.}$$

(*ii*) Drag force on one side of the plate is given by equation (13.14) as

$$C_D = \frac{F_D}{\frac{1}{2}\rho AU^2}$$

$$\therefore \quad F_D = \frac{1}{2}\rho\, AU^2 \times C_D$$

where C_D from Blasius's solution, $C_D = \dfrac{1.328}{\sqrt{R_{e_L}}} = \dfrac{1.328}{\sqrt{200000}} = 0.002969 \simeq .00297$

$$\therefore \quad F_D = \frac{1}{2} \times 1.24 \times 0.6 \times 0.5 \times 5^2 \times .002970 \qquad \{\because A = L \times b = 0.6 \times 0.5\}$$

$$= \mathbf{0.01373\ N.\ Ans.}$$

Note. If no velocity profile is given in the numerical problem but boundary layer is laminar, then Blasius's solution is used.

Problem 13.12 *A plate of 600 mm length and 400 mm wide is immersed in a fluid of sp. gr. 0.9 and kinematic viscosity (ν =) 10^{-4} m^2/s. The fluid is moving with a velocity of 6 m/s. Determine (i) boundary layer thickness, (ii) shear stress at the end of the plate, and (iii) drag force on one side of the plate.*

Solution. As no velocity profile is given in the above problem, hence Blasius's solution will be used.

Given : length of plate, $L = 600$ mm $= 0.60$ m
Width of plate, $b = 400$ mm $= 0.40$ m
Sp. gr. of fluid, $S = 0.9$
$\therefore$ Density, $\rho = 0.9 \times 1000 = 900$ kg/m^3
Velocity of fluid, $U = 6$ m/s
Kinematic viscosity, $\nu = 10^{-4}$ m^2/s

Reynold number, $R_{e_L} = \dfrac{U \times L}{\nu} = \dfrac{6 \times 0.6}{10^{-4}} = 3.6 \times 10^4$.

As R_{e_L} is less than 5×10^5, hence boundary layer is laminar over the entire length of the plate.

(*i*) Thickness of boundary layer at the end of the plate from Blasius's solution is

$$\delta = \frac{4.91\,x}{\sqrt{R_{e_x}}}, \text{ where } x = 0.6 \text{ m and } R_{e_x} = 3.6 \times 10^4$$

$$= \frac{4.91 \times 0.6}{\sqrt{3.6 \times 10^4}} = 0.0155 \text{ m} = \mathbf{15.5\ mm.\ Ans.}$$

(*ii*) Shear stress at the end of the plate is

$$\tau_0 = 0.332\ \frac{\rho U^2}{\sqrt{R_{e_L}}} = \frac{0.332 \times 900 \times 6^2}{\sqrt{3.6 \times 10^4}} = \mathbf{56.6\ N/m^2.\ Ans.}$$

(*iii*) Drag force (F_D) on one side of the plate is given by

$$F_D = \frac{1}{2} \rho\, AU^2 \times C_D$$

where C_D from Blasius's solution is $C_D = \dfrac{1.328}{\sqrt{R_{e_L}}} = \dfrac{1.328}{\sqrt{3.6 \times 10^4}} = 0.00699$

$$\therefore \quad F_D = \frac{1}{2} \rho\, AU^2 \times C_D$$

$$= \frac{1}{2} \times 900 \times 0.6 \times 0.4 \times 6^2 \times .00699 \qquad \{\because A = L \times b = 0.6 \times .4\}$$

$$= \mathbf{26.78\ N.\ Ans.}$$

▶ 13.4 TURBULENT BOUNDARY LAYER ON A FLAT PLATE

The thickness of the boundary layer, drag force on one side of the plate and co-efficient of drag due to turbulent boundary layer on a smooth plate at zero pressure gradient are determined as in case of laminar boundary layer provided the velocity profile is known. Blasius on the basis of experiments give the following velocity profile for turbulent boundary layer

$$\frac{u}{U} = \left(\frac{y}{\delta}\right)^n \qquad ...(13.35)$$

where $n = \frac{1}{7}$ for $R_e < 10^7$ but more than 5×10^5

$$\therefore \qquad \frac{u}{U} = \left(\frac{y}{\delta}\right)^{1/7} \qquad ...(13.36)$$

Equation (13.36) is not applicable very near the boundary, where the thin laminar sub-layer of thickness δ' exists. Here velocity distribution is influenced only by viscous effects.

The value of τ_0 for flat plate is taken as $\tau_0 = 0.0225\ \rho\ U^2 \left(\frac{\mu}{\rho \delta U}\right)^{1/4}$ ...(13.37)

Problem 13.13 *For the velocity profile for turbulent boundary layer* $\frac{u}{U} = \left(\frac{y}{\delta}\right)^{1/7}$, *obtain an expression for boundary layer thickness, shear stress, drag force on one side of the plate and co-efficient of drag in terms of Reynold number. Given the shear stress* (τ_0) *for turbulent boundary layer as*

$$\tau_0 = 0.0225\ \rho\ U^2 \left(\frac{\mu}{\rho U \delta}\right)^{1/4}.$$

Solution. Given : $\frac{u}{U} = \left(\frac{y}{\delta}\right)^{1/7}$

(*i*) Substituting this value in Von Karman momentum integral equation (13.10),

$$\frac{\tau_0}{\rho U^2} = \frac{\partial}{\partial x}\left[\int_0^\delta \frac{u}{U}\left(1 - \frac{u}{U}\right) dy\right]$$

$$= \frac{\partial}{\partial x}\left[\int_0^\delta \left(\frac{y}{\delta}\right)^{1/7}\left[1 - \left(\frac{y}{\delta}\right)^{1/7}\right] dy\right] = \frac{\partial}{\partial x}\left[\int_0^\delta \left(\frac{y^{1/7}}{\delta^{1/7}} - \frac{y^{2/7}}{\delta^{2/7}}\right) dy\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{y^{1/7+1}}{\left(\frac{1}{7}+1\right)\delta^{1/7}} - \frac{y^{2/7+1}}{\left(\frac{2}{7}+1\right)\delta^{2/7}}\right]_0^\delta$$

$$= \frac{\partial}{\partial x}\left[\frac{7}{8}\frac{y^{8/7}}{\delta^{1/7}} - \frac{7}{9}\frac{y^{9/7}}{\delta^{2/7}}\right]_0^\delta = \frac{\partial}{\partial x}\left[\frac{7}{8}\frac{\delta^{8/7}}{\delta^{1/7}} - \frac{7}{9}\frac{\delta^{9/7}}{\delta^{2/7}}\right]$$

$$= \frac{\partial}{\partial x}\left[\frac{7}{8}\delta - \frac{7}{9}\delta\right] = \frac{\partial}{\partial x}\left[\frac{63-56}{72}\right]\delta = \frac{\partial}{\partial x}\left[\frac{7}{72}\right]\delta = \frac{7}{72}\frac{\partial \delta}{\partial x}$$

In the above expression, the integration limits should be from δ' to δ. But as the laminar sub-layer is very thin that is δ' is very small. Hence the limits of integration are taken from 0 to δ.

Now $$\tau_0 = \frac{7}{72}\rho U^2 \frac{\partial \delta}{\partial x} \qquad ...(13.38)$$

But the value of τ_0 for turbulent boundary layer is given,

$$\tau_0 = 0.0225\,\rho U^2 \left(\frac{\mu}{\rho U \delta}\right)^{1/4} \qquad ...(13.39)$$

Equating the two values of τ_0 given by equations (13.38) and (13.39), we have

$$\frac{7}{72}\rho U^2 \frac{\partial \delta}{\partial x} = .0225\,\rho U^2 \left(\frac{\mu}{\rho U \delta}\right)^{1/4}$$

or $$\frac{7}{72}\frac{\partial \delta}{\partial x} = .0225\left(\frac{\mu}{\rho U}\right)^{1/4} \times \frac{1}{\delta^{1/4}} \qquad \{\text{cancelling } \rho U^2\}$$

or $$\delta^{1/4}\partial\delta = .0225 \times \frac{72}{7} \times \left(\frac{\mu}{\rho U}\right)^{1/4} \partial x = 0.2314\left(\frac{\mu}{\rho U}\right)^{1/4} \partial x.$$

Integrating, we get $$\frac{\delta^{1/4+1}}{\left(\frac{1}{4}+1\right)} = 0.2314\left(\frac{\mu}{\rho U}\right)^{1/4} x + C$$

or $$\frac{4}{5} \times \delta^{5/4} = 0.2314\left(\frac{\mu}{\rho U}\right)^{1/4} x + C$$

where C is constant of integration.

To determine the value of C, assume turbulent boundary layer starts from the leading edge, though in actual practice the turbulent boundary layer starts after the transition from laminar boundary layer. The laminar layer exists for a very short distance and hence this assumption will not affect the subsequent analysis.

Hence at $x = 0$, $\delta = 0$ and so $C = 0$

$\therefore$ $$\frac{4}{5}\delta^{5/4} = 0.2314\left(\frac{\mu}{\rho U}\right)^{1/4} x \text{ or } \delta^{5/4} = \frac{0.2314 \times 5}{4}\left(\frac{\mu}{\rho U}\right)^{1/4} x$$

or $$\delta = \left[\frac{0.2314 \times 5}{4}\left(\frac{\mu}{\rho U}\right)^{1/4} x\right]^{4/5} = \left(\frac{0.2314 \times 5}{4}\right)^{4/5}\left(\frac{\mu}{\rho U}\right)^{1/5} x^{4/5}$$

$$= 0.37\left(\frac{\mu}{\rho U}\right)^{1/5} x^{4/5} \qquad ...(13.40)$$

$$= 0.37\left(\frac{\mu}{\rho U x}\right)^{1/5} x^{1/5} \times x^{4/5} = 0.37\left(\frac{1}{R_{e_x}}\right)^{1/5} \times x = \frac{0.37x}{\left(R_{e_x}\right)^{1/5}} \quad ...(13.41)$$

From equation (13.40), it is clear that δ varies as $x^{4/5}$ in turbulent boundary layer while in case of laminar boundary layer δ varies as $\sqrt{x}$.

(*ii*) **Shear Stress (τ_0)** at any point from leading edge is given by equation (13.40) as

$$\tau_0 = 0.225\ \rho U^2 \left(\frac{\mu}{\rho U \delta}\right)^{1/4}$$

Substituting the value of δ from equation (13.40), we have

$$\tau_0 = 0.0225\ \rho U^2 \left(\frac{\mu}{\rho U \times 0.37 \times \left(\frac{\mu}{\rho U}\right)^{1/5} \times x^{4/5}}\right)^{1/4}$$

$$= \frac{.0225 \times 2}{2} \rho U^2 \left(\frac{\mu^{4/5}}{0.37 \times (\rho U)^{4/5} \times x^{4/5}}\right)^{1/4}$$

$$= .0225 \times 2 \times \frac{\rho U^2}{2} \times \frac{1}{(0.37)^{1/4}} \left(\frac{\mu}{\rho U x}\right)^{1/5}$$

$$= 0.0577 \times \frac{\rho U^2}{2} \left(\frac{\mu}{\rho U x}\right)^{1/5} \quad ...(13.42)$$

(*iii*) **Drag force (F_D)** on one side of the plate is

$$F_D = \int_0^L \tau_0 \times b \times dx = \int_0^L 0.0577 \times \frac{\rho U^2}{2} \times \left(\frac{\mu}{\rho U}\right)^{1/5} \frac{1}{x^{1/5}} \times b \times dx$$

$$= 0.0577 \times \frac{\rho U^2}{2} \times \left(\frac{\mu}{\rho U}\right)^{1/5} \times b \int_0^L x^{-1/5}\, dx$$

$$= .0577 \times \frac{\rho U^2}{2} \times \left(\frac{\mu}{\rho U}\right)^{1/5} \times b \times \left[\frac{x^{4/5}}{4/5}\right]_0^L$$

$$= .0577 \times \frac{5}{4} \times \frac{\rho U^2}{2} \left(\frac{\mu}{\rho U}\right)^{1/5} \times b \times L^{4/5}$$

$$= 0.072 \times \frac{\rho U^2}{2} \times \left(\frac{\mu}{\rho U}\right)^{1/5} \times b \times L^{4/5}$$

(*iv*) **Drag co-efficient, C_D** is given by

$$C_D = \frac{F_D}{\frac{1}{2}\rho A U^2}, \text{ where } A = L \times b$$

$$= \frac{.072 \times \frac{\rho U^2}{2} \times \left(\frac{\mu}{\rho U}\right)^{1/5} \times b \times L^{4/5}}{\frac{\rho U^2}{2} \times b \times L}$$

$$= 0.072 \times \left(\frac{\mu}{\rho U}\right)^{1/5} \cdot \frac{1}{L^{1/5}} = 0.072 \left(\frac{\mu}{\rho UL}\right)^{1/5}$$

$$= \frac{.072}{R_{e_L}^{1/5}} \qquad ...(13.43) \left\{\because \quad R_{e_L} = \frac{\rho UL}{\mu}\right\}$$

This is valid for $R_{e_L} > 5 \times 10^5$ but less than 10^7.

▶ 13.5 ANALYSIS OF TURBULENT BOUNDARY LAYER

(*a*) If Reynold number is more than 5×10^5 and less than 10^7 the thickness of boundary layer and drag co-efficient are given as :

$$\delta = \frac{0.37x}{\left(R_{e_x}\right)^{1/5}} \text{ and } C_D = \frac{0.072}{\left(R_{e_L}\right)^{1/5}} \qquad ...(13.44)$$

where x = Distance from the leading edge

R_{e_x} = Reynold number for length x

R_{e_L} = Reynold number at the end of the plate.

(*b*) If Reynold number is more than 10^7 but less than 10^9, Schlichting gave the empirical equation as

$$C_D = \frac{0.455}{\left(\log_{10} R_{e_L}\right)^{2.58}} \qquad ...(13.44A)$$

▶ 13.6 TOTAL DRAG ON A FLAT PLATE DUE TO LAMINAR AND TURBULENT BOUNDARY LAYER

Consider the flow over a flat plate as shown in Fig. 13.5.

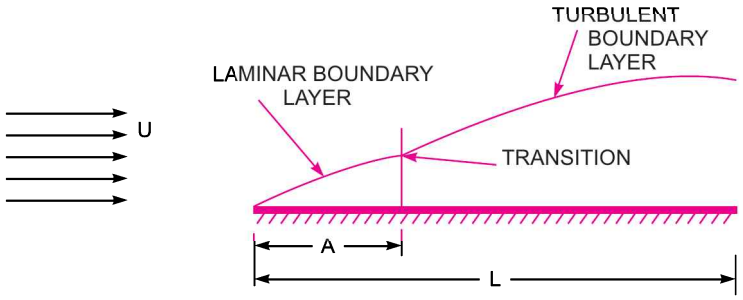


Fig. 13.5 *Drag due to laminar and turbulent boundary layer.*

Let L = Total length of the plate, b = Width of plate,
A = Length of laminar boundary layer

If the length of transition region is assumed negligible, then

$L - A$ = Length of turbulent boundary layer.

We have obtained the drag on a flat plate for the laminar as well as turbulent boundary layer on the assumption that turbulent boundary layer starts from the leading edge. This assumption is valid only when the length of laminar boundary layer is negligible. But if the length of laminar boundary layer is not negligible, then the total drag on the plate due to laminar and turbulent boundary layer is calculated as :

(1) Find the length from the leading edge upto which laminar boundary layer exists. This is done by equating $5 \times 10^5 = \frac{Ux}{\nu}$. The value of x gives the length of laminar boundary layer. Let this length is equal to A.

(2) Find drag using Blasius solution for laminar boundary layer for length A.

(3) Find drag due to turbulent boundary layer for the whole length of the plate.

(4) Find the drag due to turbulent boundary layer for a length A only

Then total drag on the plate

= Drag given by (2) + Drag given by (3) – Drag given by (4)

= Drag due to laminar boundary layer for length A
+ Drag due to turbulent boundary layer for length L
– Drag due to turbulent boundary layer for length A. ...(13.45)

Problem 13.14 (S.I. Units). *Determine the thickness of the boundary layer at the trailing edge of smooth plate of length 4 m and of width 1.5 m, when the plate is moving with a velocity of 4 m/s in stationary air. Take kinematic viscosity of air as 1.5×10^{-5} m^2/s.*

Solution. Given :

Length of plate, $L = 4$ m

Width of plate, $b = 1.5$ m

Velocity of plate, $U = 4$ m/s

Kinematic viscosity, $\nu = 1.5 \times 10^{-5}$ m^2/s

Reynold number, $R_{e_L} = \frac{U \times L}{\nu} = \frac{4.0 \times .40}{1.5 \times 10^{-5}} = 10.66 \times 10^5$

As the Reynold number is more than 5×10^5 and hence the boundary layer at the trailing edge is turbulent.

The boundary layer thickness for turbulent boundary layer is given by equation (13.44) as

$$\delta = \frac{0.37x}{\left(R_{e_x}\right)^{1/5}} \qquad \text{| Here } x = L \text{ and } R_{e_x} = R_{e_L}$$

$$= \frac{0.37 \times 4.0}{\left(10.66 \times 10^5\right)^{1/5}} = 0.0921 \text{ m} = \mathbf{92.1\ mm.\ Ans.}$$

Problem 13.15 *In Problem 13.14, determine the total drag on one side of the plate assuming that (i) the boundary layer is laminar over the entire length of the plate and (ii) the boundary layer is turbulent from the very beginning. Take ρ for air = 1.226 kg/m^3.*

Solution. The data of problem 13.14,

$L = 4$ m, $b = 1.5$ m, $U = 4$ m/s,

$\nu = 1.5 \times 10^{-5}$ m^2/s

$R_{e_L} = 10.66 \times 10^5$ and $\rho = 1.226$ kg/m^3.

(*i*) **When the boundary layer is laminar over the entire length**, the value of C_D is given by Blasius's solution as

$$C_D = \frac{1.328}{\sqrt{R_{e_L}}} = \frac{1.328}{\sqrt{10.66 \times 10^5}} = .001286$$

$\therefore$ Drag force (F_D) on one side of the plate is

$$F_D = \frac{1}{2} \rho A U^2 \times C_D$$

where $A = b \times L = 1.5 \times 4 = 6.0$ m^2

$$= \frac{1}{2} \times 1.226 \times 6.0 \times 4^2 \times .001286 = \textbf{0.0757 N. Ans.}$$

(*ii*) **When the boundary layer is turbulent from the very beginning**, the value of co-efficient of drag, C_D is given by equation (13.43) as

$$C_D = \frac{0.072}{\left(R_{e_L}\right)^{1/5}} = \frac{0.072}{\left(10.66 \times 10^5\right)^{1/5}} = .00448$$

$\therefore$ Drag force, $F_D = \frac{1}{2} \rho A U^2 \times C_D$

$$= \frac{1}{2} \times 1.226 \times 6.0 \times 4^2 \times .00448 \qquad \{\because A = b \times L = 1.5 \times 4 = 6 \text{ m}^2\}$$

$$= \textbf{0.2637 N. Ans.}$$

Problem 13.16 *Water is flowing over a thin smooth plate of length 4 m and width 2 m at a velocity of 1.0 m/s. If the boundary layer flow changes from laminar to turbulent at a Reynold number 5×10^5, find (i) the distance from leading edge upto which boundary layer is laminar, (ii) the thickness of the boundary layer at the transition point, and (iii) the drag force on one side of the plate. Take viscosity of water $\mu = 9.81 \times 10^{-4}$ Ns/m^2.*

Solution. Given :

Length of plate, $L = 4$ m

Width of plate, $b = 2$ m

Velocity of flow, $U = 1.0$ m/s

Reynold number for laminar boundary layer = 5×10^5

Viscosity of water, $\mu = 9.81 \times 10^{-4} \frac{\text{Ns}}{\text{m}^2}$

(*i*) Let the distance from leading edge upto which laminar boundary layer exists = x

$\therefore$ $$5 \times 10^5 = \frac{\rho U x}{\mu} = 1000 \times \frac{1.0 \times x}{9.81 \times 10^{-4}} \qquad (\because \rho = 1000)$$

$\therefore$ $$x = \frac{5 \times 10^5 \times 9.81 \times 10^{-4}}{1000} = 0.4900 \text{ m} = \textbf{490 mm. Ans.}$$

(*ii*) Thickness of boundary layer at the point where the boundary layer changes from laminar to turbulent *i.e.*, at Reynold number = 5×10^5, is given by Blasius's solution as

$$\delta = \frac{4.91 \times x}{\sqrt{R_{e_x}}} \qquad \text{| Here } x = 49 \text{ cm} = 0.49 \text{ m}, \; R_{e_x} = 5 \times 10^5$$

$$\delta = \frac{4.91 \times 0.49}{\sqrt{5 \times 10^5}} = 0.0034 \text{ m} = \textbf{3.4 mm. Ans.}$$

(*iii*) Drag force on the plate on one side

= Drag due to laminar boundary layer + Drag due to turbulent boundary.

(*a*) Drag due to laminar boundary layer (*i.e.*, from *E* to *F*)

$$F_{EF} = \frac{1}{2} \rho A U^2 \times C_D \qquad ...(i)$$

where C_D is given by Blasius solution for laminar boundary layer as

$$C_D = \frac{1.328}{\sqrt{R_{e_x}}}, \text{ where for } EF, \; R_{e_x} = 5 \times 10^5$$

$$= \frac{1.328}{\sqrt{5 \times 10^5}} = 0.001878$$

A = Area of plate upto laminar boundary layer

$= 0.49 \times b = 0.49 \times 2 = 0.98 \text{ m}^2$

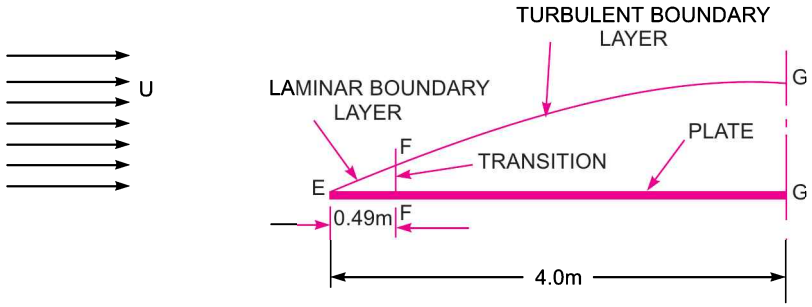


Fig. 13.6 (*a*)

Substituting the value of C_D and A in equation (*i*), we get

$$F_{EF} = \frac{1}{2} \times 1000 \times 0.98 \times 1.0^2 \times .001878 = \textbf{0.92 N}. \qquad ...(ii)$$

(*b*) Drag force due to turbulent boundary layer from *F* to *G*

= Drag force due to turbulent boundary layer from *E* to *G*

– Drag force due to turbulent flow from *E* to *F*

$$= (F_{EG})_{\text{turb.}} - (F_{EF})_{\text{turb.}}$$

Now $$(F_{FG})_{\text{turb.}} = \frac{1}{2} \rho A U^2 \times C_D$$

where C_D from equation (13.44) is $C_D = \dfrac{0.072}{\left(R_{e_L}\right)^{1/5}}$

But $$R_{e_L} = \frac{\rho UL}{\mu} = 1000 \times \frac{1.0 \times 4.0}{9.81 \times 10^{-4}} = 40.77 \times 10^5$$

$$\therefore \quad C_D = \frac{0.072}{\left(40.77 \times 10^5\right)^{1/5}} = 0.00343$$

$$\therefore \quad (F_{EG})_{\text{turb.}} = \frac{1}{2} \rho A U^2 \times C_D = \frac{1}{2} \times 1000 \times (4 \times 2) \times 1^2 \times .00343 = \mathbf{13.72\ N}$$

Also $$(F_{EF})_{\text{turb.}} = \frac{1}{2} \rho A_{EF} \times U^2 \times C_D$$

where A_{EF} = Area of plate upto $EF = EF \times b = 0.49 \times 2 = 0.98\ \text{m}^2$

and $$C_D = \frac{0.072}{(R_{EF})^{1/5}} = \frac{0.072}{\left(5 \times 10^5\right)^{1/5}} = .00522$$

$$(F_{EF})_{\text{turb.}} = \frac{1}{2} \times 1000 \times 0.98 \times 1^2 \times .00522 = \mathbf{2.557\ N}$$

$\therefore$ Drag force due to turbulent boundary layer from F to G

$$= (F_{EG})_{\text{turb.}} - (F_{EF})_{\text{turb.}} = 13.72 - 2.557 = 11.163\ \text{N}$$

$\therefore$ Drag force on the plate on one side

= Drag force due to laminar boundary layer upto F

+ Drag due to turbulent boundary layer from F to G

$= 0.92 + 11.163 =$ **12.083 N. Ans.**

Problem 13.16 (A) *Air flows at 10 m/s past a smooth rectangular flat plate 0.3 m wide and 3 m long. Assuming that the turbulence level in the oncoming stream is low and that transition ocurs at $R_e = 5 \times 10^5$, calculate ratio of total drag when the flow is parallel to the length of the plate to the value when the flow is parallel to the width.* (R.G.P.V., Bhopal S 2001)

Solution. Given :

$$U = 10\ \text{m/s} ; \ b = 0.3\ \text{m} ; \ L = 3\ \text{m} ;$$

Reynolds number for laminar B.L. $= 5 \times 10^5$.

The kinematic viscosity of air and density of air may be assumed as their values are not given in the question. Take $\rho = 1.24\ \text{kg/m}^2$ and $\nu = 0.15$ stoke

$\therefore$ $\rho = 1.24\ \text{kg/m}^3$ and $\nu = 0.15$ stoke $= 0.15 \times 10^{-4}\ \text{m}^2/\text{s}$.

(*i*) **Drag when flow is parallel to the length of the plate**

Let x = the distance from leading edge upto which laminar boundary exists

$$\therefore \quad 5 \times 10^5 = \frac{\rho \times U \times x}{\mu} = \frac{U \times x}{\nu} = \frac{10 \times x}{0.15 \times 10^{-4}}$$

$$\therefore \quad x = \frac{5 \times 10^5 \times 0.15 \times 10^{-4}}{10} = 0.75\ \text{m}$$

Now the drag force on the plate on one side

= Drag due to laminar boundary layer + Drag due to turbulent boundary layer ...(*i*)

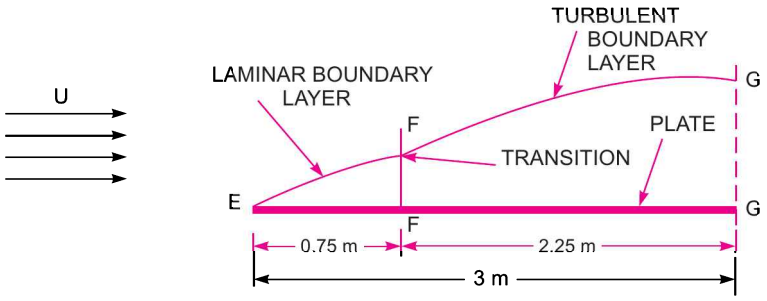


Fig. 13.6 (*b*)

(*a*) **Drag due to laminar boundary layer (*i.e.*, from E to F)**

$$F_{EF} = \frac{1}{2}\rho A U^2 \times C_D$$

where C_D is given by Blasius solution for laminar boundary layer as

$$C_D = \frac{1.328}{\sqrt{R_{e_x}}}, \text{ where } R_{ex} = 5 \times 10^5$$

$$= \frac{1.328}{\sqrt{5 \times 10^5}} = 0.001878$$

A = Area of plate upto laminar boundary layer

$= 0.75 \times b = 0.75 \times 0.3 = 0.225 \text{ m}^2$

$\rho = 1.24 \text{ kg/m}^3$

$$\therefore \quad F_{EF} = \frac{1}{2} \times 1.24 \times 0.225 \times 10^2 \times 0.001878 = 0.0262 \text{ N}$$

(*b*) **Drag force due to turbulent boundary layer from F to G**

= Drag force due to turbulent boundary layer from E to G – Drag force due to turbulent B.L. from E to F

$$= (F_{EG})_{\text{turb.}} - (F_{EF})_{\text{turb.}}$$

Now $$(F_{EG})_{\text{turb.}} = \frac{1}{2}\rho A U^2 \times C_D$$

where C_D for turbulent boundary layer is given by equation (13.44) as

$$C_D = \frac{0.072}{\left(R_{e_L}\right)^{1/5}}$$

But $$R_{e_L} = \frac{U \times L}{\nu} = \frac{10 \times 3}{0.15 \times 10^{-4}} = 20 \times 10^5$$

$$\therefore \quad C_D = \frac{0.072}{\left(20 \times 10^5\right)^{1/5}} = 0.00395$$

$$\therefore \quad (F_{EG})_{\text{turb.}} = \frac{1}{2}\rho A U^2 \times C_D = \frac{1}{2} \times 1.24 \times (3 \times 0.3) \times 10^2 \times 0.00395$$

$$= \mathbf{0.2204 \text{ N}}$$

Now $$(F_{EF})_{\text{turb.}} = \frac{1}{2}\rho \times A_{EF} \times U^2 \times C_D$$

where A_{EF} = Area of plate upto $EF = EF \times b = 0.75 \times 0.3 = 0.225\ \text{m}^2$

and $$C_D = \frac{0.072}{\left[(R_e)_{EF}\right]^{1/5}} = \frac{0.072}{\left(5 \times 10^5\right)^{1/5}} = 0.00522$$

$\therefore$ $$(F_{EF})_{\text{turb.}} = \frac{1}{2} \times 1.24 \times 0.225 \times 10^2 \times 0.00522 = \mathbf{0.0728\ N}$$

$\therefore$ Drag force due to turbulent boundary layer from F to G

$= (F_{EG})_{\text{turb.}} - (F_{EF})_{\text{turb.}} = 0.2204 - 0.0728 = \mathbf{0.1476\ N}$

$\therefore$ Total drag force when flow is parallel to the length of the plate

= Drag due to laminar boundary layer upto F

+ Drag due to turbulent boundary layer from F to G

$= 0.0262 + 0.1476 = \mathbf{0.1738\ N}$ ...(*A*)

(*ii*) **Drag when flow is parallel to the width of the plate**

We have already calculated that upto the length of 0.75 m from the leading edge, the boundary layer is laminar. As the width of the plate is only 0.3 m, hence when flow is parallel to the width of the plate, only laminar boundary layer will be formed.

$\therefore$ Drag force on the plate

$$= \frac{1}{2}\rho A U^2 \times C_D$$

where C_D from Blasius solution for laminar boundary layer is given as

$$C_D = \frac{1.328}{\sqrt{R_{e_x}}}, \text{ here } \quad x = \text{width of plate} = 0.3 \text{ m hence}$$

$$R_{e_x} = \frac{U \times x}{\nu} = \frac{10 \times 0.3}{0.15 \times 10^{-4}} = 2 \times 10^5$$

$$= \frac{1.328}{\sqrt{2 \times 10^5}} = 0.00297$$

A = Area of plate upto width (0.3 m) $= 3 \times 0.3 = 0.9$

$\rho = 1.24\ \text{kg/m}^3$

$\therefore$ Total drag on the plate $$= \frac{1}{2} \times 1.24 \times 0.9 \times 10^2 \times 0.00297$$

$= \mathbf{0.1657\ N}$ ...(*B*)

$\therefore$ Ratio of two total drags given by equations (*A*) and (*B*) becomes as

$$\frac{\text{Total drag when flow is parallel to the length of the plate}}{\text{Total drag when flow is parallel to the width of the plate}} = \frac{\text{Equation } (A)}{\text{Equation } (B)} = \frac{0.1738}{0.1657} = \mathbf{1.05.\ Ans.}$$

Problem 13.17 *Oil with a free-stream velocity of 2 m/s flows over a thin plate 2 m wide and 2 m long. Calculate the boundary layer thickness and the shear stress at the trailing end point and determine the total surface resistance of the plate. Take specific gravity as 0.86 and kinematic viscosity as 10^{-5} m^2/s.*

Solution. Given :

Free-stream velocity of oil, $U = 2$ m/s

Width of plate, $b = 2$ m

Length of plate, $L = 2$ m

$\therefore$ Area of plate, $A = b \times L = 2 \times 2 = 4$ m^2

Specific gravity of oil, $S = 0.86$

$\therefore$ Density of oil, $\rho = 0.86 \times 1000 = 860$ kg/m^3

Kinematic viscosity, $\nu = 10^{-5}$ m^2/s

Now the Reynold number at the trailing end,

$$R_{e_L} = \frac{UL}{\nu} = \frac{2 \times 2}{10^{-5}} = 4 \times 10^5.$$

Since R_{e_L} is less than 5×10^5, the boundary layer is laminar over the entire length of the plate.

$\therefore$ Thickness of boundary layer at the end of the plate from Blasius's solution is,

$$\delta = \frac{4.91 \times L}{\sqrt{R_{e_L}}} = \frac{4.91 \times 2.0}{\sqrt{4 \times 10^5}} = 0.0155 \text{ m} = \textbf{15.5 mm. Ans.}$$

Shear stress at the end of the plate is, $\tau_0 = 0.332 \times$ = **1.805 N/m^2. Ans.**

Surface resistance on one side of the plate is given by

$$F_D = \frac{1}{2} \rho A U^2 \times C_D$$

where $C_D = \dfrac{1.328}{\sqrt{R_{e_L}}} = \dfrac{1.328}{\sqrt{4 \times 10^5}} = 0.0021$

$\therefore$ $$F_D = \frac{1}{2} \times 860 \times 4.0 \times 2^2 \times .0021 = 14.44 \text{ N}$$

$\therefore$ Total resistance $= 2 \times F_D = 2 \times 14.44 =$ **28.88 N. Ans.**

▶ 13.7 SEPARATION OF BOUNDARY LAYER

When a solid body is immersed in a flowing fluid, a thin layer of fluid called the boundary layer is formed adjacent to the solid body. In this thin layer of fluid, the velocity varies from zero to free-stream velocity in the direction normal to the solid body. Along the length of the solid body, the thickness of the boundary layer increases. The fluid layer adjacent to the solid surface has to do work against surface friction at the expense of its kinetic energy. This loss of the kinetic energy is recovered from the immediate fluid layer in contact with the layer adjacent to solid surface through momentum exchange process. Thus the velocity of the layer goes on decreasing. Along the length of the solid body, at a certain point a stage may come when the boundary layer may not be able to keep sticking to the solid body if it cannot provide kinetic energy to overcome the resistance offered by the solid body. In other words, the boundary layer will be separated from the surface. This phenomenon is called the boundary layer separation. The point on the body at which the boundary layer is on the verge of separation from the surface is called point of separation.

13.7.1 Effect of Pressure Gradient on Boundary Layer Separation. The effect of pressure gradient $\left(\dfrac{dp}{dx}\right)$ on boundary layer separation can be explained by considering the flow over a

curved surface *ABCSD* as shown in Fig. 13.7. In the region *ABC* of the curved surface, the area of flow decreases and hence velocity increases. This means that flow gets accelerated in this region. Due to the increase of the velocity, the pressure decreases in the direction of the flow and hence pressure gradient $\frac{dp}{dx}$ is negative in this region. As long as $\frac{dp}{dx} < 0$, the entire boundary layer moves forward as shown in Fig. 13.7.

Region *CSD* of the curved surface. The pressure is minimum at the point *C*. Along the region *CSD* of the curved surface, the area of flow increases and hence velocity of flow along the direction of fluid decreases. Due to decrease of velocity, the pressure increases in the direction of flow and hence pressure gradient $\frac{dp}{dx}$ is positive or $\frac{dp}{dx} > 0$. Thus in the region *CSD*, the pressure gradient is positive and velocity of fluid layer along the direction of flow decreases. As explained in the Art. 13.7, the velocity of the layer adjacent to the solid surface along the length of the solid surface goes on decreasing as the kinetic energy of the layer is used to overcome the frictional resistance of the surface. Thus the combined effect of positive pressure gradient and surface resistance reduce the momentum of the fluid is unable to the surface. A stage comes, when the momentum of the fluid is unable to overcome the surface resistance and the boundary layer starts separating from the surface at the point *S*. Downstream the point *S*, the flow is taking place in reverse direction and the velocity gradient becomes negative.

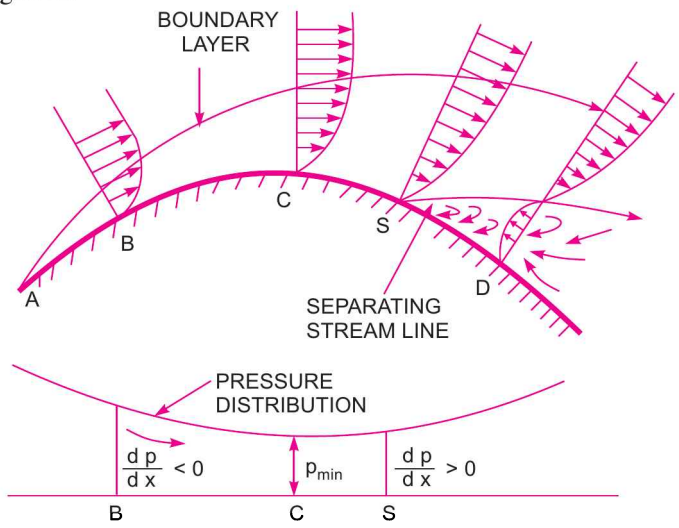


Fig. 13.7 *Effect of pressure gradient on boundary layer separation.*

Thus the positive pressure gradient helps in separating the boundary layer.

13.7.2 Location of Separation Point. The separation point *S* is determined from the condition,

$$\left(\frac{\partial u}{\partial y}\right)_{y=0} = 0 \qquad ...(13.46)$$

For a given velocity profile, it can be determined whether the boundary layer has separated, or on the verge of separation or will not separate from the following conditions :

1. If $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ is negative ... the flow has separated.

2. If $\left(\frac{\partial u}{\partial y}\right)_{y=0} = 0$ ... the flow is on the verge of separation.

3. If $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ is positive ... the flow will not separate or flow will remain attached with the surface.

Problem 13.18 *For the following velocity profiles, determine whether the flow has separated or on the verge of separation or will attach with the surface :*

(*i*) $\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3$, (*ii*) $\frac{u}{U} = 2\left(\frac{y}{\delta}\right)^2 - \left(\frac{y}{\delta}\right)^3$,

(*iii*) $\frac{u}{U} = -2\left(\frac{y}{\delta}\right) + \left(\frac{y}{\delta}\right)^2$.

Solution. Given :

1st Velocity Profile

$$\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3 \quad \text{or} \quad u = \frac{3U}{2}\left(\frac{y}{\delta}\right) - \frac{U}{2}\left(\frac{y}{\delta}\right)^3$$

Differentiating w.r.t. y, the above equation becomes,

$$\frac{\partial u}{\partial y} = \frac{3U}{2} \times \frac{1}{\delta} - \frac{U}{2} \times 3\left(\frac{y}{\delta}\right)^2 \times \frac{1}{\delta}$$

At $y = 0$, $\left(\frac{\partial u}{\partial y}\right)_{y=0} = \frac{3U}{2\delta} - \frac{3U}{2}\left(\frac{0}{\delta}\right)^2 \times \frac{1}{\delta} = \frac{3U}{2\delta}$.

As $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ is positive. Hence flow will not separate or flow will remain attached with the surface.

2nd Velocity Profile

$$\frac{u}{U} = 2\left(\frac{y}{\delta}\right)^2 - \left(\frac{y}{\delta}\right)^3$$

$\therefore$ $$u = 2U\left(\frac{y}{\delta}\right)^2 - U\left(\frac{y}{\delta}\right)^3$$

$\therefore$ $$\frac{\partial u}{\partial y} = 2U \times 2\left(\frac{y}{\delta}\right) \times \frac{1}{\delta} - U \times 3\left(\frac{y}{\delta}\right)^2 \times \frac{1}{\delta}$$

at $y = 0$, $\left(\frac{\partial u}{\partial y}\right)_{y=0} = 2U \times 2\left(\frac{0}{\delta}\right) \times \frac{1}{\delta} - U \times 3\left(\frac{0}{\delta}\right)^2 \times \frac{1}{\delta} = 0$

As $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ = 0, the flow is on the verge of separation. **Ans.**

3rd Velocity Profile

$$\frac{u}{U} = -2\left(\frac{y}{\delta}\right) + \left(\frac{y}{\delta}\right)^2$$

$$\therefore \quad u = -2U\left(\frac{y}{\delta}\right) + U\left(\frac{y}{\delta}\right)^2$$

$$\therefore \quad \frac{\partial u}{\partial y} = -2U\left(\frac{1}{\delta}\right) + 2U\left(\frac{y}{\delta}\right) \times \frac{1}{\delta}$$

at $y = 0$, $\left(\frac{\partial u}{\partial y}\right)_{y=0} = -\frac{2U}{\delta} + 2U\left(\frac{0}{\delta}\right) \times \frac{1}{\delta} = -\frac{2U}{\delta}$

As $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ is negative the flow has separated. **Ans.**

13.7.3 Methods of Preventing the Separation of Boundary Layer. When the boundary layer separates from the surface as shown in Fig. 13.7 at point *S*, a certain portion adjacent to the surface has a back flow and eddies are continuously formed in this region and hence continuous loss of energy takes place. Thus separation of boundary layer is undesirable and attempts should be made to avoid separation by various methods. The following are the methods for preventing the separation of boundary layer :

1. Suction of the slow moving fluid by a suction slot.
2. Supplying additional energy from a blower.
3. Providing a bypass in the slotted wing.
4. Rotating boundary in the direction of flow.
5. Providing small divergence in a diffuser.
6. Providing guide-blades in a bend.
7. Providing a trip-wire ring in the laminar region for the flow over a sphere.

HIGHLIGHTS

1. When a solid body is immersed in a flowing fluid, there is a narrow region of the fluid in the neighbourhood of the solid body, where the velocity of fluid varies from zero to free-stream velocity. This narrow region of fluid is called boundary layer.
2. The boundary layer is called laminar boundary layer if the Reynold number of the flow defined as

$$R_e = \frac{U \times x}{\nu} \text{ is less than } 5 \times 10^5$$

where U = Free-stream velocity of flow, x = Distance from leading edge,
and ν = Kinematic viscosity of fluid.

3. If the Reynold number is more than 5×10^5, the boundary layer is called turbulent boundary layer.

4. The distance from the surface of the solid body in the direction perpendicular to flow, where the velocity of fluid is approximately equal to 0.99 times the free-stream velocity is called boundary layer thickness and is denoted by δ. For different zones, δ is represented as
   $\delta_{lam.}$ = Thickness of laminar boundary layer
   $\delta_{tur.}$ = Thickness of turbulent boundary layer
   δ' = Thickness of laminar sub-layer.
5. Displacement thickness (δ^*) is given by $\delta^* = \int_0^\delta \left(1 - \frac{u}{U}\right) dy$.
6. Momentum thickness (θ) is given by $\theta = \int_0^\delta \frac{u}{U}\left[1 - \frac{u}{U}\right] dy$.
7. Energy thickness (δ^{**}) is given by $\delta^{**} = \int_0^\delta \frac{u}{U}\left[1 - \frac{u^2}{U^2}\right] dy$.
8. Von Karman momentum integral equation is given as $\frac{\tau_0}{\rho U^2} = \frac{\partial \theta}{\partial x}$

   where $\theta = \int_0^\delta \frac{u}{U}\left[1 - \frac{u}{U}\right] dy$, τ_0 = shear stress at surface.

   This equation is applicable to laminar, transition and turbulent boundary layer flows.
9. Thickness of laminar boundary layer and co-efficient of drag from Blasius's solution is given as

$$\delta = \frac{4.91\, x}{\sqrt{R_{e_x}}}$$

   where R_{e_x} = Reynold number, $C_D = \frac{1.328}{\sqrt{R_{e_L}}}$
10. Velocity profile for turbulent boundary layer is $\frac{u}{U} = \left(\frac{y}{\delta}\right)^{1/7}$

    This equation is not valid very near the boundary, where laminar sub-layer exists.
11. The shear stress at the boundary for turbulent boundary layer over a flat plate is given as

$$\tau_0 = 0.0225\, \rho U^2 \left(\frac{\mu}{\rho U \delta}\right)^{1/4}$$

12. Total drag on a flat plate due to laminar and turbulent boundary layer flows = Drag due to laminar boundary layer upto distance x + Drag due to turbulent boundary layer for length L – Drag due to turbulent boundary layer for length x.

    $\left[\text{where } x \text{ is given by } 5 \times 10^5 = \frac{Ux}{\nu}\right]$
13. If the pressure gradient is positive, the boundary layer separates from the surface and back flow and eddies formation take place due to which a great loss of energy occur.
14. The conditions for separation, attached flow and detached flow are :

    (*i*) $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ = 0 ..... condition for separation (*ii*) $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ = positive ..... condition for attached flow

    (*iii*) $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ = negative ..... condition for detached flow.

EXERCISE

(A) THEORETICAL PROBLEMS

1. What do you understand by the terms boundary layer, and boundary layer theory ?
2. Define : laminar boundary layer, turbulent boundary layer, laminar sub-layer and boundary layer thickness.
3. Define displacement thickness. Derive an expression for the displacement thickness.
4. Prove that the momentum thickness and energy thickness for boundary layer flows are given by

$$\theta = \int_0^{\delta} \frac{u}{U}\left[1 - \frac{u}{U}\right] dy \quad \text{and} \quad \delta^{**} = \int_0^{\delta} \frac{u}{U}\left[1 - \frac{u^2}{U}\right] dy.$$

5. Obtain an expression for the boundary shear stress in terms of momentum thickness.
6. Obtain Von Karman momentum integral equation.
7. What are the boundary conditions that must be satisfied by a given velocity profile in laminar boundary layer flows ?
8. How will you find the drag on a flat plate due to laminar and turbulent boundary layers ?
9. What do you mean by separation of boundary layer ? What is the effect of pressure gradient on boundary layer separation ?
10. How will you determine whether a boundary layer flow is attached flow, detached flow or on the verge of separation ?
11. What are the different methods of preventing the separation of boundary layers ?
12. What is meant by boundary layer ? Why does it increase with distance from the upstream edge ?
13. Define the terms : boundary layer, boundary layer thickness, drag, lift and momentum thickness.
14. What do you mean by boundary layer separation ? What is the effect of pressure gradient on boundary layer separation ? *(R.G.P.V., Bhopal S, 2001)*

(B) NUMERICAL PROBLEMS

1. (*a*) Find the ratios of displacement thickness to momentum thickness and momentum thickness to energy thickness for the velocity distribution in the boundary layer given by

$$\frac{u}{U} = 2\,(y/\delta) - (y/\delta)^2$$

where u = Velocity in boundary layer at a distance y
U = Free-stream velocity
δ = Boundary layer thickness **[Ans.** 2.5, 7/11]

(*b*) Find the displacement thickness, the momentum thickness and energy thickness for the velocity distribution in the boundary layer given by

$$\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2.$$

(Delhi University, December, 2002)

2. For the velocity profile in laminar boundary layer given as $\frac{u}{U} = \frac{3}{2}(y/\delta) - \frac{1}{2}(y/\delta)^3$, find the thickness of the boundary layer and shear stress 1.8 m from the leading edge of a plate. The plate is 2.5 m long and 1.5 m wide and is placed in water which is moving with a velocity of 15 cm per second. Find the drag on one side of the plate if the viscosity of water = 0.01 poise. [**Ans.** 1.6 cm, 0.0014 N/m^2, 0.0889 N]
3. Air is flowing over a smooth plate with a velocity of 8 m/s. The length of the plate is 1.5 m and width 1 m. If the laminar boundary exists upto a value of Reynold number = 5×10^5, find the maximum distance from the leading edge upto which laminar boundary layer exists. Find the maximum thickness of laminar boundary layer if the velocity profile is given by

$$\frac{u}{U} = (y/\delta) - (y/\delta)^2.$$ Take ν for air = 0.15 stokes. [**Ans.** 0.9375 m, 7.26 mm]

4. If in Problem 3, the velocity profile over the plate is given as $\frac{u}{U} = \sin\left(\frac{\pi}{2} \times \frac{y}{\delta}\right)$

and density of air as 1.24 kg/m^3, find : (*i*) maximum thickness of the laminar boundary layer, (*ii*) shear stress at 20 cm from the leading edge and (*iii*) drag force on one side of the plate assuming the laminar boundary layer over the entire length of the plate. [**Ans.** (*i*) 0.635 cm, (*ii*) 0.099 N/m^2, (*iii*) 0.0871 N]
5. A thin plate is moving in still atmospheric air at a velocity of 4 m/s. The length of the plate is 0.5 m and width 0.4 m. Calculate the (*i*) thickness of the boundary layer at the end of the plate and (*ii*) drag force on one side of the plate. Take density of air as 1.25 kg/m^3 and kinematic viscosity 0.15 stokes.
[**Ans.** (*i*) 0.672 cm (*ii*) 0.00728 N]
6. Find the frictional drag on one side of the plate 200 mm wide and 500 mm long placed longitudinally in a steam of crude oil (specific gravity = 0.925, kinematic viscosity = 0.9 stoke) flowing with undisturbed velocity of 5 m/s. Also find the thickness of boundary layer and the shear stress at the trailing edge of the plate. [**Ans.** 9.34 N, 14.75 mm]
7. A smooth flat plate of length 5 m and width 2 m is moving with a velocity of 4 m/s in stationary air of density as 1.25 kg/m^3 and kinematic viscosity 1.5×10^{-5} m^2/s. Determine thickness of the boundary layer at the trailing edge of the smooth plate. Find the total drag on one side of the plate assuming that the boundary layer is turbulent from the very beginning. [**Ans.** 110 mm, 0.43 N]
8. Water is flowing over a thin smooth plate of length 4.5 m and width 2.5 m at a velocity of 0.9 m/s. If the boundary layer flow changes from laminar to turbulent at a Reynold number 5×10^5, find (*i*) the distance from leading edge upto, which boundary layer is laminar, (*ii*) thickness of the boundary layer at the transition point, and (*iii*) the drag force on-one side of the plate. Take viscosity of water as 0.01 poise. [**Ans.** (*i*) 555 mm (*ii*) 3.85 mm (*iii*) 13.75 N]
9. For the velocity profile given below, state whether the boundary layer has separated or on the verge of separation or will remain attached with the surface :

(*i*) $\frac{u}{U} = 2\,(y/\delta) - (y/\delta)^2$ (*ii*) $\frac{u}{U} = -2\,(y/\delta) + \frac{1}{2}\,(y/\delta)^3$ and

(*iii*) $\frac{u}{U} = \frac{3}{2}\,(y/\delta)^2 + \frac{1}{2}(y/\delta)^3$.

[**Ans**. (*i*) Remain attached (*ii*) has separated (*iii*) on the verge of separation]
10. Oil with a free-stream velocity of 1.5 m/s flow over a thin plate 1.4 m wide and 2.2 m long. Calculate the boundary layer thickness and the shear stress at the trailing end point and determine the total surface resistance of the plate. Take specific gravity of oil as 0.80 and kinematic viscosity as 0.1 stoke.
[**Ans.** 1.88 cm, 1.04 N/cm^2, 12.8 N]

11. (*a*) For the velocity profile $\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^2$. Calculate the co-efficient of drag in terms of Reynolds number.

(*b*) A thin smooth plate of 0.3 m width and 1.0 m length moves at 4 m/s viscosity in still atmospheric air of density 1.20 kg/m^3 and kinematic viscosity of 1.49×10^{-5} m^2/s. Calculate the drag force on the plate.

[**Ans.** 0.00716 N]

12. Find the displacement thickness, the momentum thickness and energy thickness for the velocity distribution in the boundary layer given by,

$$\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$$ where δ = boundary layer thickness. $\left[\textbf{Ans.} \frac{\delta}{3}; \frac{2}{15}\delta; \frac{22}{105}\delta\right]$

14 CHAPTER

FORCES ON SUB-MERGED BODIES

▶ 14.1 INTRODUCTION

When a fluid is flowing over a stationary body, a force is exerted by the fluid on the body. Similarly, when a body is moving in a stationary fluid, a force is exerted by the fluid on the body. Also when the body and fluid both are moving at different velocities, a force is exerted by the fluid on the body. Some of the examples of the fluids flowing over stationary bodies or bodies moving in a stationary fluid are :

1. Flow of air over buildings,
2. Flow of water over bridges,
3. Submarines, ships, airplanes and automobiles moving through water or air.

▶ 14.2 FORCE EXERTED BY A FLOWING FLUID ON A STATIONARY BODY

Consider a body held stationary in a real fluid, which is flowing at a uniform velocity U as shown in Fig. 14.1.

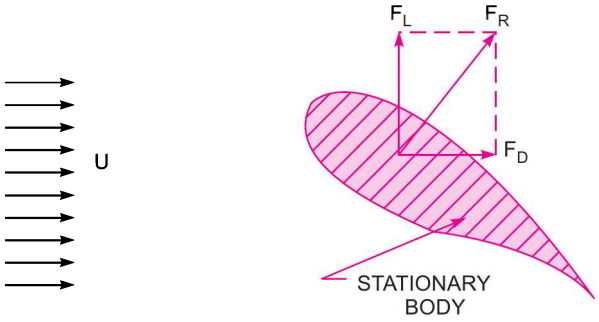


Fig. 14.1 *Force on a stationary body.*

The fluid will exert a force on the stationary body. The total force (F_R) exerted by the fluid on the body is perpendicular to the surface of the body. Thus the total force is inclined to the direction of motion. The total force can be resolved in two components, one in the direction of motion and other perpendicular to the direction of motion.

14.2.1 Drag. The component of the total force (F_R) in the direction of motion is called 'drag'. This component is denoted by F_D. Thus drag is the force exerted by the fluid in the direction of motion.

14.2.2 Lift. The component of the total force (F_R) in the direction perpendicular to the direction of motion is known as 'lift'. This is denoted by F_L. Thus lift is the force exerted by the fluid in the direction perpendicular to the direction of motion. Lift force occurs only when the axis of the body is inclined to the direction of fluid flow. If the axis of the body is parallel to the direction of fluid flow, lift force is zero. In that case only drag force acts.

If the fluid is assumed ideal and the body is symmetrical such as a sphere or cylinder, both the drag and lift will be zero.

▶ 14.3 EXPRESSION FOR DRAG AND LIFT

Consider an arbitrary shaped solid body placed in a real fluid, which is flowing with a uniform velocity U in a horizontal direction as shown in Fig. 14.2. Consider a small elemental area dA on the surface of the body. The forces acting on the surface area dA are :

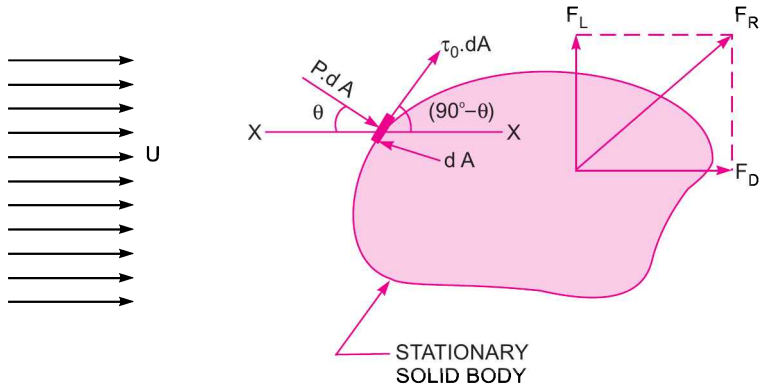


Fig. 14.2 *Drag and lift.*

1. Pressure force equal to $p \times dA$, acting perpendicular to the surface and
2. Shear force equal to $\tau_0 \times dA$, acting along the tangential direction to the surface.

Let θ = Angle made by pressure force with horizontal direction.

(*a*) **Drag Force (F_D).** The drag force on elemental area

= Force due to pressure in the direction of fluid motion
+ Force due to shear stress in the direction of fluid motion.

$= pdA \cos\theta + \tau_0 dA \cos(90° - \theta) = pdA \cos\theta + \tau_0 dA \sin\theta$

$\therefore$ Total drag, F_D = Summation of $pdA \cos\theta$ + Summation of $\tau_0 dA \sin\theta$

$= \int p \cos\theta dA + \int \tau_0 \sin\theta dA.$ ...(14.1)

The term $\int p \cos\theta dA$ is called the pressure drag or form drag while the term $\int \tau_0 \sin\theta dA$ is called the friction drag or skin drag or shear drag.

(*b*) **Lift Force (F_L).** The lift force on elemental area

= Force due to pressure in the direction perpendicular to the direction of motion
+ Force due to shear stress in the direction perpendicular to the direction of motion.

$= - pdA \sin\theta + \tau_0 dA \sin(90° - \theta) = - pdA \sin\theta + \tau_0 dA \cos\theta$

Negative sign is taken with pressure force as it is acting in the downward direction while shear force is acting vertically up.

∴ Total lift, $F_L = \int \tau_0 \, dA \cos\theta - \int pdA \sin\theta$

$= \int \tau_0 \cos\theta dA - \int p \sin\theta dA$ ...(14.2)

The drag and lift for a body moving in a fluid of density ρ, at a uniform velocity U are calculated mathematically, as

$$F_D = C_D A \frac{\rho U^2}{2} \quad ...(14.3)$$

$$F_L = C_L A \frac{\rho U^2}{2} \quad ...(14.4)$$

where C_D = Co-efficient of drag,

C_L = Co-efficient of lift,

A = Area of the body which is the projected area of the body perpendicular to the direction of flow

Or

= Largest projected area of the immersed body.

Then resultant force on the body, $F_R = \sqrt{F_D^2 + F_L^2}$ ...(14.5)

The equations (14.3) and (14.4) which give the mathematical expression for drag and lift are derived by the method of dimensional analysis.

14.3.1 Dimensional Analysis of Drag and Lift. In the chapter of Dimensional and Model Analysis it is shown in problem 12.6 that the force exerted by a fluid on a supersonic plane is given by :

$$F = \rho L^2 U^2 \phi \left[\frac{\mu}{\rho UL} \cdot \frac{K}{\rho U^2} \right] \quad ...(i)$$

Also in problem 12.7, it is shown that the force exerted by a fluid on a partially sub-merged body is given by :

$$F = \rho L^2 U^2 \phi \left[\frac{\mu}{\rho UL} \cdot \frac{Lg}{U^2} \right] \quad ...(ii)$$

Thus the general expression for the force exerted by a fluid (air or water) on a body (completely sub-merged or partially sub-merged) is given as

$$F = \rho L^2 U^2 \phi \left[\frac{\mu}{\rho UL} \cdot \frac{Lg}{U^2} \cdot \frac{K}{\rho U^2} \right] \quad ...(14.6)$$

where L = Length of body, U = Velocity of body,

μ = Viscosity of fluid, ρ = Density of fluid

F = Force exerted, k = Bulk modulus of fluid,

g = Acceleration due to gravity.

If the body is completely sub-merged in the fluid, the force exerted by the fluid on the body due to gravitational effect is negligible. Hence the non-dimensional term containing 'g' in equation (14.6), *i.e.*, $\frac{Lg}{U^2}$ is neglected. If the velocity of the body is comparable with velocity of sound, the effect due to

compressibility is to be considered. But if the ratio of the velocity of the body to the velocity of the sound is less than 0.3, the force exerted by the fluid on the body due to compressibility is negligible. Hence the non-dimensional term in equation (14.6) containing K can be neglected. Then the force exerted by fluid on the body is given as

$$F = \rho L^2 U^2 \phi \left[\frac{\mu}{\rho UL}\right] = \rho L^2 U^2 \phi \left[\frac{\rho UL}{\mu}\right]$$

where $\dfrac{\rho UL}{\mu}$ = Reynolds number = R_e

$$F = \rho L^2 U^2 \phi [R_e]. \qquad ...(14.7)$$

Now F is the total force exerted by the fluid on the body. The total force is having two components, one in the direction of motion called drag force and other component in the direction perpendicular to the direction of motion, called lift force.

The two components of F are expressed as

$$F_D = \frac{\rho L^2 U^2}{2} \times C_D$$

where C_D is a function of R_e and is called co-efficient of drag

$$= C_D A \frac{\rho U^2}{2} \qquad \{\because \quad L^2 = \text{Area} = A\}$$

And $$F_L = \frac{\rho L^2 U^2}{2} \times C_L$$

where C_L is a function of R_e and is called co-efficient of lift

$$= C_L . A \frac{\rho U^2}{2}.$$

Problem 14.1 *A flat plate 1.5 m × 1.5 m moves at 50 km/hour in stationary air of density 1.15 kg/m³. If the co-efficients of drag and lift are 0.15 and 0.75 respectively, determine :*

(i) The lift force, (ii) The drag force,

(iii) The resultant force, and

(iv) The power required to keep the plate in motion.

Solution. Given :

Area of the plate, $A = 1.5 \times 1.5 = 2.25 \text{ m}^2$

Velocity of the plate, $U = 50 \text{ km/hr} = \dfrac{50 \times 1000}{60 \times 60} \text{ m/s} = 13.89 \text{ m/s}$

Density of air $\rho = 1.15 \text{ kg/m}^3$

Co-efficient of drag, $C_D = 0.15$

Co-efficient of lift, $C_L = 0.75$

(*i*) **Lift Force (F_L).** Using equation (14.4),

$$F_L = C_L A \times \frac{\rho U^2}{2} = 0.75 \times 2.25 \times \frac{1.15 \times 13.89^2}{2} \text{ N} = \mathbf{187.20 \text{ N. Ans.}}$$

(*ii*) **Drag Force (F_D).** Using equation (14.3),

$$F_D = C_D \times A \times \frac{\rho U^2}{2} = 0.15 \times 2.25 \times \frac{1.15 \times 13.89^2}{2} \text{ N} = \textbf{37.44. N. Ans.}$$

(*iii*) **Resultant Force (F_R).** Using equation (14.5),

$$F_R = \sqrt{F_D^2 + F_L^2} = \sqrt{37.44^2 + 187.20^2} \text{ N}$$

$$= \sqrt{1400 + 35025} = \textbf{190.85 N. Ans.}$$

(*iv*) **Power Required to keep the Plate in Motion**

$$P = \frac{\text{Force in the direction of motion} \times \text{Velocity}}{1000} \text{ kW}$$

$$= \frac{F_D \times U}{1000} = \frac{37.425 \times 13.89}{1000} \text{ kW} = \textbf{0.519 kW. Ans.}$$

Problem 14.2 *Experiments were conducted in a wind tunnel with a wind speed of 50 km/hour on a flat plate of size 2 m long and 1 m wide. The density of air is 1.15 kg/m^3. The co-efficients of lift and drag are 0.75 and 0.15 respectively. Determine :*

(i) the lift force, *(ii) the drag force,*
(iii) the resultant force, *(iv) direction of resultant force and*
(v) power exerted by air on the plate.

Solution. Given :

Area of plate, $A = 2 \times 1 = 2 \text{ m}^2$

Velocity of air, $U = 50 \text{ km/hr} = \frac{50 \times 1000}{60 \times 60} \text{ m/s} = 13.89 \text{ m/s}$

Density of air, $\rho = 1.15 \text{ kg/m}^3$

Value of $C_D = 0.15$ and $C_L = 0.75$

(*i*) *Lift force (F_L)*

Using equation (14.4), $F_L = C_L \times A \times \rho \times U^2/2$

$= 0.75 \times 2 \times 1.15 \times 13.89^2/2 = \textbf{166.404 N. Ans.}$

(*ii*) *Drag force (F_D)*

Using equation (14.3), $F_D = C_D \times A \times \rho \times U^2/2$

$= 0.15 \times 2 \times 1.15 \times 13.89^2/2 = \textbf{33.28 N. Ans.}$

(*iii*) *Resultant force (F_R)*

Using equation (14.5), $F_R = \sqrt{F_D^2 + F_L^2} = \sqrt{33.28^2 + 166.404^2} = \textbf{169.67 N. Ans.}$

(*iv*) *The direction of resultant force* (θ)

The direction of resultant force is given by,

$$\tan \theta = \frac{F_L}{F_D} = \frac{166.38}{33.275} = 5$$

$\therefore$ $\theta = \tan^{-1} 5 = \textbf{78.69°. Ans.}$

(*v*) *Power exerted by air on the plate*

Power = Force in the direction of motion × Velocity

$= F_D \times U$ N m/s $= 33.280 \times 13.89$ W ($\because$ Watt = N m/s)

$= \textbf{462.26 W. Ans.}$

Problem 14.3 *Find the difference in drag force exerted on a flat plate of size 2 m × 2 m when the plate is moving at a speed of 4 m/s normal to its plane in : (i) water, (ii) air of density 1.24 kg/m³. Co-efficient of drag is given as 1.15.*

Solution. Given :

Area of plate, $A = 2 \times 2 = 4 \text{ m}^2$

Velocity of plate, $U = 4$ m/s

Co-efficient of drag, $C_D = 1.15$

(*i*) Drag force when the plate is moving in water.

Using equation (14.3), $F_D = C_D \times A \times \frac{\rho U^2}{2}$, where ρ for water = 1000

$$= 1.15 \times 4 \times 1000 \times \frac{4^2}{2} \text{ N} = 36800 \text{ N.} \quad ...(i)$$

(*ii*) Drag force when the plate is moving in air,

$$F_D = C_D \times A \times \frac{\rho U^2}{2}, \quad \text{where } \rho \text{ for air} = 1.24$$

$$\therefore \quad F_D = 1.15 \times 4.0 \times 1.24 \times \frac{4.0^2}{2.0} \text{ N} = 45.6 \text{ N} \quad ...(ii)$$

$\therefore$ Difference in drag force = (*i*) – (*ii*)

= 36800 – 45.6 = **36754. 4 N. Ans.**

Problem 14.4 *A truck having a projected area of 6.5 square metres travelling at 70 km/hour has a total resistance of 2000 N. Of this 20 per cent is due to rolling friction and 10 per cent is due to surface friction. The rest is due to form drag. Calculate the co-efficient of form drag. Take density of air = 1.25 kg/m³.*

Solution. Given :

Area of truck, $A = 6.5 \text{ m}^2$

Speed of truck, $U = 70 \text{ km/hr} = \frac{70 \times 100}{60 \times 60} = 19.44$ m/s

Total resistance, $F_T = 2000$ N

Rolling friction resistance, $F_C = 20\%$ of total resistance $= \frac{20}{100} \times 2000 = 400$ N

Surface friction resistance, $F_S = 10\%$ of total resistance $= \frac{10}{100} \times 2000 = 200$ N

$\therefore$ Form drag, $F_D = 2000 - F_C - F_S = 2000 - 400 - 200 = 1400$ N

Using equation (14.3), $F_D = C_D \times A \times \frac{\rho U^2}{2}$

where if F_D = Form drag then C_D = Co-efficient of form drag

$$\therefore \quad 1400 = C_D \times 6.5 \times 1.25 \times \frac{19.44^2}{2} \quad (\rho = \text{Density of air} = 1.25 \text{ kg/m}^3)$$

$$\therefore \quad C_D = \frac{1400 \times 2}{6.5 \times 1.25 \times 19.44 \times 19.44} = \mathbf{0.912. \text{ Ans.}}$$

Problem 14.5 *A circular disc 3 m in diameter is held normal to a 26.4 m/s wind of density 0.0012 gm/cc. What force is required to hold it at rest ? Assume co-efficient of drag of disc = 1.1.*

Solution. Given :

Diameter of disc $= 3$ m

$\therefore$ Area, $A = \frac{\pi}{4} \times (3)^2 = 7.0685 \text{ m}^2$

Velocity of wind, $U = 26.4$ m/s

Density of wind, $\rho = 0.0012 \text{ gm/cm}^3 = \frac{.0012}{1000} \text{ kg/cm}^3$

$$= \frac{0.0012}{1000} \times 10^6 \frac{\text{kg}}{\text{m}^3} = 1.2 \text{ kg/m}^3$$

Co-efficient of drag, $C_D = 1.1$

The force required to hold the disc at rest is equal to the drag exerted by wind on the disc.

Drag (F_D) is given by equation (14.3) as

$$F_D = C_D \times A \times \frac{\rho U^2}{2} = \frac{1.1 \times 7.0685 \times 1.2 \times 26.4^2}{2.0} = \mathbf{3251.4 \text{ N. Ans.}}$$

Problem 14.6 *A man weighing 90 kgf descends to the ground from an aeroplane with the help of a parachute against the resistance of air. The velocity with which the parachute, which is hemispherical in shape, comes down is 20 m/s. Find the diameter of the parachute. Assume $C_D = 0.5$ and density of air = 1.25 kg/m^3.*

Solution. Given :

Weight of man, $W = 90 \text{ kgf} = 90 \times 9.81 \text{ N} = 882.9 \text{ N}$ ($\because$ 1 kgf = 9.81 N)

Velocity of parachute, $U = 20$ m/s

Co-efficient of drag, $C_D = 0.5$

Density of air, $\rho = 1.25 \text{ kg/m}^3$

Let the diameter of parachute $= D$

$\therefore$ Area, $A = \frac{\pi}{4} D^2 \text{ m}^2$.

When the parachute with the man comes down with a uniform velocity, $U = 20$ m/s, the drag resistance will be equal to the weight of man, neglecting the weight of parachute. And projected area of the hemispherical parachute will be equal to $\frac{\pi}{4} D^2$.

$\therefore$ Drag, $F_D = 90 \text{ kgf} = 90 \times 9.81 = 882.9 \text{ N}$

Using equation (14.3), $F_D = C_D \times A \times \frac{\rho U^2}{2}$

$\therefore$ $$882.9 = 0.5 \times \frac{\pi}{4} D^2 \times \frac{1.25 \times 20^2}{2}$$

$\therefore$ $$D^2 = \frac{882.9 \times 4 \times 2.0}{0.5 \times \pi \times 1.25 \times 20 \times 20} = 8.9946 \text{ m}^2$$

or $$D = \sqrt{8.9946} = \mathbf{2.999 \text{ m. Ans.}}$$

Fig. 14.3

Problem 14.7 *A man weighing 981 N descends to the ground from an aeroplane with the help of a parachute against the resistance of air. The shape of the parachute is hemispherical of 2 m diameter. Find the velocity of the parachute with which it comes down. Assume C_d = 0.5 and ρ for air = 0.00125 gm/cc and ν = 0.015 stoke.*

Solution. Given :

Weight of the man, $W = 981$ N

$\therefore$ Drag force, $F_D = W = 981$ N

Diameter of the parachute, $D = 2$ m

$\therefore$ Projected area, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} \times 2^2 = \pi \text{ m}^2$

Co-efficient of drag, $C_d = 0.5$

Density for air, $\rho = 0.00125 \text{ gm/cm}^3 = \frac{.00125}{1000} \text{ kg/cm}^3$

$$= \frac{.00125}{1000} \times 10^6 \frac{\text{kg}}{\text{m}^3} = 1.25 \text{ kg/m}^3$$

Let the velocity of parachute $= U$

Using equation (14.3), $F_D = C_D \times A \times \frac{\rho U^2}{2}$ or $981 = 0.5 \times \pi \times \frac{1.25 \times U^2}{2.0}$

$\therefore$ $U = \sqrt{\frac{981 \times 2.0}{0.5 \times \pi \times 1.25}} =$ **31.61 m/s. Ans.**

Problem 14.8 *A man descends to the ground from an aeroplane with the help of a parachute which is hemispherical having a diameter of 4 m against the resistance of air with a uniform velocity of 25 m/s. Find the weight of the man if the weight of parachute is 9.81 N. Take C_D = 0.6 and density of air = 1.25 kg/m^3.*

Solution. Given :

Diameter of parachute, $D = 4$ m

$\therefore$ Projected area, $A = \frac{\pi}{4} \times D^2 = \frac{\pi}{4} \times 4^2 = 4\pi \text{ m}^2$

Velocity of parachute, $U = 25$ m/s

Weight of parachute, $W_1 = 9.81$ N

Co-efficient of drag, $C_D = 0.6$

Density of air, $\rho = 1.25 \text{ kg/m}^3$

Let the weight of man $= W_2$

Then weight of man + Weight of parachute = $W_2 + W_1 = (W_2 + 9.81)$

Hence drag force will be equal to the weight of man plus weight of parachute.

$\therefore$ Drag force, $F_D = (W_2 + 9.81)$

Using equation (14.3), we have $F_D = C_D \times A \times \frac{\rho U^2}{2}$

or $(W_2 + 9.81) = 0.6 \times 4\pi \times \frac{1.25 \times 25^2}{2.0} = 2945.24$ N

$\therefore$ $W_2 = 2945.24 - 9.81 =$ **2935.43 N. Ans.**

Problem 14.9 *Calculate the diameter of a parachute to be used for dropping an object of mass 100 kg so that the maximum terminal velocity of dropping is 5 m/s. The drag co-efficient for the parachute, which may be treated as hemispherical is 1.3. The density of air is 1.216 kg/m^3.*

Solution. Given :

Mass of object, $M = 100$ kg

Weight of object, $W = 100 \times 9.81 = 981$ N

$\therefore$ Drag force, $F_D = 981$ N

Velocity of object, $U = 5$ m/s

Drag co-efficient, $C_D = 1.3$

Density of air, $\rho = 1.216$ kg/m^3

Let the diameter of parachute $= D$ m

$\therefore$ Projected area, $A = \dfrac{\pi}{4} D^2$ m^2

Using equation (14.3), $F_D = C_D \times A \times \dfrac{\rho U^2}{2}$

or $$981 = 1.3 \times \frac{\pi}{4} D^2 \times \frac{1.216 \times 5^2}{2}$$

or $$D^2 = \frac{981 \times 4 \times 2}{1.3 \times \pi \times 1.216 \times 5 \times 5} = 63.21$$

$\therefore$ $D = \sqrt{63.21} =$ **7.95 m. Ans.**

Problem 14.10 *A kite 0.8 m $\times$ 0.8 m weighing 0.4 kgf (3.924 N) assumes an angle of 12° to the horizontal. The string attached to the kite makes an angle of 45° to the horizontal. The pull on the string is 2.5 kgf (24.525 N) when the wind is flowing at a speed of 30 km/hour. Find the corresponding co-efficient of drag and lift. Density of air is given as 1.25 kg/m^3.*

Solution. Given :

Projected area of kite, $A = 0.8 \times 0.8 = 0.64$ m^2

Weight of kite, $W = 0.4$ kgf $= 0.4 \times 9.81 = 3.924$ N

Angle made by kite with horizontal, $\theta_1 = 12°$

Angle made by string with horizontal, $\theta_2 = 45°$

Pull on the string, $P = 2.5$ kgf $= 2.5 \times 9.81 = 24.525$ N

Speed of wind, $U = 30$ km/hr $= \dfrac{30 \times 1000}{60 \times 60}$ m/s $= 8.333$ m/s

Density of air, $\rho = 1.25$ kg/m^3

Drag force, F_D = Force exerted by wind in the direction of motion (*i.e.*, in the *X-X* direction)
= Component of pull, P along *X-X*
= $P \cos 45° = 24.525 \cos 45° = 17.34$ N

And lift force, F_L = Force exerted by wind on the kite perpendicular to the direction of motion (*i.e.*, along *Y-Y* direction)
= Component of P in vertically downward direction + Weight of kite (W)

$= P \sin 45° + W = 24.525 \sin 45° + 3.924$ N
$= 17.34 + 3.924 = 21.264$ N.

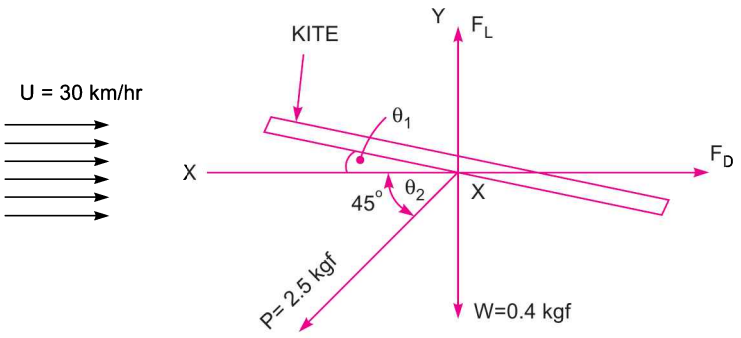


Fig. 14.4

(*i*) **Drag Co-efficient (C_D).** Using equation (14.3), we have

$$F_D = C_D \times A \times \frac{\rho U^2}{2}$$

or
$$C_D = \frac{2 \times F_D}{A \rho U^2} = \frac{2 \times 17.34}{0.64 \times 1.25 \times 8.333^2} = \mathbf{0.624.\ Ans.}$$

(*ii*) **Lift Co-efficient (C_L).** Using equation (14.4), we have

$$F_L = C_L \times A \times \frac{\rho U^2}{2}$$

or
$$C_L = \frac{2 \times F_L}{A \times \rho \times U^2} = \frac{2 \times 21.264}{0.64 \times 1.25 \times 8.333^2} = \mathbf{0.765.\ Ans.}$$

Problem 14.11 *A kite weighing 0.8 kgf (7.848 N) has an effective area of 0.8 m^2. It is maintained in air at an angle of 10° to the horizontal. The string attached to the kite makes an angle of 45° to the horizontal and at this position the value of co-efficient of drag and lift are 0.6 and 0.8 respectively. Find the speed of the wind and the tension in the string. Take the density of air as 1.25 kg/m^3.*

Solution. Given :

Weight of kite, $W = 0.8$ kgf $= 0.8 \times 9.81 = 7.848$ N

Effective area, $A = 0.8$ m^2

Angle made by kite with horizontal = 10°

Angle made by string with horizontal = 45°

$C_D = 0.6$

$C_L = 0.8$

$\therefore$ Density of air, $\rho = 1.25$ kg/m^3

Let the speed of the wind $= U$ m/s

and tension in string $= T$

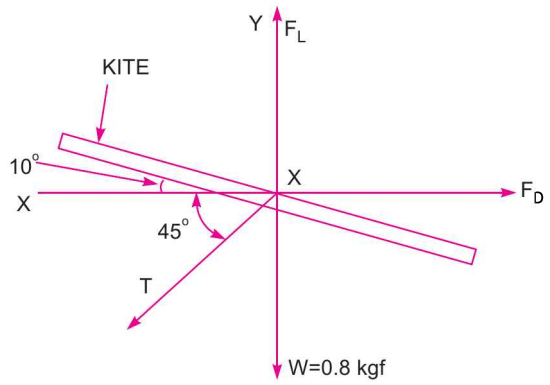


Fig. 14.5

The free body diagram for the kite is shown in Fig. 14.5.

Drag force, F_D = Component of T along X-X = $T \cos 45°$

But drag force F_D is also $= C_D \times A \times \dfrac{\rho U^2}{2} = \dfrac{0.6 \times 0.8 \times 1.25 \times U^2}{2} = 0.3\ U^2$

Equating the two values of F_D, $T \cos 45° = 0.3\ U^2$ ...(*i*)

Now lift force from Fig. 14.5, F_L = Component of T vertically downward + Weight of kite

$= T \sin 45° + 7.848$

Also lift force, $F_L = \dfrac{C_L \times A \times \rho U^2}{2} = \dfrac{0.8 \times 0.8 \times 1.25 \times U^2}{2.0} = 0.4\ U^2$

Equating the two values of F_L, $T \sin 45° + 7.848 = 0.4\ U^2$

or $T \sin 45° = .04076\ U^2 - 7.848$ ...(*ii*)

From equations (*i*) and (*ii*), as $T \cos 45° = T \sin 45°$,

$0.3\ U^2 = 0.4\ U^2 - 7.848$

or $7.848 = 0.4\ U^2 - 0.3\ U^2 = 0.1\ U^2$

$\therefore \quad U^2 = \dfrac{7.848}{0.1} = 78.48$ ...(*iii*)

$\therefore \quad U = \sqrt{78.48} = 8.86$ m/s $= \dfrac{8.86 \times 60 \times 60}{1000}$ km/hr = **31.89 km/hr. Ans.**

Substituting the value of $U^2 = 78.48$ given by equation (*iii*) into equation (*i*), we get

$T \cos 45° = 0.3 \times 78.48 = 23.544$

$\therefore \quad T = \dfrac{23.544}{\cos 45°} =$ **33.3 N. Ans.**

Problem 14.12 *The air is flowing over a cylinder of diameter 50 mm and infinite length with a velocity of 0.1 m/s. Find the total drag, shear drag and pressure drag on 1 m length of the cylinder if the total drag co-efficient is equal to 1.5 and shear drag co-efficient equal to 0.2. Take density of air = 1.25 kg/cm^3.*

Solution. Given :

Diameter of cylinder, $D = 50 \text{ mm} = 0.05 \text{ m}$

Length of cylinder, $L = 1.0 \text{ m}$

$\therefore$ Projected Area, $A = L \times D = 1 \times .05 = 0.05 \text{ m}^2$

Velocity of air, $U = 0.1 \text{ m/s}$

Total drag co-efficient, $C_{DT} = 1.5$

Shear drag co-efficient, $C_{DB} = 0.2$

Density of air, $\rho = 1.25 \text{ kg/m}^3$

Total drag is given by, $F_{DT} = C_{DT} \times A \times \dfrac{\rho U^2}{2}$

$$= 1.5 \times .05 \times \frac{1.25 \times (0.1)^2}{2} = \textbf{0.000468 N. Ans.}$$

Shear drag is given by, $F_{DS} = C_{DS} \times A \times \dfrac{\rho U^2}{2}$

$$= 0.2 \times .05 \times \frac{1.25 \times (0.1)^2}{2} = \textbf{0.0000625 N. Ans.}$$

From equation (14.1), Total drag, F_{DT} = Pressure drag + Shear drag

$\therefore$ Pressure drag, = Total drag – Shear drag

= 0.000468 – 0.0000625 = **0.0004055 N. Ans.**

Problem 14.13 *A body of length 2.0 m has a projected area 1.5 m^2 normal to the direction of its motion. The body is moving through water, which is having viscosity = 0.01 poise. Find the drag on the body if it has a drag co-efficient 0.5 for a Reynold number of 8×10^6.*

Solution. Given :

Length of body, $L = 2.0 \text{ m}$

Projected Area, $A = 1.5 \text{ m}^2$

Viscosity of water, $\mu = 0.01 \text{ poise} = \dfrac{0.01}{10} = 0.001 \dfrac{\text{Ns}}{\text{m}^2}$

Drag co-efficient, $C_d = 0.5$

Reynold number, $R_e = 8 \times 10^6$

Let the drag force on body $= F_D$

First find the velocity with which body is moving in water. It is calculated from the given Reynold number.

$$R_e = \frac{\rho UL}{\mu}, \text{ where } \rho \text{ for water} = 1000$$

or $$8 \times 10^6 = \frac{1000 \times U \times 2.0}{0.001} = 2 \times 10^6 \, U$$

$$\therefore \quad U = \frac{8 \times 10^6}{2 \times 10^6} = 4.0 \text{ m/s}$$

Using equation (14.3), $$F_D = C_d \times A \times \frac{\rho U^2}{2} = 0.5 \times 1.5 \times 1000 \times \frac{4.0^2}{2} = \textbf{6049 N. Ans.}$$

Problem 14.14 *A sub-marine which may be supposed to approximate a cylinder 4 m in diameter and 20 m long travels sub-merged at 1.3 m/s in sea-water. Find the drag exerted on it, if the drag co-efficient for Reynold number greater than 10^5 may be taken as 0.75. The density of sea-water is given as 1035 kg/m^3 and kinematic viscosity as .015 stokes.*

Solution. Given :

Dia. of cylinder, $D = 4$ m

Length of cylinder, $L = 20$ m

Velocity of cylinder, $U = 1.3$ m/s

Density of sea-water, $\rho = 1035$ kg/m^3

Kinematic viscosity $\nu = 0.015$ stokes $= .015$ cm^2/s $= .015 \times 10^{-4}$ m^2/s

Let the drag force $= F_D$

Reynold number, $$R_e = \frac{U \times D}{\nu} = \frac{1.3 \times 4.0}{.015 \times 10^{-4}} = 3.466 \times 10^6$$

Since $R_e > 10^5$, hence $C_D = 0.75$

Drag force is given by equation (14.3) as

$$F_D = C_D \times A \times \frac{\rho U^2}{2}$$

where A = projected area of cylinder $= L \times D = 20 \times 4.0 = 80.0$ m^2

$$\therefore \quad F_D = 0.75 \times 80 \times \frac{1035 \times 1.3^2}{2.0} \text{ kgf} = \textbf{52472.2 N. Ans.}$$

Problem 14.15 *A jet plane which weighs 29.43 kN and having a wing area of 20 m^2 flies at a velocity of 950 km/hour, when the engine delivers 7357.5 kW power. 65% of the power is used to overcome the drag resistance of the wing. Calculate the co-efficients of lift and drag for the wing. The density of the atmospheric air is 1.21 kg/m^3.*

Solution. Given :

Weight of plane, $W = 29.43$ kN $= 29.43 \times 1000$ N $= 29430$ N

Wing area, $A = 20$ m^2

Speed of plane, $$U = 950 \text{ km/hr} = \frac{950 \times 1000}{60 \times 60} = 263.88 \text{ m/s}$$

Engine power, $P = 7357.5$ kW

Power used to overcome drag resistance = 65% of 7357.5 $= \frac{65}{100} \times 7357.5 = 4782.375$ kW

∴ Density of air, $\rho = 1.21 \text{ kg/m}^3$

Let C_D = Co-efficient of drag and C_L = Co-efficient of lift.

Now power used in kW to overcome drag resistance = $\frac{F_D \times U}{1000}$ or $4782.375 = \frac{F_D \times 263.88}{1000}$

$$\therefore \quad F_D = \frac{4782.375 \times 1000}{263.88}$$

But from equation (14.3), we have $F_D = C_D \cdot A \cdot \frac{\rho U^2}{2}$

$$\therefore \quad \frac{4782.375 \times 1000}{263.88} = C_D \times 20 \times 1.21 \times \frac{263.88^2}{2}$$

$$\therefore \quad C_D = \frac{4782.375 \times 1000 \times 2}{20 \times 1.21 \times 263.88^3} = \mathbf{0.0215. \text{ Ans.}}$$

The lift force should be equal to weight of the plane

$$\therefore \quad F_L = W = 29430 \text{ N}$$

But $F_L = C_L \cdot A \cdot \frac{\rho U^2}{2}$ or $29430 = C_L \times 20 \times 1.21 \times \frac{263.88^2}{2}$

$$\therefore \quad C_L = \frac{29430 \times 2}{20 \times 1.21 \times 263.88^2} = \mathbf{0.0349. \text{ Ans.}}$$

14.3.2 Pressure Drag and Friction Drag. The total drag on a body is given by equation (14.1) as

Total drag, $F_D = \int p \cos\theta dA + \int \tau_0 \sin\theta dA$ ...(*i*)

where $\int p \cos\theta dA$ = Pressure drag or form drag, and

$\int \tau_0 \sin\theta dA$ = Friction drag or skin drag or shear drag

The relative contribution of the pressure drag and friction drag to the total drag depends on :

(*i*) Shape of the immersed body,

(*ii*) Position of the body immersed in the fluid, and

(*iii*) Fluid characteristics.

Consider the flow of a fluid over a flat plate when the plate is placed parallel to the direction of the flow as shown in Fig. 14.6. In this cos θ, which is the angle made by pressure with the direction motion, will be 90°. Thus the term $\int p \cos\theta dA$ will be zero and hence total drag will be equal to friction drag (or shear drag). If the plate is placed perpendicular to the flow as shown in Fig. 14.7, the angle θ, made by the pressure with the direction of motion will be zero. Hence the term $\int \tau_0 \sin\theta dA$ will become equal to zero and hence total drag will be due to the pressure difference between the upstream and downstream side of the plate. If the plate is held at an angle with the direction of flow, both the terms $\int p \cos\theta dA$ and $\int \tau_0 \sin\theta dA$ will exist and total drag will be equal to the sum of pressure drag and friction drag.

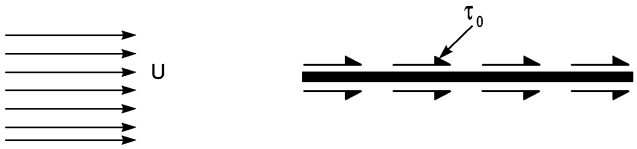


Fig. 14.6 *Flat plate parallel to flow.*

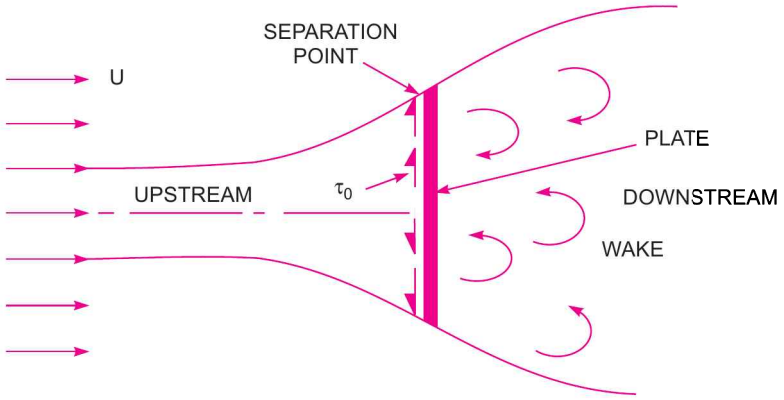


Fig. 14.7 *Flat plate perpendicular to flow.*

14.3.3 Stream-lined Body. A stream-lined body is defined as that body whose surface coincides with the stream-lines, when the body is placed in a flow. In that case the separation of flow will take place only at the trailing edge (or rearmost part of the body). Though the boundary layer will start at the leading edge, will become turbulent from laminar, yet it does not separate upto the rearmost part of the body in the case of stream-lined body. Thus behind a stream-lined body, wake formation zone will be very small and consequently the pressure drag will be very small. Then the total drag on the stream-lined body will be due to friction (shear) only. A body may be stream-lined :

1. at low velocities but may not be so at higher velocities.
2. when placed in a particular position in the flow but may not be so when placed in another position.

14.3.4 Bluff Body. A bluff body is defined as that body whose surface does not coincide with the streamlines, when placed in a flow. Then the flow is separated from the surface of the body much ahead of its trailing edge with the result of a very large wake formation zone. Then the drag due to pressure will be very large as compared to the drag due to friction on the body. Thus the bodies of such a shape in which the pressure drag is very large as compared to friction drag are called bluff bodies.

▶ 14.4 DRAG ON A SPHERE

Consider the flow of a real fluid past a sphere.

Let U = Velocity of the flow of fluid over sphere,
D = Diameter of sphere,
ρ = Mass density of fluid, and
μ = Viscosity of fluid.

If the Reynolds number of the flow is very small less than 0.2 $\left(i.e., R_e = \dfrac{UD\rho}{\mu} < 0.2 \right)$, the viscous forces are much more important than the inertial forces as in this case viscous forces are much more predominate than the inertial forces, which may be assumed negligible. G.G. Stokes, developed a mathematical equation for the total drag on a sphere immersed in a flowing fluid for which Reynolds number is upto 0.2, so that inertia forces may be assumed negligible. According to his solution, total drag is

$$F_D = 3\pi\mu DU. \qquad ...(14.8)$$

He further observed that out of the total drag given by equation (14.8), two-third is contributed by skin friction and the remaining one-third by pressure difference. Thus

Skin friction drag, $F_{D_f} = \frac{2}{3} F_D = \frac{2}{3} \times 3\pi\mu DU = 2\pi\mu DU$

and pressure drag, $F_{D_p} = \frac{1}{3} F_D = \frac{1}{3} \times 3\pi\mu DU = \pi\mu DU.$

(*i*) **Expression of C_d for Sphere when Reynolds Number is less than 0.2.** From equation (14.3), the total drag is given by

$$F_D = C_D \times A \times \frac{\rho U^2}{2}$$

For sphere, $F_D = 3\pi\mu DU$

A = Projected area of the sphere = $\frac{\pi}{4} D^2$

$\therefore$ $$3\pi\mu DU = C_D \times \frac{\pi}{4} D^2 \frac{\rho U^2}{2}$$

$$C_D = \frac{3\pi\mu DU}{\frac{\pi}{4} D^2 \times \frac{\rho U^2}{2}} = \frac{24\mu}{\rho UD} = \frac{24}{R_e} \qquad ...(14.9) \left(\because \frac{\mu}{\rho DU} = R_e \right)$$

Equation (14.9) is called **'Stoke's law'.**

(*ii*) **Value of C_D for Sphere when R_e is between 0.2 and 5.** With the increase of Reynolds number, the inertia forces increase and must be taken into account. When R_e lies between 0.2 and 5, Oseen, a Swedish physicist, improved Stoke's law as

$$C_D = \frac{24}{R_e}\left[1 + \frac{3}{16R_e}\right] \qquad ...(14.10)$$

Equation (14.10) is called Oseen formulae and is valid for R_e between 0.2 and 5.

(*iii*) **Value of C_D for R_e from 5.0 to 1000.** The drag co-efficient for the Reynolds number from 5 to 1000 is equal to 0.4.

(*iv*) **Value of C_D for R_e from 1000 to 100,000.** In this range, C_D is independent of the Reynolds number and its value is approximately equal to 0.5.

(*v*) **Value of C_D for R_e more than 10^5.** The value of C_D is approximately equal to 0.2 for the Reynolds number more than 10^5.

Problem 14.16 *Calculate the weight of a ball of diameter 80 mm which is just supported in a vertical air stream which is flowing at a velocity of 7 m/s. The density of air is given as 1.25 kg/m^3. The kinematic viscosity of air = 1.5 stokes.*

Solution. Given :

Dia. of ball, $D = 80$ mm $= 0.08$ m

Velocity of air, $U = 7$ m/s

Density of air, $\rho = 1.25$ kg/m^3

Kinematic viscosity, $\nu = 1.5$ stokes $= 1.5 \times 10^{-4}$ m^2/s

Reynold number, $R_e = \dfrac{U \times D}{\nu} = \dfrac{7 \times .08}{1.5 \times 10^{-4}} = 0.373 \times 10^4 = 3730.$

Thus the value of R_e lies between 1000 and 100,000 and hence $C_D = 0.5$.

When the ball is supported in a vertical air stream, the weight of ball is equal to the drag force as shown in Fig. 14.8.

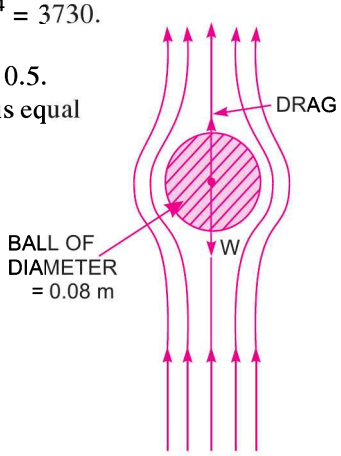


Fig. 14.8

But drag force, $F_D = C_D \times A \times \dfrac{\rho U^2}{2}$

where A = projected area of ball

$$= \frac{\pi}{4} D^2 = \frac{\pi}{4} (0.08)^2 = 0.005026 \text{ m}^2$$

$\therefore$ Drag force, $F_D = 0.5 \times 0.005026 \times \dfrac{1.25 \times 7^2}{2}$ N

$= 0.07696$ N

$\therefore$ Weight of ball $= F_D =$ **0.07696 N. Ans.**

▶ 14.5 TERMINAL VELOCITY OF A BODY

Terminal velocity is defined as the maximum constant velocity of a falling body (such as sphere or a composite body such as parachute together with man) with which the body will be travelling. When the body is allowed to fall from rest in the atmosphere, the velocity of the body increases due to acceleration of gravity. With the increase of the velocity, the drag force, opposing the motion of body also increases. A stage is reached when the upward drag force acting on the body will be equal to the weight of the body. Then the net external force acting on the body will be zero and the body will be travelling at constant speed. This constant speed is called terminal velocity of the falling body.

If the body drops in a fluid, at the instant it has acquired terminal velocity, the net force acting on the body will be zero. The forces acting on the body at this state will be :

1. Weight of body (W), acting downward,
2. Drag force (F_D), acting vertically upward, and
3. Buoyant force (F_B), acting vertically up.

The net force on the body should be zero, *i.e.*, $W = F_D + F_B$ ...(14.11)

Problem 14.17 *A metallic sphere of sp. gr. 7.0 falls in an oil of density 800 kg/m^3. The diameter of the sphere is 8 mm and it attains a terminal velocity of 40 mm/s. Find the viscosity of the oil in poise.*

Solution. Given :

Sp. gr. of metallic sphere $= 7.0$

$\therefore$ Density of metallic sphere, $\rho_s = 7 \times 1000 = 7000$ kg/m^3

Density of oil, $\rho_o = 800$ kg/m^3

Dia. of sphere, $D = 8$ mm $= 8 \times 10^{-3}$ m

Terminal velocity, $U = 40$ mm/s $= .04$ m/s

Let the viscosity of oil $= \mu$

Weight of sphere, $W = \rho_s \times g \times$ Volume of sphere

$$= 7000 \times 9.81 \times \frac{\pi}{6} D^3$$

$$= 7000 \times 9.81 \times \frac{\pi}{6} \times (8 \times 10^{-3})^3 = 18.4 \times 10^{-3} = 0.0184 \text{ N}$$

Buoyant force on sphere, F_B = Density of oil × g × Volume of sphere

$$= 800 \times 9.81 \times \frac{\pi}{6} (8 \times 10^{-3})^3 \text{ N} = 0.002103 \text{ N}.$$

Drag force, F_D on the sphere is given by equation (14.8) as

$$F_D = 3\pi\mu DU = 3\pi\mu \times 8 \times 10^{-3} \times .04 = .003015\ \mu$$

Using equation (14.11), $W = F_D + F_B$

or $0.0184 = .003015\ \mu + 0.002103$

or $.003015\ \mu = 0.0184 - 0.002103 = 0.016297$

or $$\mu = \frac{0.016297}{.003015} = 5.4 \frac{\text{Ns}}{\text{m}^2} = 5.4 \times 10 = \textbf{54.0 poise. Ans.}$$

The expression for drag given by equation (14.11) is valid only upto Reynolds number less than 0.2. Hence it is necessary to calculate Reynold number for the flow.

$\therefore$ Reynold number, $R_e = \dfrac{\rho UD}{\mu}$, where ρ for oil = 800 kg/m^3

$\therefore$ $$R_e = 800 \times \frac{.04 \times 8 \times 10^{-3}}{5.4} = 0.0474$$

Hence $R_e < 0.2$ and so the expression $F_D = 3\pi\mu DU$ is valid.

Problem 14.18 *A spherical steel ball of diameter 40 mm and of density 8500 kg/m^3 is dropped in large mass of water. The co-efficient of drag of the ball in water is given as 0.45. Find the terminal velocity of the ball in water. If the ball is dropped in air, find the increase in terminal velocity of ball. Take the density of air = 1.25 kg/m^3 and C_D = 0.1.*

Solution. Given :

Diameter of steel ball, D = 40 mm = 0.04 m

Density of ball, ρ_s = 8500 kg/m^3

C_D for ball in water = 0.45

Let the terminal velocity in water = U_1

The forces acting on the spherical ball are :

1. Weight, W = Density of ball × g × Volume of spherical ball

$$= \rho_s \times g \times \frac{\pi}{6} D^3 = 8500 \times 9.81 \times \frac{\pi}{6} (.04)^3 = \textbf{2.794 N.}$$

2. Buoyant force, F_B = Density of water × g × Volume of ball

$$= 1000 \times 9.81 \times \frac{\pi}{6} (0.04)^3 = 0.3286 \text{ N}.$$

3. Drag force, $$F_D = C_D \times A \times \frac{\rho U^2}{2}$$

where A = projected area = $\dfrac{\pi}{4} D^2 = \dfrac{\pi}{4} (.04)^2$, ρ = 1000 for water

$$F_D = 0.45 \times \frac{\pi}{4} \times (.04)^2 \times 1000 \times \frac{U_1^2}{2} = 0.2825\ U_1^2$$

Using equation (14.11), we get $W = F_D + F_B$

or $2.794 = 0.2825\ U_1^2 + 0.3286$

or $U_1^2 = \dfrac{2.794 - 0.3286}{0.2825} = 8.725$

$\therefore$ $U_1 = \sqrt{8.725} = \mathbf{2.953\ m/s.\ Ans.}$

When ball is dropped in air. Let the terminal velocity = U_2

Weight, $W = 2.794$

Buoyant force, F_B = Density of air $\times\ g \times$ Volume of ball

$$= 1.25 \times 9.81 \times \frac{\pi}{6} (.04)^3 = 0.000411 \text{ N}$$

Drag force, $F_D = C_D \times A \times \dfrac{\rho U^2}{2}$, where ρ for air = 1.25

$$F_D = 0.1 \times \frac{\pi}{4} (.04)^2 \times 1.25 \frac{U_2^2}{2} = 0.0000785\ U_2^2.$$

The buoyant force in air is 0.000411, while weight of the ball is 2.794 N. Hence buoyant force is negligible.

$\therefore$ For equilibrium of the ball in air, F_D = Weight of ball

or $0.0000785\ U_2^2 = 2.794$ or $U_2 = \sqrt{\dfrac{2.794}{0.0000785}} = 188.67$ m/s

$\therefore$ Increase in terminal velocity in air = $U_2 - U_1 = 188.67 - 2.9533 = \mathbf{185.717\ m/s.\ Ans.}$

Problem 14.19 *A metallic ball of diameter 2×10^{-3} m drops in a fluid of sp. gr. 0.95 and viscosity 15 poise. The density of the metallic ball is 12000 kg/m^3. Find :*

(i) The drag force exerted by fluid on metallic ball,

(ii) The pressure drag and skin friction drag,

(iii) The terminal velocity of ball in fluid.

Solution. Given :

Diameter of metallic ball, $D = 2 \times 10^{-3}$ m

Sp. gr. of fluid, $S_0 = 0.95$

$\therefore$ Density of fluid, $\rho_0 = 0.95 \times 1000 = 950$ kg/m^3

Viscosity of fluid, $\mu = 15$ poise $= \dfrac{15}{10} = 1.5 \dfrac{\text{Ns}}{\text{m}^2}$

Density of ball, $\rho_s = 12000$ kg/m^3

The forces acting on the ball are :

Weight of ball, W = Density of ball $\times\ g \times$ Volume of ball

$$= 12000 \times 9.81 \times \frac{\pi}{6} D^3$$

$$= 12000 \times 9.81 \times \frac{\pi}{6} \times (2 \times 10^{-3})^3 \text{ N} = 0.000493 \text{ N}$$

Buoyant force, F_B = Density of fluid $\times\ g \times$ Volume of ball

$$= 950 \times 9.81 \times \frac{\pi}{6} (2 \times 10^{-3})^3 \text{ N} = 0.000039 \text{ N}$$

When the metallic ball reaches the terminal velocity, equation (14.11) is applicable.

$\therefore$ $W = F_D + F_B$ or $F_D = W - F_B = 0.000493 - 0.000039$

$= \mathbf{0.000454\ N.\ Ans.}$

(*i*) Drag force, $F_D = 0.000454$ N

(*ii*) Pressure drag $= \frac{1}{3} F_D = \frac{1}{3} \times 0.000454 = \mathbf{0.0001513\ N.\ Ans.}$

Skin friction drag $= \frac{2}{3} \times F_D = \frac{2}{3} \times 0.000454 = \mathbf{0.0003028\ N.\ Ans.}$

(*iii*) Let the terminal velocity $= U$

Then drag force (F_D) is given by equation (14.8) as $F_D = 3\pi\mu DU$

But $F_D = 0.000454$ N

Equating the two values of F_D, we have

$$3\pi\mu DU = 0.000454 \quad \text{or} \quad 3\pi \times \frac{15}{10} \times 2 \times 10^{-3} \times U = 0.000454$$

or $$U = \frac{10 \times 0.000454}{3\pi \times 15 \times 2 \times 10^{-3}} = \mathbf{0.016\ m/s.\ Ans.}$$

Let us check for Reynold number, R_e

$$R_e = \frac{\rho UD}{\mu}, \quad \text{where } \rho = 950 \text{ kg/m}^3$$

$$= 950 \times \frac{0.016 \times 2 \times 10^{-3}}{1.5} = 0.02$$

Hence the Reynolds number is less than 0.2 and so the expression $F_D = 3\pi\mu DU$ for calculating terminal velocity is valid.

Problem 14.20 *Determine the velocity of fall of rain drops of a 30×10^{-3} cm diameter, density 0.0012 gm/cm^3 and kinematic viscosity 0.15 cm^2/s.*

Solution. Given :

Diameter of rain drops, $D = 30 \times 10^{-3}$ cm

Density of rain drops, $\rho = 0.0012$ gm/cm^3

Kinematic viscosity, $\nu = 0.15$ cm^2/s

Using the relation, $\nu = \frac{\mu}{\rho}$ or $0.15 = \frac{\mu}{.0012}$

$\therefore$ $$\mu = 0.15 \times .0012 = 0.00018 \frac{\text{gm}}{\text{cm sec}}$$

Now weight of rain drop $= \rho \times g \times$ Volume of rain drop

$= \rho \times g \times \frac{\pi}{6} D^3$ ($\because$ Rain drop is a sphere)

Drag force, F_D, on rain drop is given by equation (14.8) as $F_D = 3\pi\mu DU$

When rain drop is falling with a uniform velocity U, the drag force must be equal to the weight of rain drop. Hence equating these two values, we get

Weight of rain drop = Drag force

or $$\rho \times g \times \frac{\pi}{6} D^3 = 3\pi\mu DU \quad \text{or} \quad U = \frac{\rho \times g \times \frac{\pi}{6} \times D^3}{3\pi\mu D} = \frac{\rho g D^2}{18\mu}$$

$$= \frac{0.0012 \times 981 \times \left(30 \times 10^{-3}\right)^2}{18 \times .00018} = \textbf{0.327 cm/s. Ans.}$$

Let us check for Reynolds number, R_e

$$R_e = \frac{\rho UD}{\mu} = \frac{UD}{\nu} = \frac{0.327 \times 30 \times 10^{-3}}{0.15} = 0.0654$$

As the Reynolds number is less than 0.2, the expression $F_D = 3\pi\mu DU$ is valid.

▶ 14.6 DRAG ON A CYLINDER

Consider a real fluid flowing over a circular cylinder of diameter D and length L, when the cylinder is placed in the fluid such that its length is perpendicular to the direction of flow. If the Reynolds number of the flow is less than 0.2 $\left(i.e., \frac{U \times d}{\nu} < 0.2\right)$, the inertia force is negligibly small as compared to viscous force and hence the flow pattern about the cylinder will be symmetrical. As the Reynolds number is increased, inertia forces increase and hence they must be taken into consideration for analysis of flow over cylinder. With the increase of the Reynolds number, the flow pattern becomes unsymmetrical with respect to an axis perpendicular to the direction of flow. The drag force, *i.e.*, the force exerted by the flowing fluid on the cylinder in the direction of flow depends upon the Reynolds number of the flow. From experiments, it has been observed that :

(*i*) When Reynolds number (R_e) < 1, the drag force is directly proportional to velocity and hence the drag co-efficient (C_D) is inversely proportional to Reynolds number.

(*ii*) With the increase of the Reynolds number from 1 to 2000, the drag co-efficient decreases and reaches a minimum value of 0.95 at $R_e = 2000$.

(*iii*) With the further increase of the Reynolds number from 2000 to 3×10^4, the co-efficient of drag increases and attains maximum value of 1.2 at $R_e = 3 \times 10^4$.

(*iv*) The value of co-efficient of drag decreases if the Reynolds number is increased from 3×10^4 to 3×10^5. At $R_e = 3 \times 10^5$, the value of $C_D = 0.3$.

(*v*) If the Reynolds number is increased beyond 3×10^6, the value of C_D increases and it becomes equal to 0.7 in the end.

▶ 14.7 DEVELOPMENT OF LIFT ON A CIRCULAR CYLINDER

When a body is placed in a fluid in such a way that its axis is parallel to the direction of fluid flow and body is symmetrical, the resultant force acting on the body is in the direction of flow. There is no force component on the body perpendicular to the direction of flow. But the component to the force on the body perpendicular to the direction of flow, is known as '*Lift*'. Hence in this case lift will be zero.

The lift will be acting on the body when the axis of the symmetrical body is inclined to the direction of flow or body is unsymmetrical. In the case of circular cylinder, the body is symmetrical and the axis is parallel to the direction of flow when cylinder is stationary. Hence the lift will be zero. But if the cylinder is rotated, the axis of the cylinder is no longer parallel to the direction of flow and hence lift will be acting on the rotating cylinder. This is explained by considering the following cases :

14.7.1 Flow of Ideal Fluid over Stationary Cylinder. Consider the flow of an ideal fluid over a cylinder, which is stationary as shown in Fig. 14.9.

Let U = Free stream velocity of fluid
R = Radius of the cylinder
θ = Angle made by any point say C on the circumference of the cylinder with the direction of flow.

The flow pattern will be symmetrical and the velocity at any point say C on the surface of the cylinder is given by $u_\theta = 2U \sin \theta$ ...(14.12)

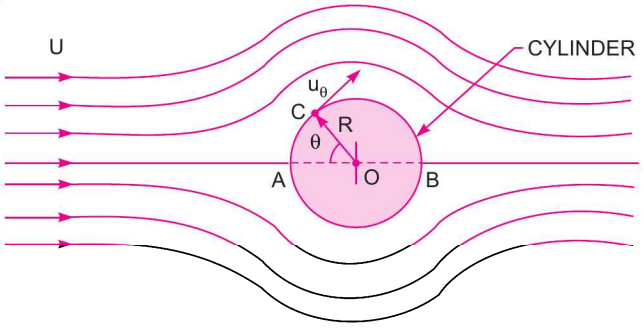


Fig. 14.9 *Flow of ideal fluid over stationary cylinder.*

The velocity distribution over the upper half and lower half of the cylinder from the axis AB of the cylinder are identical and hence the pressure distributions will also be same. Hence the lift acting on the cylinder will be zero.

14.7.2 Flow Pattern Around the Cylinder when a Constant Circulation Γ is Imparted to the Cylinder. Circulation is defined as the flow along a closed curve. Mathematically, the circulation is obtained if the product of the velocity component along the curve at any point and the length of the small element containing that point is integrated around the curve.

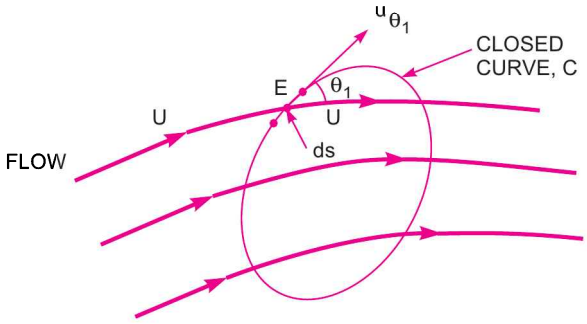


Fig. 14.10 *Circulation.*

Consider a fluid flowing with a free stream velocity equal to U. Within the fluid consider a closed curve as shown in Fig. 14.10. Let E is any point on the closed curve and 'dS' is a small length of the closed curve containing point E.

Let θ_1 = Angle made by the tangent at E with the direction of flow,
u_{θ_1} = Component of free stream velocity along the tangent at E and is given as
$= U \cos \theta_1$

∴ By definition, circulation along the closed curve is

$$\Gamma = \oint \text{velocity component along curve} \times \text{Length of element}$$

$$= \oint U \cos \theta_1 \times dS \qquad ...(14.13)$$

where $\oint$ = Integral for the complete closed curve.

Circulation for the Flow-field in a Free-Vortex. The equation for the free vortex flow is given by

$$u_{\theta_1} \times r = \text{Constant say} = k \qquad ...(i)$$

where u_{θ_1} = velocity of fluid in a free-vortex flow

r = Radius, where velocity is u_{θ_1}.

The flow-pattern for a free-vortex flow consists of streamlines which are series of concentric circles as shown in Fig. 14.11 (*a*).

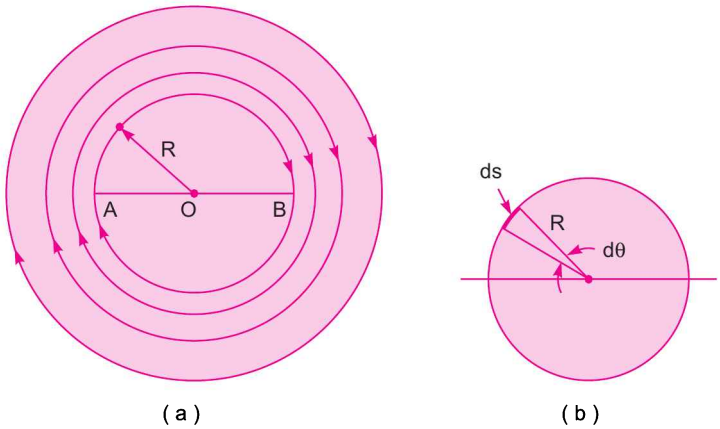


Fig. 14.11 *Stream-lines for free vortex.*

In case of free-vortex flow, the stream velocity at any point on a circle of radius R is equal to the tangential velocity at that point. This means that angle between the stream-lines and tangent on the stream is zero. Also from Fig. 14.11 (*b*), the length of the element '*ds*' is given as $ds = Rd\theta$

∴ For a free-vortex flow, $U = u_{\theta_1}$; $\cos \theta_1 = 1$; $ds = Rd\theta$

Substituting these values in equation (14.13), we get the circulation for a free vortex as

$$\Gamma = \oint u_{\theta_1} \times 1 \times Rd\theta$$

But from equation (*i*), for a radius R, we have

$$u_{\theta_1} \times R = K$$

$$\therefore \quad \Gamma = \oint Kd\theta = 2\pi K \qquad \left(\because \oint d\theta = 2\pi\right)$$

$$= 2\pi u_{\theta_1} \times R \qquad (\because K = u_{\theta_1} \times R)$$

$$\therefore \quad u_{\theta_1} = \frac{\Gamma}{2\pi R}. \qquad ...(14.14)$$

Flow over Cylinder due to Constant Circulation. The flow pattern over a cylinder to which a constant circulation (Γ) is imparted is obtained by combining the flow patterns shown in Figs. 14.9 and Fig. 14.11 (*a*). The resultant flow pattern is shown in Fig. 14.12. The velocity at any point on the surface of the cylinder is obtained by adding equations (14.12) and (14.14) as

$$u = u_\theta + u_{\theta_1} = 2U \sin \theta + \frac{\Gamma}{2\pi R}. \qquad ...(14.15)$$

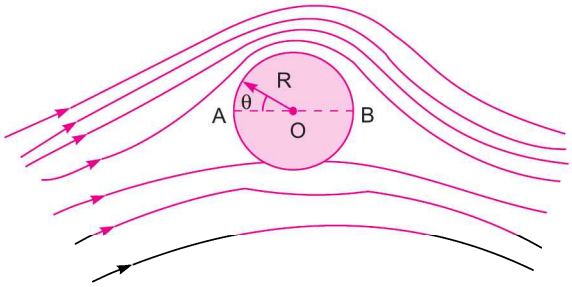


Fig. 14.12 *Flow pattern over a rotating cylinder.*

For the upper half portion of the cylinder, θ varies from 0° to 180° and hence component of velocity, $2U \sin \theta$ is positive. But for the lower half portion of the cylinder, θ varies from 180° to 360°. As $\sin \theta$ for the values of θ more than 180° and less than 360° is negative and hence component of velocity $2U \sin \theta$ will be negative. This means, the velocity on the upper half portion of the cylinder will be more than the velocity on the lower half portion of the cylinder. But from Bernoulli's theorem we know that at a surface where velocity is less, pressure will be more there and *vice-versa*. Hence on the lower half portion of cylinder, where velocity is less, pressure will be more than the pressure on the upper half portion of the cylinder. Due to this difference of pressure on the two portions of the cylinder, a force will be acting of the cylinder in a direction perpendicular to the direction of flow. This force is nothing but a lift force. Thus by rotating a cylinder at constant velocity in a uniform flow field, a lift force can be developed.

14.7.3 Expression for Lift Force Acting on Rotating Cylinder.

Let a cylinder is rotating in a uniform flow field. The resultant flow pattern will be as shown in Fig. 14.12. Consider a small length of the element on the surface of the cylinder.

Let p_s = Pressure on the surface of the element on cylinder

ds = Length of element

R = Radius of cylinder

$d\theta$ = Angle made by the length ds at the centre of the cylinder as shown in Fig. 14.13.

p = Pressure of the fluid far away from the cylinder

U = Velocity of fluid far away from the cylinder

u_s = Velocity of fluid on the surface of the cylinder.

Applying Bernoulli's equation to a point far away from cylinder and to a point lying on the surface of cylinder such that both the points are on the same horizontal line, we have

$$\frac{p}{\rho g}+\frac{U^2}{2g}=\frac{p_s}{\rho g}+\frac{u_s^2}{2g}$$

$$\therefore \quad \frac{p_s}{\rho g}=\frac{p}{\rho g}+\frac{U^2}{2g}-\frac{u_s^2}{2g}$$

$$=\frac{p}{\rho g}+\frac{U^2}{2g}\left[1-\frac{u_s^2}{U^2}\right] \quad ...(i)$$

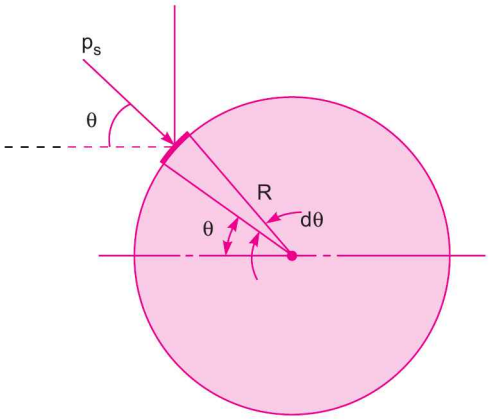


Fig. 14.13 *Lift on a rotating cylinder.*

But the velocity on the surface of the cylinder is given by equation (14.15). Hence

$$u_s = u = 2U \sin\theta + \frac{\Gamma}{2\pi R}$$

Substituting this value of u_s in (i), we get

$$\frac{p_s}{\rho g}=\frac{p}{\rho g}+\frac{U^2}{2g}\left[1-\frac{\left(2U\sin\theta+\frac{\Gamma}{2\pi R}\right)^2}{U^2}\right]$$

$$=\frac{p}{\rho g}+\frac{U^2}{2g}\left[1-\frac{\left(4U^2\sin^2\theta+\frac{\Gamma^2}{4\pi^2R^2}+4U\sin\theta\frac{\Gamma}{2\pi R}\right)}{U^2}\right]$$

$$=\frac{p}{\rho g}+\frac{U^2}{2g}\left[1-\left(4\sin^2\theta+\frac{\Gamma^2}{4\pi^2R^2U^2}+\frac{4\sin\theta\,\Gamma}{U\times 2\pi R}\right)\right]$$

or
$$p_s = p+\frac{\rho g U^2}{2g}\left[1-4\sin^2\theta-\frac{\Gamma^2}{4\pi^2R^2U^2}-\frac{4\sin\theta\,\Gamma}{U\times 2\pi R}\right] \quad ...(ii)$$

From Fig. 14.13, we have the lift force acting on the small length ds on the element, due to pressure p_s as

= Component of p_s in the direction perpendicular to flow × Area of the element

= $(-p_s \sin\theta) \times (ds \times L)$

Negative sign is taken, as the component of p_s perpendicular to the flow is acting in the downward direction.

Now L = length of the cylinder

$ds = R \times d\theta$

$\therefore$ Lift force on the element $= -p_s \sin\theta \times Rd\theta \times L$ ($\because$ $ds = Rd\theta$) ...(iii)

The total force is obtained by integrating equation (iii) over the centre surface of the cylinder.

$\therefore$ Total lift, $F_L = \int_0^{2\pi} -p_s \sin\theta \times Rd\theta \times L = \int_0^{2\pi} -p_s \times R \times L \times \sin\theta\, d\theta$

Substituting the value of p_s from equation (ii), we get

$$F_L = \int_0^{2\pi} -\left[p+\frac{\rho g U^2}{2g}\left(1-4\sin^2\theta-\frac{\Gamma}{4\pi^2R^2U^2}-\frac{4\sin\theta\,\Gamma}{U\times 2\pi R}\right)\right] RL\times\sin\theta\, d\theta$$

$$= -RL \int_0^{2\pi} \left[p \sin\theta + \frac{\rho g U^2}{2g} \left(\sin\theta - 4\sin^3\theta - \frac{\Gamma \sin\theta}{4\pi^2 R^2 U^2} - \frac{4\sin^2\theta\,\Gamma}{U \times 2\pi R} \right) \right] d\theta$$

But $$\int_0^{2\pi} \sin\theta\, d\theta = \int_0^{2\pi} \sin^3\theta\, d\theta = 0$$

$\therefore$ $$F_L = -R \times L \int_0^{2\pi} \frac{\rho g U^2}{2g} \left(-\frac{4\sin^2\theta\,\Gamma}{U \times 2\pi R} \right) d\theta$$

$$= R \times L \times \frac{\rho g U^2}{2g} \times \frac{4\Gamma}{U \times 2\pi R} \int_0^{2\pi} \sin^2\theta\, d\theta = \frac{L}{g} \frac{\rho g U \Gamma}{\pi} \int_0^{2\pi} \sin^2\theta\, d\theta$$

But $$\int_0^{2\pi} \sin^2\theta\, d\theta = \left[\frac{\theta}{2} - \frac{\sin 2\theta}{4} \right]_0^{2\pi} = \left(\frac{2\pi}{2} - \frac{\sin 4\pi}{4} \right) = \pi$$

$\therefore$ $$F_L = \frac{L}{g} \frac{\rho g}{\pi} U\Gamma \times \pi = \frac{L}{g} \rho g U \Gamma = \frac{\rho g}{g} LU\Gamma = \rho LU\Gamma \qquad ...(14.16)$$

Equation (14.16) is known as Kutta-Joukowski equation.

14.7.4 Drag Force Acting on a Rotating Cylinder. The resultant flow pattern for a rotating cylinder in a uniform flow field is shown in Fig. 14.12. The resultant flow pattern is symmetrical about the vertical axis of the cylinder. Hence the velocity distribution and also pressure distribution is symmetrical about the vertical axis and as such there will be no drag on the cylinder.

14.7.5 Expression for Lift Co-efficient for Rotating Cylinder. The lift co-efficient is defined by the equation (14.4) as

$$F_L = C_L A \frac{\rho U^2}{2} \qquad ...(i)$$

where C_L = Lift co-efficient, A = Projected area

U = Free stream velocity or uniform velocity of flow.

For a rotating cylinder, the lift force is given by equation (14.16)

$$F_L = \rho LU\Gamma$$

A = Projected area of cylinder = $2RL$

$\therefore$ Substituting these values in equation (i), we get

$$\rho LU\Gamma = C_L \times 2RL \times \frac{\rho U^2}{2} \quad \text{or} \quad C_L = \frac{\rho LU\Gamma}{RL\rho U^2} = \frac{\Gamma}{RU} \qquad ...(14.17)$$

From equation (14.14), we have $u_{\theta_1} = \frac{\Gamma}{2\pi R}$ or $\frac{\Gamma}{R} = 2\pi u_{\theta_1}$

Substituting this value of $\frac{\Gamma}{R}$ in equation (14.17), C_L is also expressed

$$C_L = \frac{2\pi u_{\theta_1}}{U} \qquad ...(14.18)$$

where u_{θ_1} = Velocity of rotation of the cylinder in the tangential direction.

14.7.6 Location of Stagnation Points for a Rotating Cylinder in a Uniform Flow-field. Stagnation points are those points on the surface of the cylinder, where velocity is zero. For a rotating cylinder as shown in Fig. 14.12, the resultant velocity is given by equation (14.15) as

$$u = 2U \sin\theta + \frac{\Gamma}{2\pi R}.$$

For stagnation point, $u = 0$

$$\therefore \quad 2U \sin\theta + \frac{\Gamma}{2\pi R} = 0 \text{ or } 2U \sin\theta = -\frac{\Gamma}{2\pi R}$$

or
$$\sin\theta = -\frac{\Gamma}{4\pi UR}. \qquad ...(14.19)$$

The solution of equation (14.19) gives the location of stagnation points on the surface of the cylinder. There are two values of θ, which satisfy equation (14.19). As $\sin\theta$ is negative in equation (14.19), it means θ is more than 180° but less than 360°. The two values of θ are such that one value is between 180° and 270° and other value is between 270° and 360°.

For a single stagnation point, $\theta = 270°$ and then equation (14.19) becomes as

$$\sin 270° = -\frac{\Gamma}{4\pi UR} \text{ or } -1 = -\frac{\Gamma}{4\pi UR} \qquad (\because \quad \sin 270° = -1)$$

$$\therefore \quad \Gamma = 4\pi UR. \qquad ...(14.20)$$

14.7.7 Magnus Effect. When a cylinder is rotated in a uniform flow, a lift force is produced on the cylinder. This phenomenon of the lift force produced by a rotating cylinder in a uniform flow is known as Magnus Effect. This fact was investigated by a German physicist H.G. Magnus and hence the name is given as Magnus Effect.

Problem 14.21 *A cylinder rotates at 150 r.p.m. with its axis perpendicular in an air stream which is having uniform velocity of 25 m/s. The cylinder is 1.5 m in diameter and 10 m long. Assuming ideal fluid theory, find (i) the circulation, (ii) lift force, and (iii) position of stagnation points. Take density of air as 1.25 kg/m^3.*

Solution. Given :

Speed of cylinder, $N = 150$ r.p.m.

Velocity of air, $U = 25$ m/s

Diameter of cylinder, $D = 1.5$ m

$\therefore$ Radius of cylinder, $R = \frac{D}{2} = \frac{1.5}{2} = 0.75$ m

Length of cylinder, $L = 10$ m

Density of air, $\rho = 1.25$ kg/m^3

Tangential velocity of cylinder is given as $u_\theta = \frac{\pi DN}{60} = \frac{\pi \times 1.5 \times 150}{60} = 11.78$ m/s.

(*i*) Circulation (Γ) is obtained from equation (14.14), as $u_\theta = \frac{\Gamma}{2\pi R}$

$$\therefore \quad \Gamma = 2\pi R \times u_\theta = 2\pi \times 0.75 \times 11.78 = \mathbf{55.51\ m^2/s.} \quad \textbf{Ans.}$$

(*ii*) Lift force, F_L is given by equation (14.16) as

$$F_L = \rho LU\Gamma = 1.25 \times 10 \times 25 \times 55.51$$
$$= \mathbf{17344\ N.} \quad \textbf{Ans.}$$

(*iii*) Position of stagnation points are given by equation (14.19) as

$$\sin\theta = -\frac{\Gamma}{4\pi UR} = -\frac{55.51}{4\pi \times 25 \times 0.75} = 0.2356$$

$$= -\sin(13.62°)$$

$$= \sin[180° + 13.62°] \text{ and } \sin[360° - 13.62°]$$

$$\therefore \quad \theta = (180° + 13.62°) \text{ and } (360° - 13.62°)$$

$$= \mathbf{193.62° \text{ and } 346.38°.} \quad \textbf{Ans.}$$

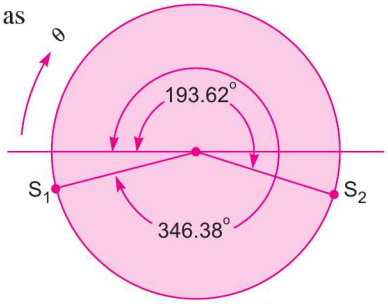


Fig. 14.14

The location of stagnation points are shown in Fig. 14.14.

Problem 14.22 *A cylinder whose axis is perpendicular to the stream of air having a velocity of 20 m/s, rotates at 300 r.p.m. The cylinder is 2 m in diameter and 10 m long. (a) Find : (i) the circulation, (ii) theoretical lift force per unit length, (iii) position of stagnation points, and (iv) the actual lift, drag and direction of resultant force. Take density of air 1.24 kg/m^3. For actual drag and lift, take $C_L = 3.4$, $C_D = 0.65$ and $\frac{u_\theta}{U} = 1.57$. (b) Find the speed of rotation of the cylinder which will give only a single stagnation point.*

Solution. Given :

Velocity of air,	$U = 20$ m/s
Speed of rotation,	$N = 300$ r.p.m.
Diameter of cylinder,	$D = 2$ m
Length of cylinder,	$L = 10$ m
Density of air,	$\rho = 1.24$ kg/m^3

Tangential velocity of cylinder is given as $u_\theta = \frac{\pi DN}{60} = \frac{\pi}{60} \times 2.0 \times 300 = 31.42$ m/s.

(a) (*i*) Now the circulation (Γ) is given by equation (14.14) as $u_\theta = \frac{\Gamma}{2\pi R}$

$$\therefore \quad \Gamma = 2\pi R u_\theta = 2\pi \times \frac{D}{2} \times 31.42$$

$$= 2\pi \times \frac{3}{2} \times 31.42 = \mathbf{197.41 \text{ m}^2\text{/s.}} \quad \textbf{Ans.}$$

(*ii*) The theoretical lift (F_L) is given by equation (14.16) as

$$F_L = \rho UL\Gamma = 1.24 \times 20 \times 10 \times 197.41 = 48957.7 \text{ N}$$

$$\therefore \quad \text{Theoretical lift per unit length} = \frac{F_L}{L} = \frac{48957.7}{10} = \mathbf{4895.77 \text{ N/m.}} \quad \textbf{Ans.}$$

(*iii*) Position of stagnation points are obtained from equation (14.19) as

$$\sin\theta = -\frac{\Gamma}{4\pi UR} = -\frac{197.41}{4\pi \times 20 \times D/2} = \frac{197.41}{4\pi \times 20 \times 1} = -0.7854$$

$$= \sin[180° + 51.75°] \text{ and } \sin[360° - 51.75°]$$

$$= \sin[231.75°] \text{ and } \sin[308.25°]$$

$$\therefore \quad \theta = \mathbf{231.75° \text{ and } 308.25°.} \quad \textbf{Ans.}$$

Stagnation points will be at an angle of 231.75° and 308.25°.

(*iv*) *Actual Lift, Drag and Direction of Resultant Force*

For actual lift and drag, given $\frac{u_\theta}{U} = 1.57$, $C_L = 3.4$ and $C_D = 0.65$.

The ratio of $\frac{u_\theta}{U}$ from theoretical consideration is given as $\frac{u_\theta}{U} = \frac{31.42}{20} = 1.57$

Now actual lift force is given by $F_L = \frac{1}{2} \rho A U^2 \times C_L$

$$= \frac{1}{2} \times 1.24 \times (2 \times 10) \times 20^2 \times 3.4 = 16864 \text{ N}$$

where A = projected area of cylinder = 2×10 m^2

$\therefore$ F_L = **16864 N. Ans.**

Actual drag force, $F_D = \frac{1}{2} \rho A U^2 \times C_D = \frac{1}{2} \times 1.24 \times (2 \times 10) \times 20^2 \times 0.65$ = **3224 N. Ans.**

$\therefore$ Resultant force $= \sqrt{F_L^2 + F_D^2} = \sqrt{16864^2 + 3224^2}$

$= \sqrt{284394496 + 10394176}$ = **17169.4 N. Ans.**

The direction of the resultant force with the horizontal is given by

$$\tan \theta = \frac{F_L}{F_D} = \frac{16864}{3224} = 5.23$$

$\therefore$ $\theta = \tan^{-1} 5.23$ = **79.1°. Ans.**

(*b*) Speed of rotation of the cylinder for single stagnation point.

For a single point stagnation, the equation (14.20) is used.

$\therefore$ $\Gamma = 4\pi UR = 4\pi \times 20 \times 1 = 251.32$ m^2/s $\left(\because \quad R = \frac{D}{2} = \frac{2}{2} = 1 \text{ m}\right)$

The speed of rotation, corresponding to the circulation = 251.32 is given by equation (14.14) as

$$u_\theta = \frac{\Gamma}{2\pi R} = \frac{251.32}{2\pi \times 1} = 40.0$$

But $u_\theta = \frac{\pi D N}{60}$

$\therefore$ $N = \frac{60 \times u_\theta}{\pi \times D} = \frac{60 \times 40}{\pi \times 2.0} = 381.97$ r.p.m. $\simeq$ **382.0 (say) r.p.m. Ans.**

Problem 14.23 *The air having a velocity of 40 m/s is flowing over a cylinder of diameter 1.5 m and length 10 m, when the axis of the cylinder is perpendicular to the air stream. The cylinder is rotated about its axis and a lift of 6867 N per metre length of the cylinder is developed. Find the speed of rotation and location of the stagnation points. The density of air is given as 1.25 kg/m^3.*

Solution. Given :

Velocity of air, $U = 40$ m/s

Diameter of cylinder, $D = 1.5$ m

Length of the cylinder, $L = 10$ m

Lift/metre length, $\frac{F_L}{L} = 6867$ N

Density of air, $\rho = 1.25$ kg/m^2

From equation (14.16), we have $F_L = \rho L U \Gamma$ or $\frac{F_L}{L} = \rho U \Gamma$

$\therefore \quad 6867 = 1.25 \times 40 \times \Gamma$

$\therefore \quad \Gamma = \frac{6867}{1.25 \times 40} = 137.36$ m^2/s.

Let the speed of rotation corresponding to circulation 137.36 = u_θ. Using equation (14.14),

$$u_\theta = \frac{\Gamma}{2\pi R} = \frac{137.36}{2\pi \times \frac{D}{2}} = \frac{137.36 \times 2}{2\pi \times 1.5} = 29.15 = \frac{\pi D N}{60}$$

$\therefore \quad N = \frac{60 \times 29.15}{\pi \times D} = \frac{60 \times 29.15}{\pi \times 1.5} =$ **371.15 r.p.m. Ans.**

Position of stagnation points are given by equation (14.19)

$$\sin\theta = -\frac{\Gamma}{4\pi U R} = -\frac{137.36}{4\pi \times 40 \times \frac{D}{2}} = -\frac{137.36 \times 2}{4\pi \times 53 \times 1.5}$$

$= -0.3643 = -\sin 21.36°$

$= \sin(180° + 21.36°)$ and $\sin[360° - 21.36°]$

$= \sin 201.36°$ and $\sin 338.64°$

$\therefore \quad \theta =$ **201.36° and 338.64°. Ans.**

▶ 14.8 DEVELOPMENT OF LIFT ON AN AIRFOIL

Fig. 14.15 shows the two shapes of the airfoils, which are stream-lines bodies which may be symmetrical or unsymmetrical in shapes. The airfoil is characterized by its chord length C, angle of attack α (which is the angle between the direction of the fluid flowing and chord line) and span L of the airfoil. The lift on the airfoil is due to negative pressure created on the upper part of the airfoil. The drag force on the airfoil is always small due to the design of the shape of the body, which is stream-lined.

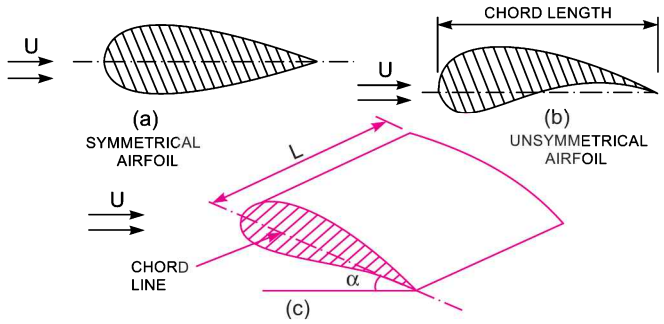


Fig. 14.15 *Shapes of airfoils.*

From the theoretical analysis, the circulation Γ developed on the airfoil so that the stream-line at the trailing edge of the airfoil is tangential to the airfoil, is given as

$$\Gamma = \pi CU \sin \alpha \qquad ...(14.21)$$

where C = Chord length, U = Free stream velocity of airfoil, α = Angle of attack

Lift force F_L is given by equation (14.16) as

$$F_L = \rho UL\Gamma = \rho UL \times \pi CU \sin \alpha \qquad (\because \Gamma = \pi CU \sin \alpha)$$
$$= \pi \rho CU^2 L \sin \alpha \qquad ...(14.22)$$

The lift force is also given by equation (14.4), as

$$F_L = C_L \times A \times \frac{\rho U^2}{2}$$

where C_L = Co-efficient of lift

A = Projected area = $C \times L$ for airfoil

$$\therefore \qquad F_L = C_L \times C \times L \times \frac{\rho U^2}{2} \qquad ...(14.23)$$

Equating the two value of lift force given by equations (14.22) and (14.23), we get

$$\pi \rho CU^2 L \sin \alpha = C_L \times C \times L \times \frac{\rho U^2}{2}$$

$$\therefore \qquad C_L = \frac{2\pi \rho CU^2 L \sin \alpha}{C \times L \times \rho U^2} = 2\pi \sin \alpha \qquad ...(14.24)$$

Thus it is clear from equation (14.23), that co-efficient of lift depends upon the angle of attack.

14.8.1 Steady-state of a Flying Object. When a flying object for example airplane is in a steady-state, the weight of the airplane is equal to the lift force and thrust developed by the engine is equal to the drag force. Hence

$$W = \text{Lift force} = C_L \frac{\rho AU^2}{2} \qquad ...(14.25)$$

where W = Weight of the airplane and $C_L \frac{\rho AU^2}{2}$ = Lift force.

Problem 14.24 *An airfoil of chord length 2 m and of span 15 m has an angle of attack as 6°. The airfoil is moving with a velocity of 80 m/s in air whose density is 1.25 kg/m^3. Find the weight of the airfoil and the power required to drive it. The values of co-efficient of drag and lift corresponding to angle of attack are given as 0.03 and 0.5 respectively.*

Solution. Given :

Chord length, $C = 2$ m

Span of airfoil, $L = 15$ m

Angle of attack, $\alpha = 6°$

Velocity of airfoil, $U = 80$ m/s

Density of air, $\rho = 1.25$ kg/m^3

Co-efficient of drag, $C_D = 0.03$

Co-efficient of lift, $C_L = 0.50$

From equation (14.25), we know that

$$\text{Weight of airfoil} = \text{Lift force} = C_L \frac{\rho AU^2}{2} = 0.50 \times 1.25 \times (C \times L) \times \frac{80^2}{2}$$

$$= 0.50 \times 1.25 \times (2 \times 15) \times \frac{80^2}{2} = \textbf{60000 N. Ans.}$$

Now drag force, $F_D = C_D \times \rho \times \frac{AU^2}{2}$

$$= 0.03 \times 1.25 \times \frac{(2 \times 15) \times 80^2}{2} = \textbf{3600 N. Ans.}$$

$\therefore$ Power required in kW $= \frac{F_D \times U}{1000} = \frac{3600 \times 80}{1000} = \textbf{288 kW. Ans.}$

Problem 14.25 *A jet plane which weighs 29430 N and has a wing area of 20 m^2 flies at a velocity of 250 km/hr. When the engine delivers 7357.5 kW. 65% of the power is used to overcome the drag resistance of the wing. Calculate the co-efficient of lift and co-efficient of drag for the wing. Take density of air equal to 1.21 kg/m^3.*

Solution. Given :

Weight of plane, $W = 29430$ N

Wing area, $A = 20$ m^2

Velocity of plane, $U = 250$ km/hr $= \frac{250 \times 1000}{60 \times 60}$ m/s $= 69.44$ m/s

Power delivered by engine $= 7357.5$ kW

Power required to overcome drag resistance

$= 65\%$ of $7357.5 = 0.65 \times 7357.5 = 4782.375$ kW.

Density of air, $\rho = 1.21$ kg/m^3

Now, weight of plane = Lift force $= C_L \times A \times \frac{\rho U^2}{2}$

$\therefore \quad 29430 = C_L \times 20 \times 1.21 \times \frac{69.44^2}{2}$

$\therefore \quad C_L = \frac{29430 \times 2}{20 \times 1.21 \times 69.44^2} = \textbf{0.5046. Ans.}$

Let F_D = Drag force

Power required to overcome drag resistance = $\frac{F_D \times U}{1000}$ kW

$\therefore \quad 4782.375 = \frac{F_D \times 69.44}{1000}$

$\therefore \quad F_D = \frac{4782.375 \times 1000}{69.44} = 68870.6$ N

Now drag force, $F_D = C_D \times A \times \frac{\rho U^2}{2}$

$\therefore \quad 68870.6 = C_D \times 20 \times \frac{1.21 \times 69.44^2}{2}$

$\therefore \quad C_D = \frac{68870.6 \times 2}{20 \times 1.21 \times 69.44^2} = \textbf{1.18. Ans.}$

HIGHLIGHTS

1. The force exerted by a fluid on a solid body immersed in the fluid in the direction of motion is called drag force while the force perpendicular to the direction of motion, on the body is known as lift force.
2. The mathematical expression for the drag and lift force are,

$$F_L = C_D A \frac{\rho U^2}{2} \; ; F_L = C_L A \frac{\rho U^2}{2}$$

where C_D = Co-efficient of drag, C_L = Co-efficient of lift,
A = Projected area of the body, ρ = Density of fluid,
U = Free-stream velocity of fluid.

3. The resultant force exerted by fluid on solid body is $F_R = \sqrt{F_D^2 + F_L^2}$.
4. Total drag on a body is the sum of pressure drag and friction drag.
5. A body whose surface coincides with the stream-lines, when the body is placed in a flow is called stream-lined body. If the surface of the body does not coincide with the stream-lines, the body is called bluff body.
6. The drag on a sphere for Reynolds number less than 0.2 is given by $F_D = 3\pi\mu DU$

Out of this total drag, Skin friction drag $= \frac{2}{3} \times 3\pi\mu DU = 2\pi\mu DU$

and pressure drag $= \frac{1}{3} \times 3\pi\mu DU = \pi\mu DU.$

7. Values of C_D for sphere for different Reynold number is

$$C_D = \frac{24}{R_e} \text{ ... when } R_e < 0.2$$

$$= \frac{24}{R_e}\left[1 + \frac{3}{16R_e}\right] \text{ ..... when } 0.2 < R_e < 5.0$$

$$= 0.4 \text{ ..... when } R_e \text{ lies between 5 and 1000}$$

$$= 0.5 \text{ ..... when } R_e \text{ lies between 100 and 100000}$$

$$= 0.2 \text{ ..... when } R_e > 10^5.$$

8. Terminal velocity is defined as the maximum constant velocity of a falling body with which it will travel. At the terminal velocity, the weight of the body is equal to the drag force plus the buoyant force. Hence $W = F_D + F_B$.
9. The velocity of ideal fluid at any point on the surface of the cylinder is given by

$$u_\theta = 2U \sin\theta$$

where u_θ = Tangential velocity on the surface of the cylinder
U = Uniform velocity or free stream velocity
θ = Angle made by any point on the surface of the cylinder with the direction of flow.

10. Circulation is the flow along a closed curve and is obtained when the product of velocity along the closed curve and length of the small element is integrated around the curve. Circulation for free vortex at any radius R is given by

$$\Gamma = 2\pi R \times u_\theta.$$

11. The resultant velocity on a circular cylinder which is rotated at constant speed in uniform flow-field is

$$u = 2U \sin\theta + \frac{\Gamma}{2\pi R}.$$

12. When a circular cylinder is rotated in a uniform flow, a lift force is produced on the cylinder. The magnitude of the lift force F_L is given by

$$F_L = \rho L U \Gamma$$

where L = Length of cylinder, U = Free stream velocity, Γ = Circulation.

13. The expression for lift co-efficients for a rotating cylinder in a uniform flow is given by

$$C_L = \frac{\Gamma}{RU} \quad \text{...(in term of circulation)}$$

$$= \frac{2\pi u_\theta}{U} \quad \text{...(in term of tangential speed)}$$

14. The location of stagnation points is given by $\sin\theta = -\dfrac{\Gamma}{4\pi UR}$

where R = Radius of cylinder, U = Free stream velocity.

15. For a single stagnation point, the condition is

$$\Gamma = 4\pi UR \quad \text{...(in terms of circulation)}$$

or $$u_\theta = 2U \quad \text{...(in terms of tangential velocity)}$$

16. Circulation developed on the airfoil is given by

$$\Gamma = \pi C U \sin\alpha$$

where C = Chord length, U = Velocity of airfoil, α = Angle of attack.

17. The expression for co-efficient of lift for an airfoil is $C_L = 2\pi \sin\alpha$.

18. When an airplane is in steady-state,

Weight of plane = Lift force

Thrust by engine = Drag force.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define the terms : drag and lift.

2. What do you understand by : Total drag on a body, resultant force on a body, co-efficient of drag and co-efficient of lift.

3. Differentiate between (*i*) stream-lines body and bluff body, (*ii*) Friction drag and pressure drag.

4. (*a*) What is the expression for the drag on a sphere, when the Reynolds number of the flow is upto 0.2 ? Hence prove that the co-efficient of drag for sphere for this range of the Reynolds number is given by

$$C_D = \frac{24}{R_e}, \text{ where } R_e = \text{Reynolds number.}$$

(*b*) Draw C_D versus R_e diagram for a sphere and explain why C_D suddenly drops at $R_e = 3 \times 10^3$.

(*c*) Draw pressure distribution diagrams in dimensionless form for flow past sphere when fluid has no viscosity, when $R_e = 10^4$ and when $R_e = 10^6$.

5. What do you mean by 'Terminal velocity of a body' ? What is the relation between the weight of the body, drag force on the body and buoyant force when the body has acquired terminal velocity ?

6. What is circulation ? Find an expression for circulation for a free-vortex of radius R.

7. Obtain an expression for the lift produced on a rotating cylinder placed in a uniform flow field such that the axis of the cylinder is perpendicular to the direction of flow.

8. What is Magnus effect ? Why is it known as Magnus effect ?

9. Prove that the co-efficient of lift for a rotating cylinder placed in a uniform flow is given by

$$C_L = \frac{\Gamma}{RU}$$

where Γ = Circulation, R = Radius of cylinder, U = Free-stream velocity.

10. Define stagnation points. How the position of the stagnation points for a rotating cylinder in a uniform flow is determined ? What is the condition for single stagnation point ?
11. Define the terms : Airfoil, chord length, angle of attack, span of an airfoil.
12. If the circulation developed on an airfoil is equal to $\pi CU \sin \alpha$, then prove that co-efficient of lift for airfoil is given by $C_L = 2\pi \sin \alpha$, where α = angle of attack.
13. Explain the terms : (*i*) Friction drag, (*ii*) Pressure drag and profile drag.
14. (*a*) How are drag and lift forces caused on a body immersed in a moving fluid ?
(*b*) What is the drag force on a sphere in the stoke range ?

(B) NUMERICAL PROBLEMS

1. A flat plate 2 m × 2 m moves at 40 km/hour in stationary air of density 1.25 kg/m^3. If the co-efficient of drag and lift are 0.2 and 0.8 respectively, find : (*i*) the lift force, (*ii*) the drag force, (*iii*) the resultant force, and (*iv*) the power required to keep the plate in motion.
[**Ans.** (*i*) 246.86 N, (*ii*) 61.715 N, (*iii*) 254.4 N, (*iv*) 0.684 kW]
2. Find the drag force difference on a flat plate of size 1.5 m × 1.5 m when the plate is moving at a speed of 5 m/s normal to its plate first in water and second in air of density 1.24 kg/m^3. Co-efficient of drag is given as 1.10. [**Ans.** 30899 N]
3. A truck having a projected area of 12 square metres travelling at 60 km/hr has a total resistance of 2943 N. Of this 25% is due to rolling friction and 15% is due to surface fraction. The rest is due to form drag. Calculate the coefficient of form drag if the density of air = 1.25 kg/m^3. [**Ans.** 0.847]
4. A circular disc 4 m in diameter is held normal to 30 m/s wind of density 1.25 kg/m^3. If the co-efficient of drag of disc = 1.1, what force is required to hold the disc at rest ? [**Ans.** 7775.4 N]
5. Find the diameter of a parachute with which a man of mass 80 kg descends to the ground from an aeroplane against the resistance of air, with a velocity of 25 m/s. Take C_d = 0.5 and density of air = 1.25 kg/m^3. [**Ans.** 2.26 m]
6. A man descends to the ground with the help of a parachute from an aeroplane against the resistance of air with a uniform velocity of 10 m/s. The parachute is hemispherical in shape and is having diameter of 5 m. Find the weight of man if C_d = 0.5 and density of air = 1.25 kg/m^3. [**Ans.** 613.52 N]
7. A kite 60 cm × 60 cm weighing 2.943 N assumes an angle of 10° to the horizontal. The string attached to the kite makes an angle of 45° to the horizontal. If the pull on the string is 29.43 N when the wind is flowing at a speed of 40 km/hr. Find the corresponding co-efficient to drag and lift. Density of air is given as 1.25 kg/m^3. [**Ans.** C_D = 0.7489, C_L = .8548]
8. The air is flowing over a cylinder of diameter 100 mm and of infinite length with a velocity of 150 mm/s. Find the total drag, shear drag and pressure drag on 1 m length of the cylinder if the total drag co-efficient = 1.5 m and shear drag co-efficient = 0.25. The density of air is given as = 1.25 kg/m^3.
[**Ans.** 0.00211 N, 0.000351 N, 0.001756 N]
9. A body of length 2.5 m has a projected area 1.8 m^2 normal to the direction of its motion. The body is moving through water with a velocity such that the Reynold number = 6×10^6 and the drag co-efficient = 0.5. Find the drag on the body. Take viscosity of water = 0.01 poise. [**Ans.** 2592 N]
10. Calculate the weight of a ball of diameter 50 mm which is just supported in a vertical air stream which is flowing at a velocity of 10 m/s. The density of air = 1.25 kg/m^3 and kinematic viscosity = 1.5 stokes.
[**Ans.** 0.0613 N]

11. A metallic sphere of sp. gr. 8.0 falls in an oil of density 800 kg/m^3. The diameter of the sphere is 10 mm and it attains a terminal velocity of 50 mm/s. Find the viscosity of the oil in poise. [**Ans.** 78.48 poise]

12. A metallic ball of diameter 5 mm drops in a fluid of sp. gr. 0.8 and viscosity 30 poise. The specific gravity of the metallic ball, is 9.0. Find : (*i*) the drag force exerted by fluid on metallic ball, (*ii*) the pressure drag and skin friction drag, and (*iii*) terminal velocity of ball in fluid.
[**Ans.** (*i*) 0.005264 N, (*ii*) 0.001754 N, 0.003527 N, (*iii*) 3.7 cm/s]

13. A cylinder rotates at 200 r.p.m. with its axis perpendicular in an air stream which is having uniform velocity of 20 m/s. The cylinder is 2 m in diameter and 8 m long. Assuming ideal fluid theory, find (*i*) the circulation, (*ii*) lift force, and (*iii*) position of stagnation points. Take density of air as 1.25 kg/m^3.
[**Ans.** (*i*) 131.57 m^2/s, (*ii*) 26309.4 N, (*iii*) $\theta = 211.56°$ and 328.44°]

14. For the problem 13, find the speed of rotation of the cylinder which will give only a single stagnation point. [**Ans.** 381.97 r.p.m.]

15. The air having a velocity of 30 m/s is flowing over a cylinder of diameter 1.4 m and length 10 m, when the axis of the cylinder is perpendicular to the air stream. The cylinder is rotated about its axis and a total lift of 58860 N is produced. Find the speed of rotation and location of the stagnation points. The density of air is given as 1.25 kg/m^3. [**Ans.** $N = 486.87$ r.p.m., $\theta = 216.5°$ and 323.5°]

16. A jet plane which weighs 19620 N has a wing area of 25 m^2. It is flying at a speed of 200 km per hour. When the engine develops 588.6 kW, 70% of this power is used to overcome the drag resistance of the wing. Calculate the co-efficient of lift and co-efficient of drag for the wing. Taken density of air as 1.25 kg/m^3. [**Ans.** $C_L = .407$, $C_D = .114$]

17. Experiments were conducted in a wind tunnel with a wind speed of 50 km/hour on a flat plate of size 2 m long and 1 m wide. The density of air is 1.15 kg/m^3. The plate is kept at such an angle that co-efficients of lift and drag are 0.75 and 0.15 respectively. Determine : (*i*) lift force, (*ii*), drag force, (*iii*) resultant force, (*iv*) its direction, and (*v*) power exerted by the air stream on the plate.
[**Ans.** (*i*) 166.4 N, (*ii*) 33.28 N, (*iii*) 169.64 N, (*iv*) 78.7°, (*v*) 0.461 kW]

15 CHAPTER

COMPRESSIBLE FLOW

▶ 15.1 INTRODUCTION

Compressible flow is defined as that flow in which the density of the fluid does not remain constant during flow. This means that the density changes from point to point in compressible flow. But in case of incompressible flow, the density of the fluid is assumed to be constant. In the previous chapters, the fluid was assumed incompressible, and the basic equations such as equation of continuity, Bernoulli's equation and impulse momentum equations were derived on the assumption that fluid is incompressible. This assumption is true for flow of liquids, which are incompressible fluids. But in case of flow of fluids, such as

(*i*) flow of gases through orifices and nozzles,

(*ii*) flow of gases in machines such as compressors, and

(*iii*) projectiles and airplanes flying at high altitude with high velocities, the density of the fluid changes during the flow. The change in density of a fluid is accompanied by the changes in pressure and temperature and hence the thermodynamic behaviour of the fluids will have to be taken into account.

▶ 15.2 THERMODYNAMIC RELATIONS

The thermodynamic relations have been discussed in Chapter 1, which are as follows :

15.2.1 Equation of State. Equation of state is defined as the equation which gives the relationship between the pressure, temperature and specific volume of a gas. For a perfect gas the equation of state is

$$p\forall = RT \qquad ...(15.1)$$

where p = Absolute pressure in kgf/m^2 or N/m^2

$\forall$ = Specific volume or volume per unit mass

T = Absolute temperature = 273 + $t°$ (centigrade)

R = Gas constant in kgf-m/kg °K or (J/kg K)

= 29.2 kgf-m/kg °K or 287 J/kg K for air.

In equation (15.1), $\forall$ is the specific volume which is the reciprocal of density or

$$\forall = \frac{1}{\rho}.$$

Substituting this value of $\forall$ in equation (15.1), we get

$$\frac{p}{\rho} = RT \qquad ...(15.2)$$

Note. In the equation of state given by equation (15.2), the dimensions of p, ρ and R should be used with care. The following points must be remembered :

1. If the value of R is given as 29.2 kgf-m/kg °K for air, the corresponding value of p and ρ should be taken in kgf/m^2 and kg/m^3. The mass rate of flow of the gas will be in kg/sec.
2. If the value of R is given as 287 J/kg K, the corresponding value of p and ρ should be in N/m^2 and kg/m^3. The mass rate of flow will be in kg/sec.

Value of $\frac{p}{\rho}$ in Bernoulli's Equation*. (*i*) If the value of p is taken in N/m^2, the corresponding value of ρ is in kg/m^3. And as mentioned above (point number 2), the value of R should be 287 J/kg K.

(*ii*) If the value of p is taken in kgf/m^2 in Bernoulli's equation, the corresponding value of ρ should be in ms1/m^3. But as mentioned in point number 1, if the value of R is taken 29.2, the corresponding values of p and ρ are in kgf/m^2 and kg/m^3. Hence the mass density in equation of state is in kg/m^3 while in Bernoulli's equation it is in ms1/m^3. The density calculated from equation of state must be converted into ms1/m^3.

Note. It is better to use pressure in N/m^2, density in kg/m^3 and value of R = 287 J/kg K. The value of density calculated from equation of state will be in the same dimensions as used in Bernoulli's equation.

15.2.2 Expansion and Compression of Perfect Gas.

When the expansion or compression of a perfect gas takes place, the pressure, temperature and density are changed. The change in pressure, temperature and density of a gas is brought about by the two processes which are known as

1. Isothermal process, and 2. Adiabatic process.

1. Isothermal Process. This is the process in which a gas is compressed or expanded while the temperature is kept constant. The gas obeys Boyle's law, according to which we have

$$p\forall = \text{Constant, where } \forall = \text{Specific volume}$$

or
$$\frac{p}{\rho} = \text{Constant} \qquad \left(\because \ \forall = \frac{1}{\rho}\right) \qquad ...(15.3)$$

2. Adiabatic Process. If the compression or expansion of a gas takes place in such a way that the gas neither gives heat, nor takes heat from its surrounding, then the process is said to be adiabatic. According to this process,

$$p\forall^k = \text{Constant}$$

where k = Ratio of the specific heat at constant pressure to the specific heat at constant volume

$$= \frac{C_p}{C_v} = 1.4 \text{ for air.}$$

The above relation is also written as $\frac{p}{\rho^k} = \text{Constant.}$ ...(15.4)

If the adiabatic process is reversible (or frictionless), it is known as isentropic process. And if the pressure and density are related in such a way that k is not equal to $\frac{C_p}{C_v}$ but equal to some positive value then the process is known as polytropic. According to which

$$\frac{p}{\rho^n} = \text{Constant} \qquad ...(15.5)$$

where $n \neq k$ but equal to some positive constant.

* Please, refer to equations (15.10) and (15.11).

▶ 15.3 BASIC EQUATIONS OF COMPRESSIBLE FLOW

The basic equations of the compressible flows are

1. Continuity Equation,
2. Bernoulli's Equation or Energy Equation,
3. Momentum Equation,
4. Equation of state.

15.3.1 Continuity Equation. This is based on law of conservation of mass which states that matter cannot be created nor destroyed. Or in other words, the matter or mass is constant. For one-dimensional steady flow, the mass per second = ρAV

where ρ = Mass density, A = Area of cross-section, V = Velocity

As mass or mass per second is constant according to law of conservation of mass. Hence

$$\rho AV = \text{Constant.} \qquad \ldots(15.6)$$

Differentiating equation (15.6), $d(\rho AV) = 0$ or $\rho d(AV) + AVd\rho = 0$

or $\rho[AdV + VdA] + AVd\rho = 0$ or $\rho AdV + \rho VdA + AVd\rho = 0$

Dividing by ρAV, we get $$\frac{dV}{V} + \frac{dA}{A} + \frac{d\rho}{\rho} = 0. \qquad \ldots(15.7)$$

Equation (15.7) is also known as continuity equation in differential form.

15.3.2 Bernoulli's Equation. Bernoulli's equation has been derived for incompressible fluids in Chapter 6. The same procedure is followed. The flow of a fluid particle along a stream-line in the direction of S is considered. The resultant force on the fluid particle in the direction of S is equated to the mass of the fluid particle and its acceleration. As the flow of compressible fluid is steady, the same Euler's equation as given by equation (6.3) is obtained as

$$\frac{dp}{\rho} + VdV + gdZ = 0 \qquad \ldots(15.8)$$

Integrating the above equation, we get

$$\int \frac{dp}{\rho} + \int VdV + \int gdZ = \text{Constant}$$

or $$\int \frac{dp}{\rho} + \frac{V^2}{2} + gZ = \text{Constant} \qquad \ldots(15.9)$$

In case of incompressible flow, the density ρ is constant and hence integration of $\frac{dp}{\rho}$ is equal to $\frac{p}{\rho}$.

But in case of compressible flow, the density ρ is not constant. Hence ρ cannot be taken outside the integration sign. With the change of ρ, the pressure p also changes for compressible fluids. This change of ρ and p takes place according to equations (15.3) or (15.4) depending upon the type of process during compressible flow. The value of ρ from these equations in terms of p is obtained and is substituted in $\int \frac{dp}{\rho}$ and then the integration is done. The Bernoulli's equation will be different for isothermal process and for adiabatic process.

(A) Bernoulli's Equation for Isothermal Process. For isothermal process, the relation between pressure (p) and density (ρ) is given by equation (15.3) as

$$\frac{p}{\rho} = \text{Constant} = C_1 \text{ (say)} \qquad ...(i)$$

$$\therefore \qquad \rho = \frac{p}{C_1}$$

Hence
$$\int \frac{dp}{\rho} = \int \frac{dp}{p / C_1} = \int \frac{C_1 dp}{p} = C_1 \int \frac{dp}{p} \qquad (\because C_1 \text{ is constant})$$

$$= C_1 \log_e p = \frac{p}{\rho} \log_e p \qquad \left(\because C_1 = \frac{p}{\rho} \text{ from equation } (i)\right)$$

Substituting the value $\int \frac{dp}{\rho}$ in equation (15.9), we get

$$\frac{p}{\rho} \log_e p + \frac{V^2}{2} + gZ = \text{Constant}$$

Dividing by 'g', $\quad \frac{p}{\rho g} \log_e p + \frac{V^2}{2g} + Z = \text{Constant.} \qquad ...(15.10)$

Equation (15.10) is the Bernoulli's equation for compressible flow undergoing isothermal process. For the two points 1 and 2, this equation is written as

$$\frac{p_1}{\rho_1 g} \log_e p_1 + \frac{V_1^2}{2g} + Z_1 = \frac{p_2}{\rho_2 g} \log_e p_2 + \frac{V_2^2}{2g} + Z_2 \qquad ...(15.11)$$

(B) Bernoulli's Equation for Adiabatic Process. For the adiabatic process, the relation between pressure (p) and density (ρ) is given by equation (15.4) as

$$\frac{p}{\rho^k} = \text{Constant} = \text{say } C_2 \qquad ...(ii)$$

$$\therefore \qquad \rho^k = \frac{p}{C_2} \quad \text{or} \quad \rho = \left(\frac{p}{C_2}\right)^{1/k}$$

Hence
$$\int \frac{dp}{\rho} = \int \frac{dp}{\left(\frac{p}{C_2}\right)^{1/k}} = \int \frac{C_2^{1/k}}{p^{1/k}} \, dp = C_2^{1/k} \int \frac{1}{p^{1/k}} dp$$

$$= C_2^{1/k} \int p^{-1/k} \, dp = C_2^{1/k} \frac{p^{\left(-\frac{1}{k}+1\right)}}{\left(-\frac{1}{k}+1\right)}$$

$$= \frac{C_2^{1/k} \, p^{\left(\frac{k-1}{k}\right)}}{\left(\frac{k-1}{k}\right)} = \left(\frac{k}{k-1}\right) C_2^{1/k} p^{\left(\frac{k-1}{k}\right)}$$

$$= \left(\frac{k}{k-1}\right)\left(\frac{p}{\rho^k}\right)^{1/k} p^{\left(\frac{1-k}{k}\right)} \qquad \left(\because\ C_2^{1/k} = \frac{p}{\rho^k} \text{ from } (ii)\right)$$

$$= \left(\frac{k}{k-1}\right)\frac{p^{1/k}}{\rho^{k\times 1/k}} p^{\left(\frac{k-1}{k}\right)} = \left(\frac{k}{k-1}\right)\frac{p^{\frac{1}{k}+\frac{k-1}{k}}}{\rho} = \left(\frac{k}{k-1}\right)\frac{p}{\rho}$$

Substituting the value of $\int \frac{dp}{\rho} = \left(\frac{k}{k-1}\right)\frac{p}{\rho}$ in equation (15.9), we get

$$\left(\frac{k}{k-1}\right)\frac{p}{\rho} + \frac{V^2}{2} + gZ = \text{Constant}$$

Dividing by 'g' $$\left(\frac{k}{k-1}\right)\frac{p}{\rho g} + \frac{V^2}{2g} + Z = \text{Constant.} \qquad ...(15.12)$$

Equation (15.12) is the Bernoulli's equation for compressible flow undergoing adiabatic process. For the two points 1 and 2, this equation is written as

$$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1 g} + \frac{V_1^2}{2g} + Z_1 = \left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2 g} + \frac{V_2^2}{2g} + Z_2 \qquad ...(15.13)$$

Problem 15.1 *A gas is flowing through a horizontal pipe at a temperature of 4°C. The diameter of the pipe is 8 cm and at a section 1-1 in this pipe, the pressure is 30.3 N/cm^2 (gauge). The diameter of the pipe changes from 8 cm to 4 cm at the section 2-2, where pressure is 20.3 N/cm^2 (gauge). Find the velocities of the gas at these sections assuming an isothermal process. Take R = 287.14 Nm/kg K, and atmospheric pressure = 10 N/cm^2.*

Solution. Given :

For the section 1-1,

Temperature, $t_1 = 4°C$

$\therefore$ Absolute temperature, $T_1 = 4 + 273 = 277°K$

Diameter pipe, $D_1 = 8 \text{ cm} = 0.08 \text{ m}$

$\therefore$ Area of pipe, $A_1 = \frac{\pi}{4} D_1^2 = \frac{\pi}{4}(.08)^2 = .005026 \text{ m}^2$

Pressure, $p_1 = 30.3 \text{ N/cm}^2$ (gauge)

$= 30.3 + 10 = 40.3 \text{ N/cm}^2$ (absolute) $= 40.3 \times 10^4 \text{ N/m}^2$ (abs.)

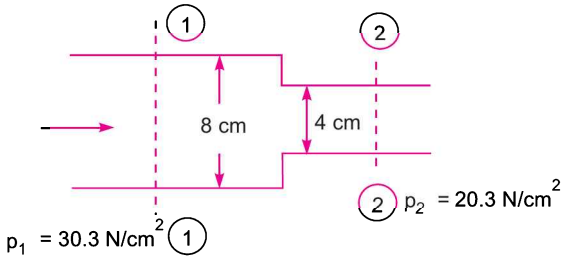


Fig. 15.1

For the section 2-2,

Diameter of pipe, $D_2 = 4 \text{ cm} = .04 \text{ m}$

$\therefore$ Area, $A_2 = \frac{\pi}{4} (.04)^2 = .0012565 \text{ m}^2$

Pressure, $p_2 = 20.3 + 10 = 30.3 \text{ N/cm}^2 \text{ (abs.)} = 30.3 \times 10^4 \text{ N/m}^2 \text{ (abs.)}$

Gas constant, $R = 287.14 \text{ N-m/kg°K}$

Ratio of specific heat, $k = 1.4$.

Applying continuity equation at sections (1) and (2), we get

$$\rho_1 A_1 V_1 = \rho_2 A_2 V_2$$

or
$$\frac{V_2}{V_1} = \frac{\rho_1 A_1}{\rho_2 A_2} = \frac{\rho_1 \times .005026}{\rho_2 \times .0012565} = 4 \times \frac{\rho_1}{\rho_2} \quad ...(i)$$

For isothermal process using equation (15.3),

$$\frac{p_1}{\rho_1} = \frac{p_2}{\rho_2} \text{ or } \frac{\rho_1}{\rho_2} = \frac{p_1}{p_2} = \frac{40.3 \times 10^4}{30.3 \times 10^4} = 1.33$$

Substituting the value of $\frac{\rho_1}{\rho_2} = 1.33$ in equation (i), we get

$$\frac{V_2}{V_1} = 4 \times 1.33 = 5.32$$

$\therefore$
$$V_2 = 5.32 \; V_1 \quad ...(ii)$$

Applying Bernoulli's equation at sections 1-1 and 2-2 for isothermal process which is given by equation (15.11), we get

$$\frac{p_1}{\rho_1 g} \log_e p_1 + \frac{V_1^2}{2g} + Z_1 = \frac{p_2}{\rho_2 g} \log_e p_2 + \frac{V_2^2}{2g} + Z_2$$

For horizontal pipe, $Z_1 = Z_2$

$\therefore$
$$\frac{p_1}{\rho_1 g} \log_e p_1 + \frac{V_1^2}{2g} = \frac{p_2}{\rho_2 g} \log_e p_2 + \frac{V_2^2}{2g}$$

or
$$\frac{p_1}{\rho_1 g} \log_e p_1 - \frac{p_2}{\rho_2 g} \log_e p_2 = \frac{V_2^2}{2g} - \frac{V_1^2}{2g}$$

But for isothermal process, $\frac{p_1}{\rho_1} = \frac{p_2}{\rho_2}$

$\therefore$
$$\frac{p_1}{\rho_1 g} \log_e p_1 - \frac{p_1}{\rho_1 g} \log_e p_2 = \frac{V_2^2}{2g} - \frac{V_1^2}{2g}$$

or
$$\frac{p_1}{\rho_1 g} \left[\log_e \frac{p_1}{p_2} \right] = \frac{(5.32 \; V_1)^2}{2g} - \frac{V_1^2}{2g} \quad (\because \text{ From } (ii),\ V_2 = 5.32 \; V_1)$$

or $$\frac{p_1}{\rho_1 g} \log_e \left(\frac{40.3 \times 10^4}{30.3 \times 10^4} \right) = \frac{V_1^2}{2g} (5.32^2 - 1) = 27.30 \frac{V_1^2}{2g}$$

or $$\frac{p_1}{\rho_1 g} \log_e 1.33 = 27.30 \frac{V_1^2}{2g}$$

or $$\frac{p_1}{\rho_1 g} \times 0.285 = 27.30 \frac{V_1^2}{2g}$$

or $$\frac{p_1}{\rho_1} = \frac{27.30}{2 \times .285} V_1^2 = 47.894 V_1^2 \quad ...(iii)$$

Now from equation of state, *i.e.*, from equation (15.2), we have

$$\frac{p}{\rho} = RT \text{ or at section 1, } \frac{p_1}{\rho_1} = RT_1$$

or $$\frac{p_1}{\rho_1} = RT_1 = 287.14 \times 277 = 79537.4$$

Substituting this value of $\frac{p_1}{\rho_1} = 79537.4$ in equation (*iii*), we get $79537.4 = 47.894 V_1^2$

$\therefore$ $$V_1 = \sqrt{\frac{79537.4}{47.894}} = \mathbf{40.75 \text{ m/s. Ans.}}$$

From equation (*ii*), $V_2 = 5.32 \times V_1 = 5.32 \times 40.75 = \mathbf{216.79 \text{ m/s. Ans.}}$

Problem 15.2 *A gas is flowing through a horizontal pipe which is having area of cross-section as 40 cm^2, where pressure is 40 N/cm^2 (gauge) and temperature is 15°C. At another section the area of cross-section is 20 cm^2 and pressure is 30 N/cm^2 (gauge). If the mass rate of flow of gas through the pipe is 0.5 kg/s, find the velocities of the gas at these sections, assuming an isothermal change. Take R = 292 N-m/kg°K, and atmospheric pressure = 10 N/cm^2.*

Solution. Given :

	Section 1	Section 2
Area,	$A_1 = 40 \text{ cm}^2 = 40 \times 10^{-4} \text{ m}^2$	$A_2 = 20 \text{ cm}^2 = 20 \times 10^{-4} \text{ m}^2$
Pressure,	$p_1 = 40 \text{ N/cm}^2$ (gauge)	$p_2 = 30 \text{ N/cm}^2$ (gauge)
	$= 40 + 10 = 50 \text{ N/cm}^2$ (abs.)	$= 30 + 10 = 40 \text{ N/cm}^2$ (abs.)
	$= 50 \times 10^4 \text{ N/m}^2$	$= 40 \times 10^4 \text{ N/m}^2$
Temperature,	$t_1 = 15°\text{C}$	
$\therefore$	$T_1 = 15 + 273 = 288°\text{K}$	

Mass rate of flow $= 0.5$ kg/s.

Gas constant, $R = 292$ N-m/kg°K

From equation of state, *i.e.*, equation (15.2), $\frac{p_1}{\rho_1} = RT_1$

$$\therefore \quad \rho_1 = \frac{p_1}{RT_1} = \frac{50 \times 10^4}{292 \times 288} \frac{\text{kg}}{\text{m}^3} = 5.945 \frac{\text{kg}}{\text{m}^3}$$

Mass rate of flow is given by $= \rho_1 A_1 V_1$ or $0.5 = 5.945 \times 40 \times 10^{-4} \times V_1$

$$\therefore \quad V_1 = \frac{0.5}{5.945 \times 40 \times 10^{-4}} = 21.02 \text{ m/s}$$

For isothermal process, temperature is constant and hence temperature at section 2 is also 288°K.

$$\therefore \quad T_2 = 288°\text{K}$$

Using equation (15.2), we get $\frac{p_2}{\rho_2} = RT_2$

$$\therefore \quad \rho_2 = \frac{p_2}{RT_2} = \frac{40 \times 10^4}{292 \times 288} = 4.756 \text{ kg/m}^3$$

Now mass rate of flow $= \rho_2 A_2 V_2$

$$\therefore \quad 0.5 = 4.756 \times 20 \times 10^{-4} \times V_2$$

$$\therefore \quad V_2 = \frac{0.5}{4.756 \times 20 \times 10^{-4}} = \textbf{52.565 m/s. Ans.}$$

Problem 15.3 *A gas with a velocity of 300 m/s is flowing through a horizontal pipe at a section where pressure is 6×10^4 N/m^2 (absolute) and temperature 40°C. The pipe changes in diameter and at this section the pressure is 9×10^4 N/m^2. Find the velocity of the gas at this section if the flow of the gas is adiabatic.*

Take R = 287 J/kg°K and k = 1.4.

Solution. Given :

Section 1	**Section 2**
$V_1 = 300$ m/s	$p_2 = 9 \times 10^4$ N/m^2
$p_1 = 6 \times 10^4$ N/m^2	V_2 = velocity at section 2
$t_1 = 40°\text{C}$	$R = 287$ J/kg°K
$\therefore \quad T_1 = 273 + 40 = 313°\text{K}$	

Adiabatic flow, $k = 1.4$

Applying Bernoulli's equation at sections 1 and 2, given by equation (15.13), we get

$$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1 g} + \frac{V_1^2}{2g} = \left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2 g} + \frac{V_2^2}{2g} \qquad (\because \quad Z_1 = Z_2)$$

or $$\left(\frac{k}{k-1}\right)\left[\frac{p_1}{\rho_1 g} - \frac{p_2}{\rho_2 g}\right] = \frac{V_2^2}{2g} - \frac{V_1^2}{2g}$$

Dividing by 'g', we get $\left(\frac{k}{k-1}\right)\left[\frac{p_1}{\rho_1} - \frac{p_2}{\rho_2}\right] = \frac{V_2^2}{2} - \frac{V_1^2}{2}$

or $$\left(\frac{1.4}{1.4-1.0}\right)\frac{p_1}{\rho_1}\left[1-\frac{p_2}{\rho_2}\times\frac{\rho_1}{p_1}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2} \qquad ...(i)$$

For adiabatic flow, using equation (15.4), we get

$$\frac{p_1}{\rho_1^k}=\frac{p_2}{\rho_2^k} \quad \text{or} \quad \frac{p_1}{p_2}=\left(\frac{\rho_1}{\rho_2}\right)^k \text{ or } \left(\frac{\rho_1}{\rho_2}\right)=\left(\frac{p_1}{p_2}\right)^{1/k}$$

Substituting the value of $\frac{\rho_1}{\rho_2}$ in equation (i), we get

$$\frac{1.4}{0.4}\frac{p_1}{\rho_1}\left[1-\frac{p_2}{p_1}\times\left(\frac{p_1}{p_2}\right)^{1/k}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2}$$

or $$3.5\,\frac{p_1}{\rho_1}\left[1-\left(\frac{p_2}{p_1}\right)\times\left(\frac{p_2}{p_1}\right)^{-1/k}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2}$$

or $$3.5\,\frac{p_1}{\rho_1}\left[1-\left(\frac{p_2}{p_1}\right)^{1-\frac{1}{k}}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2} \text{ or } 3.5\,\frac{p_1}{\rho_1}\left[1-\left(\frac{p_2}{p_1}\right)^{\frac{k-1}{k}}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2}$$

Substituting the value of p_2 and p_1, we get

$$3.5\,\frac{p_1}{\rho_1}\left[1-\left(\frac{9\times10^4}{6\times10^4}\right)^{\frac{1.4-1}{1.4}}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2}$$

or $$3.5\,\frac{p_1}{\rho_1}\left[1-1.5^{2/7}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2} \text{ or } 3.5\,\frac{p_1}{\rho_1}\,[1-1.5^{.2857}]=\frac{V_2^2}{2}-\frac{V_1^2}{2}$$

or $$-0.4298\,\frac{p_1}{\rho_1}=\frac{V_2^2}{2}-\frac{V_1^2}{2}=\frac{V_2^2}{2}-\frac{300^2}{2}$$

or $$-.4298\,\frac{p_1}{\rho_1}=\frac{V_2^2}{2}-45000 \qquad ...(ii)$$

From equation of state, we have $\frac{p_1}{\rho_1}=RT_1=287\times313=89831$

Substituting the value of $\frac{p_1}{\rho_1}=89831$ in equation (ii),

$$-0.4298\times89831=\frac{V_2^2}{2}-45000$$

or $$\frac{V_2^2}{2}=45000-.4298\times89831=6390.6$$

$\therefore$ $$V_2=\sqrt{2\times6390.6}=\mathbf{113.0\ m/s.\ Ans.}$$

15.3.3 Momentum Equations. The momentum per second of a flowing fluid (or momentum flux) is equal to the product of mass per second and the velocity of the flow. Mathematically, the momentum per second of a flowing fluid (compressible or incompressible) is

$$= \rho AV \times V, \quad \text{where } \rho AV = \text{Mass per second.}$$

The term ρAV is constant at every section of flow due to continuity equation. This means the momentum per second at any section is equal to the product of a constant quantity and the velocity. This also implies that momentum per second is independent of compressible effect. Hence the momentum equation for incompressible and compressible fluid is the same. The momentum equation for compressible fluid for any direction may be expressed as,

Net force in the direction of S = Rate of change of momentum in the direction of S

$$= \text{Mass per second [change of velocity]}$$

$$= \rho AV[V_2 - V_1] \qquad ...(15.14)$$

where V_2 = Final velocity in the direction of S,

V_1 = Initial velocity in the direction of S.

▶15.4 VELOCITY OF SOUND OR PRESSURE WAVE IN A FLUID

The disturbance in a solid, liquid or gas is transmitted from one point to the other. The velocity with which the disturbance is transmitted depends upon the distance between the molecules of the medium. In case of solids, molecules are closely packed and hence the disturbance is transmitted instantaneously. In case of liquids and gases (or fluids) the molecules are relatively apart. The disturbance will be transmitted from one molecule to the next molecule. But in case of fluids, there is some distance between two adjacent molecules. Hence each molecule will have to travel a certain distance before it can transmit the disturbance. Thus the velocity of disturbance in case of fluids will be less than the velocity of the disturbance in solids.

The distance between the molecules is related with the density, which in turn depends upon pressure in case of fluids. Hence the velocity of disturbance depends upon the changes of pressure and density of the fluid.

15.4.1 Expression for Velocity of Sound Wave in a Fluid. The disturbance creates the pressure waves in a fluid. These pressure waves travel with a velocity of sound waves in all directions. But for the sake of simplicity, one-dimensional case will be considered.

Fig. 15.2 shows the model for one-dimensional propagation of the pressure waves. It is a right long pipe of uniform cross-sectional area, fitted with a piston. Let the pipe is filled with a compressible fluid, which is at rest initially. The piston is moved towards right and a disturbance is created in the fluid. This disturbance is in the form of pressure wave, which travels in the fluid with a velocity of sound wave.

Let A = Cross-sectional area of the pipe

V = Velocity of piston

p = Pressure of the fluid in pipe before the movement of the piston

ρ = Density of fluid before the movement of the piston

dt = A small interval of time with which piston is moved

C = Velocity of pressure wave or sound wave travelling in the fluid

Distance travelled by the piston in time dt

$$= \text{Velocity of piston} \times dt = V \times dt$$

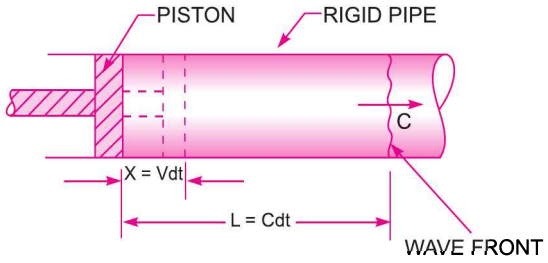


Fig. 15.2 *Propagation of pressure wave.*

Distance travelled by the pressure wave in time dt

$$= \text{Velocity of pressure wave} \times dt = C \times dt$$

As the value of C will be very large, hence $C \times dt$ will be more than $V \times dt$. For the time interval 'dt', the pressure wave has travelled a distance L and piston has moved through x. Thus in the length of the tube equal to $(L - x)$, the fluid will be compressed. Due to compression of the fluid, the pressure and density of the fluid will change.

Let $p + dp$ = Pressure after compression

$\rho + d\rho$ = Density after compression or the density of fluid in the length $(L - x)$

Now mass of fluid for a length 'L' before compression

$$= \rho \times \text{Volume of fluid upto length } L$$

$$= \rho \times A \times L = \rho \times A \times C \times dt \qquad (\because \quad L = Cdt) \quad ...(i)$$

Mass of fluid after compression for length $(L - x)$

$$= \text{Density after compression} \times \text{Area} \times \text{Length}$$

$$= (\rho + d\rho)\, A \times (L - x)$$

$$= (\rho + d\rho)\, A \times (Cdt - Vdt) \qquad (\because \quad L = Cdt,\ x = Vdt) \quad ...(ii)$$

From the continuity equation, we have

Mass of fluid before compression

$$= \text{Mass of fluid after compression}$$

$$\therefore \qquad \rho ACdt = (\rho + d\rho)\, A \times (Cdt - Vdt) \text{ or } \rho ACdt = (\rho + d\rho)\, A \times dt\,(C - V)$$

Dividing by $A \times dt$, $\quad \rho C = (\rho + d\rho)\,(C - V) = \rho C - \rho V + Cd\rho - Vd\rho$

$$\therefore \qquad Cd\rho = \rho C - \rho C + \rho V + Vd\rho = \rho V + Vd\rho. \qquad ...(iii)$$

But the velocity of the piston, V, is very small as compared to the velocity of the pressure wave C. Also the value of $d\rho$ is very small. Hence the term $(V \times d\rho)$ will be very-very small and can be neglected. Hence equation (iii) becomes,

$$Cd\rho = \rho V \qquad ...(iv)$$

Now when the piston is moved with a velocity V for time dt, the fluid which is at rest initially will move with a velocity equal to the velocity of the piston. Also the pressure of the fluid will increase from p to $p + dp$ due to the movement of the piston. Hence applying the impulse momentum equation, we get

Net force on the fluid = Rate of change of momentum

or $\qquad (p + dp)\, A - p \times A = \text{Mass per second} \qquad$ [change of velocity of fluid]

or $$dp \times A = \frac{\text{Total mass}}{\text{Time}} [V - 0] = \frac{\rho AL}{dt} [V - 0]$$

$$= \frac{\rho ACdt}{dt} [V - 0] \qquad (\because \quad L = Cdt)$$

$$= \rho AC [V - 0] = \rho ACV \quad \text{or} \quad dp = \frac{\rho ACV}{A} = \rho CV$$

or $$C = \frac{dp}{\rho V} \qquad ...(v)$$

Multiplying equations (*iv*) and (*v*), we get

$$C^2 d\rho = \rho V \times \frac{dp}{\rho V} = dp$$

$$C^2 = \frac{dp}{d\rho}$$

$\therefore$ $$C = \sqrt{\frac{dp}{d\rho}} \qquad ...(15.15)$$

Hence equation (15.15) gives the velocity of sound wave which is the square root of the ratio of change of pressure to the change of density of a fluid due to disturbance.

15.4.2 Velocity of Sound in Terms of Bulk Modulus. Bulk modulus K is defined as

$$K = \frac{\text{Increase in pressure}}{\frac{\text{Decrease in volume}}{\text{Original volume}}} = \frac{dp}{-\left(\frac{d\forall}{\forall}\right)} \qquad ...(vi)$$

where $d\forall$ = Decrease in volume, $\forall$ = Original volume.

Negative sign is taken, as with the increase of pressure, volume decreases.

Now we know mass of a fluid is constant. Hence

$$\rho \times \text{Volume} = \text{Constant} \qquad (\because \quad \text{Mass} = \rho \times \text{Volume})$$

or $$\rho \times \forall = \text{Constant}$$

Differentiating the above equation (ρ and $\forall$ are variables),

$$\rho d\forall + \forall d\rho = 0 \quad \text{or} \quad \rho d\forall = - \forall d\rho \text{ or } - \frac{d\forall}{\forall} = \frac{d\rho}{\rho}$$

Substituting the value $\left(-\frac{d\forall}{\forall}\right)$ in equation (*vi*), we get

$$K = \frac{dp}{\frac{d\rho}{\rho}} = \rho \frac{dp}{d\rho} \text{ or } \frac{dp}{d\rho} = \frac{K}{\rho}$$

From equation (15.15), the velocity of sound wave is

$$C = \sqrt{\frac{dp}{d\rho}} = \sqrt{\frac{K}{\rho}} \qquad ...(15.16)$$

Equation (15.16) gives the velocity of sound wave in terms of bulk modulus and density. This equation is applicable for liquids and gases.

15.4.3 Velocity of Sound for Isothermal Process. Isothermal process is given by equation (15.3), as

$$\frac{p}{\rho} = \text{Constant or } p\rho^{-1} = \text{Constant}$$

Differentiating the above equation (p and ρ both are variable),

$$p\,(-1)\,\rho^{-2}\,d\rho + \rho^{-1}\,dp = 0$$

Dividing by ρ^{-1}, we get $-p\rho^{-1}\,d\rho + dp = 0$ or $\frac{-p}{\rho}\,d\rho + dp = 0$

$$\therefore \qquad dp = \frac{p}{\rho}\,d\rho \text{ or } \frac{dp}{d\rho} = \frac{p}{\rho} = RT \qquad \left(\because \text{ From equation of state } \frac{p}{\rho} = RT\right)$$

Substituting the value of $\frac{dp}{d\rho}$ in equation (15.15), $C = \sqrt{\frac{p}{\rho}} = \sqrt{RT}$ ...(15.17)

15.4.4 Velocity of Sound for Adiabatic Process. Adiabatic process is given by equation (15.4), as

$$\frac{p}{\rho^k} = \text{Constant or } p\rho^{-k} = \text{Constant}$$

Differentiating the above equation, we get

$$p\,(-k)\,\rho^{-k-1}\,d\rho + \rho^{-k}\,dp = 0$$

Dividing by ρ^{-k}, we get $-pk\rho^{-1}\,d\rho + dp = 0$ or $dp = \frac{pk}{\rho}\,d\rho$

$$\therefore \qquad \frac{dp}{d\rho} = \frac{p}{\rho}\,k = RTk \qquad \left(\because \frac{p}{\rho} = RT\right)$$

$$= kRT$$

Substituting the value of $\frac{dp}{d\rho}$ in equation (15.15), we get $C = \sqrt{kRT}$. ...(15.18)

Note 1. For the propagation of the minor disturbances through air, the process is assumed to be adiabatic. The velocity of the disturbances (pressure waves) through air is very high and hence there is no time for any appreciable heat transfer.

2. Isothermal process is considered for the calculation of the velocity of the sound waves (or pressure waves) only when it is given in the numerical problem that process is isothermal. If no process is mentioned, it is assumed to be adiabatic.

▶15.5 MACH NUMBER

In Chapter 12, Art. 12.8.5, Mach number has been defined as the square root of the ratio of the inertia force of a flowing fluid to the elastic force.

$\therefore$ Mach number $= M = \sqrt{\dfrac{\text{Inertia force}}{\text{Elastic force}}} = \sqrt{\dfrac{\rho A V^2}{KA}}$

$$= \sqrt{\frac{V^2}{K/\rho}} = \frac{V}{\sqrt{K/\rho}} = \frac{V}{C} \qquad \left[\because \sqrt{\frac{K}{\rho}} = C \text{ from equation (15.16)}\right]$$

Thus Mach number $= M$

$$= \frac{\text{Velocity of fluid or body moving in fluid}}{\text{Velocity of sound in the fluid}}$$

$$= \frac{V}{C}. \qquad \ldots(15.19)$$

For the compressible fluid flow, Mach number is an important non-dimensional parameter. On the basis of the Mach number, the flow is defined as :

1. Sub-sonic flow, 2. Sonic flow, and 3. Super-sonic flow.

1. Sub-sonic Flow. A flow is said sub-sonic flow if the Mach number is less than 1.0 (or $M < 1$) which means the velocity of flow is less than the velocity of sound wave (or $V < C$).

2. Sonic Flow. A flow is said to be sonic flow if the Mach number (M) is equal to 1.0. This means that when the velocity of flow V is equal to the velocity of sound C, the flow is said to be sonic flow.

3. Super-sonic Flow. A flow is said to be super-sonic flow if the Mach number is greater than 1.0 (or $M > 1$). This means that when velocity of flow V is greater than the velocity of sound wave, the flow is said to be super-sonic flow.

Problem 15.4 *Find the sonic velocity for the following fluids :*

(i) Crude oil of sp. gr. 0.8 and bulk modulus 153036 N/cm^2.

(ii) Mercury having a bulk modulus of 2648700 N/cm^2.

Solution. Given :

(*i*) For Crude oil, sp. gr. = 0.8

$\therefore$ Density, $\rho = 0.8 \times 1000 = 800$ kg/m^3

Bulk modulus, $K = 153036$ N/cm^2 $= 153036 \times 10^4$ N/m^2

Using equation (15.16) for sonic velocity, as

$$C = \sqrt{\frac{K}{\rho}} = \sqrt{\frac{153036 \times 10^4}{800}} = 1383.09 \simeq \textbf{1383 m/s. Ans.}$$

(*ii*) For Mercury, sp. gr. = 13.6

$\therefore$ Density of Mercury, $\rho = 13.6 \times 1000$ kg/m^3

Bulk modulus, $K = 2648700$ N/cm^2 $= 2648700 \times 10^4$ N/m^2

The sonic velocity, $C = \sqrt{\dfrac{K}{\rho}} = \sqrt{\dfrac{2648700 \times 10^4}{13.6 \times 1000}} = \textbf{1395.55 m/s. Ans.}$

Problem 15.5 *Find the speed of the sound wave in air at sea-level where the pressure and temperature are 10.1043 N/cm^2 (abs.) and 15°C respectively. Take R = 287 J/kg°K and k = 1.4.*

Solution. Given :

Pressure, $p = 10.1043 \text{ N/cm}^2 = 10.1043 \times 10^4 \text{ N/m}^2$

Temperature, $t = 15°\text{C}$

$\therefore$ $T = 273 + 15 = 288 \text{ K}$, $R = 287 \text{ J/kg°K}$, $k = 1.4$

For adiabatic process, the velocity of sound is given by equation (15.18), as

$$C = \sqrt{kRT} = \sqrt{1.4 \times 287 \times 288} = \mathbf{340.17 \text{ m/s. Ans.}}$$

Problem 15.6 *Calculate the Mach number at a point on a jet propelled aircraft, which is flying at 1100 km/hour at sea-level where air temperature is 20°C. Take k = 1.4 and R = 287 J/kg°K.*

Solution. Given :

Speed of aircraft, $V = 1100 \text{ km/hour} = \dfrac{1100 \times 1000}{60 \times 60} = 305.55 \text{ m/s}$

Temperature, $t = 20°\text{C}$

$\therefore$ $T = 273 + 20 = 293°\text{K}$, $k = 1.4$, $R = 287 \text{ J/kg°K}$

Using equation (15.18), the velocity of sound is

$$C = \sqrt{kRT} = \sqrt{1.4 \times 287 \times 293} = 343.11 \text{ m/s}$$

Mach number is given by equation (15.19) as

$$M = \frac{V}{C} = \frac{305.55}{343.11} = \mathbf{0.89. \text{ Ans.}}$$

Problem 15.7 *An aeroplane is flying at an height of 15 km where the temperature is – 50°C. The speed of the plane is corresponding to M = 2.0. Assuming k = 1.4 and R = 287 J/kg°K, find the speed of the plane.*

Solution. Given :

Height of the plane, $Z = 15 \text{ km}$ (Extra Data)

Temperature, $t = -50°\text{C}$

$\therefore$ $T = -50 + 273 = 223°\text{K}$

Mach number, $M = 2.0$, $k = 1.4$, $R = 287 \text{ J/kg°K}$.

Using equation (15.18), we get the velocity of sound as

$$C = \sqrt{kRT} = \sqrt{1.4 \times 287 \times 223} = 299.33 \text{ m/s}$$

Using equation (15.19), we have $M = \dfrac{V}{C}$ or $2.0 = \dfrac{V}{299.33}$

$\therefore$ $V = 2.0 \times 299.33 = 598.66 \text{ m/s}$

$$= \frac{598.66 \times 60 \times 60}{1000} = \mathbf{2155.17 \text{ km/hour. Ans.}}$$

▶15.6 PROPAGATION OF PRESSURE WAVES (OR DISTURBANCES) IN A COMPRESSIBLE FLUID

Whenever any disturbance is produced in a compressible fluid, the disturbance is propagated in all directions with a velocity of sound (*i.e.*, equal to C). The nature of propagation of the disturbance depends upon the Mach number. Let us consider a small projectile moving from left to right in a straight line in a stationary fluid. Due to the movement of the projectile, the disturbances will be created in the fluid. This disturbance will be moving in all directions with a velocity C.

Hence let V = Velocity of the projectile,

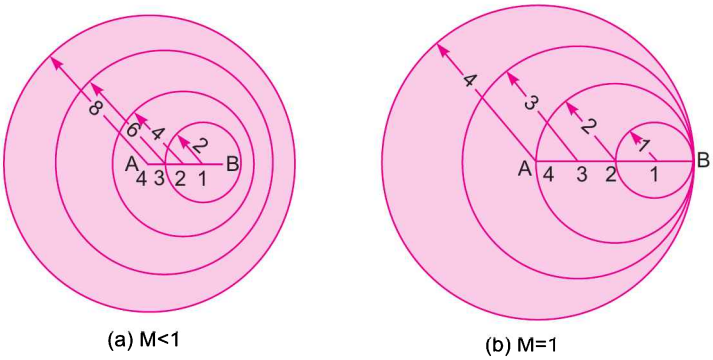
C = Velocity of pressure wave or disturbance created in the fluid.

Let us find the nature of propagation of the disturbance for different Mach numbers.

1st Case : When M < 1. When Mach number is less than 1.0, the flow is called sub-sonic flow. For $M < 1$ means $\frac{V}{C} < 1$ or $V < C$. To find the nature of propagation for this case, let $V = 1$ unit and $C = 2$ unit, so that $\frac{V}{C} = \frac{1}{2}$ which is less than 1.0. Let the projectile is at A and is moving towards right. Let in 4 seconds the projectile reaches to the position B. At A, the point 4 is also marked. The position of the projectile after 1 sec, 2 sec, 3 sec and 4 sec along the lines are shown by the points 3, 2, 1 and B respectively. The projectile moves from A to B in 4 seconds and hence the distance $AB = 4 \times V = 4 \times 1 = 4$ units. The disturbance created at A in 4 seconds will move a distance $= 4C = 4 \times 2 = 8$ units in all directions. Hence taking A as centre and radius equal to 8 units, a circle is drawn. This circle gives the position of disturbance after 4 seconds. When the projectile is at point 3, it will reach B in three seconds and distance $3B = 3 \times V = 3 \times 1 = 3$ units. But the disturbance created at point 3 in three seconds will move a distance having a radius $= 3 \times C = 3 \times 2 = 6$ units. Similarly at point 2, the disturbance will have a radius $= 2 \times C = 4$ units and at point 1, the disturbance will have a radius $= 1 \times C = 1 \times 2 = 2$ units. This is shown in Fig. 15.3 (*a*).

As in this case $V < C$, the pressure wave is always ahead of the projectile and point B is inside the sphere of radius 8 units.

2nd Case : When M = 1. When $M = 1$, the flow is known as sonic flow. In this case, the disturbance always travels with the projectile as shown in Fig. 15.3 (*b*). Let $V = 1$ unit, and $C = 1$ unit so that $M = \frac{V}{C} = \frac{1}{1} = 1.0$. Let the projectile moves from A to B in 4 seconds. The disturbance created at A in 4 seconds will move a distance having radius $= 4 \times C = 4 \times 1 = 4$ units in all directions. The projectile



(a) M<1 (b) M=1

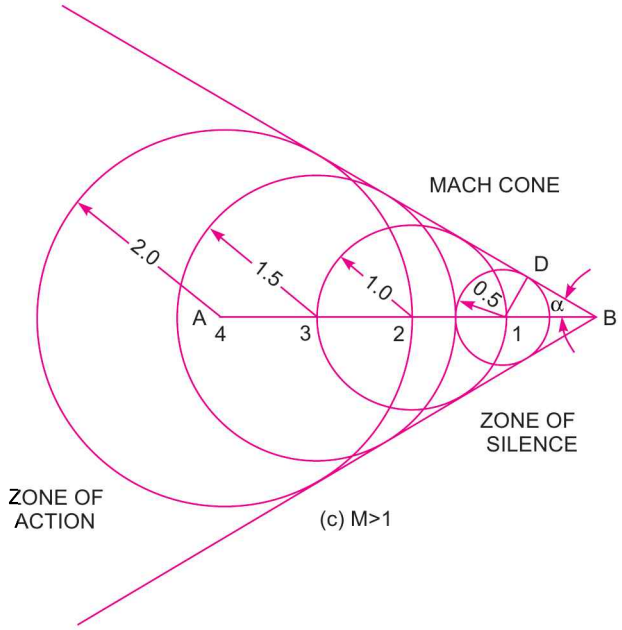


Fig. 15.3 *Propagation of disturbance for different Mach numbers.*

from point 3 will move to position B in three seconds. The disturbance created at point 3, will move a distance having radius = $3 \times C = 3 \times 1 = 3$ units in all directions in three seconds. Similarly at the point 2 and point 1, the disturbance created at these points will move a distance having radius 2 and 1 in all directions respectively.

3rd Case : When M > 1. When $M > 1$, the flow is known as supersonic flow. Let $V = 1$ unit and $C = 0.5$ unit so that $M = \frac{V}{C} = \frac{1}{0.5} = 2.0$, which is greater than unity. Let the projectile moves from A to B in 4 seconds. The distance travelled by the projectile in 4 seconds = $4 \times V = 4 \times 1 = 4$ units. Hence, take $AB = 4$ units. The disturbance created at A will move in all directions and in 4 seconds, the radius of disturbance will be equal to $4 \times C = 4 \times 0.5 = 2$ units. Hence taking A as centre, draw a circle with radius equal to 2 units. After one second from A, the projectile will be at point 3 and distance $A3 = V \times 1 = 1 \times 1 = 1$ unit. The projectile from point 3 will reach point B in three seconds. Hence the disturbance created at point 3 will move in all directions and in three seconds, the radius of disturbance from point 3 will be equal to $3 \times C = 3 \times 0.5 = 1.5$ units. Similarly the radius of disturbance at point 2 and 1 will be $2 \times C = 2 \times 0.5 = 1$ unit and $1 \times C = 1 \times 0.5 = 0.5$ unit respectively as shown in Fig. 15.3 (*c*). In this case the sphere of propagation of disturbance always lags behind the projectile. If we draw a tangent to the different circles which represent the propagated spherical waves on both sides, we shall get a cone with vertex at B. This cone is known as **Mach Cone.**

15.6.1 Mach Angle. This is defined as the half of the angle of the Mach cone. In Fig. 15.3 (*c*), angle α is known as Mach angle. In the $\Delta 1BD$ of Fig. 15.3 (*c*), the distance $1B$ = Velocity of projectile = V, the distance $1D$ = Velocity of sound wave = C. Hence we have

$$\sin \alpha = \frac{1D}{1B} = \frac{C}{V} = \frac{1}{V/C} = \frac{1}{M} \qquad ...(15.20)$$

15.6.2 Zone of Action. When $M > 1$, the effect of the disturbance is felt only in the region inside the Mach cone. This region is called the zone of action.

15.6.3 Zone of Silence. When $M > 1$, there is no effect of disturbance in the region outside the Mach cone. The region which is outside the Mach cone is called zone of silence.

Problem 15.8 *A projectile is travelling in air having pressure and temperature as 8.829 N/cm^2 and – 2°C. If the Mach angle is 40°, find the velocity of the projectile. Take k = 1.4 and R = 287 J/kg°K.*

Solution. Given :

Pressure of air, $p = 8.829 \text{ N/cm}^2 = 8.829 \times 10^4 \text{ N/m}^2$

Temperature of air, $t = -2°\text{C}$

$\therefore$ $T = -2 + 273 = 271°\text{K}$

Mach angle, $\alpha = 40°$, $k = 1.4$, $R = 287 \text{ J/kg°K}$

Let the velocity of projectile $= V$

Using equation (15.20), we have $\sin \alpha = \dfrac{C}{V}$ or $\sin 40° = 0.6427 = \dfrac{C}{V}$

The velocity of sound, C is given by equation (15.18) as

$$C = \sqrt{kRT} = \sqrt{1.4 \times 287 \times 271} = 329.98 \text{ m/s} \simeq 330 \text{ m/s}$$

$\therefore$ $$\sin 40° = 0.6427 = \frac{C}{V} = \frac{330}{V}$$

$\therefore$ $$V = \frac{330}{0.6427} = \mathbf{513 \text{ m/s.}} \quad \textbf{Ans.}$$

Problem 15.9 *A projectile travels in air of pressure 10.1043 N/cm^2 at 10°C at a speed of 1500 km/hour. Find the Mach number and the Mach angle. Take k = 1.4 and R = 287 J/kg°K.*

Solution. Given :

Pressure, $p = 10.1043 \text{ N/cm}^2 = 10.1043 \times 10^4 \text{ N/cm}^2$

Temperature, $t = 10°\text{C}$

$\therefore$ $T = 10 + 273 = 283°\text{K}$

Speed of projectile, $V = 1500 \text{ km/hour} = \dfrac{1500 \times 1000}{60 \times 60} \text{ m/s} = 416.67 \text{ m/s}$

$k = 1.4$, $R = 287 \text{ J/kg°K}$

For adiabatic process, the velocity of sound is given by

$$C = \sqrt{kRT} = \sqrt{1.4 \times 287 \times 283} = 337.20 \text{ m/s}$$

$\therefore$ Mach number, $$M = \frac{V}{C} = \frac{416.67}{337.20} = \mathbf{1.235.} \quad \textbf{Ans.}$$

$\therefore$ Mach angle is obtained from equation (15.20) as

$$\sin \alpha = \frac{C}{V} = \frac{1}{M} = \frac{1}{1.235} = 0.8097$$

$\therefore$ Mach angle, $\alpha = \sin^{-1} 0.8097 =$ **54.06°. Ans.**

Problem 15.10 *Find the velocity of bullet fired in standard air if the Mach angle is 30°. Take R = 287.14 J/kg°K and k = 1.4 for air. Assume temperature as 15°C.*

Solution. Given :

Mach angle $\alpha = 30°$

$R = 287.14$ J/kg°K

$k = 1.4$

Temperature, $t = 15°C$

$\therefore$ $T = 15 + 273 = 288°K$

Velocity of sound is given by equation (15.18) as

$$C = \sqrt{kRT} = \sqrt{1.4 \times 287.14 \times 288} = 340.25 \text{ m/s}$$

Using the relation, $\sin \alpha = \frac{C}{V}$ given by equation (15.20)

$$\sin 30° = \frac{340.25}{V}$$

$\therefore$ $$V = \frac{340.25}{\sin 20} = \textbf{680.50 m/s. Ans.}$$

▶ 15.7 STAGNATION PROPERTIES

When a fluid is flowing past an immersed body, and at a point on the body if the resultant velocity becomes zero, the values of pressure, temperature and density at that point are called stagnation properties. The point is called the **stagnation point.** The values of pressure, density and temperature are called stagnation pressure, stagnation density and stagnation temperature respectively. They are denoted by p_s, ρ_s and T_s respectively.

15.7.1 Expression for Stagnation Pressure (p_s). Consider a compressible fluid flowing past an immersed body under frictionless adiabatic conditions as in Fig. 15.4. Consider two points 1 and 2 on a stream-line as shown in Fig. 15.4.

Let p_1 = Pressure of compressible fluid at point 1,

V_1 = Velocity of fluid at 1, and

ρ_1 = Density of fluid at 1,

p_2, V_2, ρ_2 = Corresponding values of pressure, velocity and density at point 2.

Applying Bernoulli's equation for adiabatic flow given by equation (15.13) at point 1 and 2, we get

$$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1 g} + \frac{V_1^2}{2g} + Z_1 = \left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2 g} + \frac{V_2^2}{2g} + Z_2$$

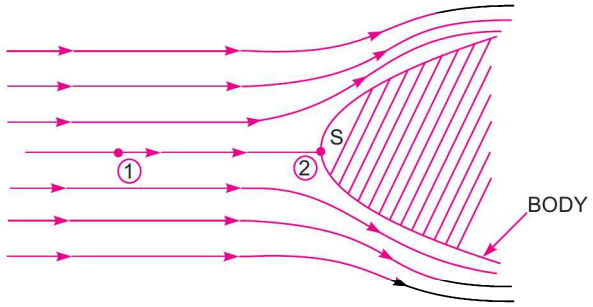


Fig. 15.4 *Stagnation properties.*

But $$Z_1 = Z_2$$

$$\therefore \quad \left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1 g} + \frac{V_1^2}{2g} = \left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2 g} + \frac{V_2^2}{2g}$$

or $$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1} + \frac{V_1^2}{2} = \left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2} + \frac{V_2^2}{2} \qquad \left(\text{Cancelling } \frac{1}{g}\right)$$

Point 2 is a stagnation point. Hence velocity will become zero at stagnation point and pressure and density will be denoted by p_s and ρ_s.

$$\therefore \quad V_2 = 0,\ p_2 = p_s \text{ and } \rho_2 = \rho_s$$

Substituting these values in the above Bernoulli's equation,

$$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1} + \frac{V_1^2}{2} = \left(\frac{k}{k-1}\right)\frac{p_s}{\rho_s} + 0$$

or $$\left(\frac{k}{k-1}\right)\left[\frac{p_1}{\rho_1} - \frac{p_s}{\rho_s}\right] = -\frac{V_1^2}{2} \text{ or } \left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1}\left[1 - \frac{p_s}{\rho_s} \times \frac{\rho_1}{p_1}\right] = -\frac{V_1^2}{2}$$

or $$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1}\left[1 - \frac{p_s}{p_1} \times \frac{\rho_1}{\rho_s}\right] = -\frac{V_1^2}{2} \qquad \ldots(i)$$

But for adiabatic process from equation (15.4), we have

$$\frac{p}{\rho^k} = \text{constant or } \frac{p_1}{\rho_1^k} = \frac{p_s}{\rho_s^k} \text{ or } \frac{p_1}{p_s} = \frac{\rho_1^k}{\rho_s^k} \text{ or } \left(\frac{\rho_1}{\rho_s}\right) = \left(\frac{p_1}{p_s}\right)^{\frac{1}{k}} \qquad \ldots(ii)$$

Substituting the value of $\frac{\rho_1}{\rho_s}$ in equation (i), we get

$$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1}\left[1 - \frac{p_s}{p_1} \times \left(\frac{p_1}{p_s}\right)^{\frac{1}{k}}\right] = -\frac{V_1^2}{2}$$

or $$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1}\left[1-\frac{p_s}{p_1}\times\left(\frac{p_s}{p_1}\right)^{-\frac{1}{k}}\right]=-\frac{V_1^2}{2}$$

or $$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1}\left[1-\left(\frac{p_s}{p_1}\right)^{1-\frac{1}{k}}\right]=-\frac{V_1^2}{2}$$

or $$\left[1-\left(\frac{p_s}{p_1}\right)^{\frac{k-1}{k}}\right]=-\frac{V_1^2}{2}\times\left(\frac{k-1}{k}\right)\frac{\rho_1}{p_1}$$

or $$1+\frac{V_1^2}{2}\left(\frac{k-1}{k}\right)\frac{\rho_1}{p_1}=\left(\frac{p_s}{p_1}\right)^{\frac{k-1}{k}} \qquad ...(iii)$$

Now for adiabatic process, the velocity of sound is given by equation (15.18) as

$$C=\sqrt{kRT}=\sqrt{k\frac{p}{\rho}} \qquad \left(\because\ \frac{p}{\rho}=RT\right)$$

For the point 1, $$C_1=\sqrt{k\frac{p_1}{\rho_1}} \text{ or } {C_1}^2=k\frac{p_1}{\rho_1}$$

Substituting the value of $\frac{k\,p_1}{\rho_1}={C_1}^2$ in equation (*iii*)

$$1+\frac{V_1^2}{2}(k-1)\times\frac{1}{C_1^2}=\left(\frac{p_s}{p_1}\right)^{\left(\frac{k-1}{k}\right)} \text{ or } 1+\frac{V_1^2}{2C_1^2}\times(k-1)=\left(\frac{p_s}{p_1}\right)^{\left(\frac{k-1}{k}\right)}$$

or $$1+\frac{M_1^2}{2}(k-1)=\left(\frac{p_s}{p_1}\right)^{\left(\frac{k-1}{k}\right)} \qquad \left(\because\ \frac{V_1^2}{C_1^2}=M_1^2\right)$$

or $$\left(\frac{p_s}{p_1}\right)^{\frac{k-1}{k}}=\left[1+\frac{(k-1)}{2}M_1^2\right] \text{ or } \frac{p_s}{p_1}=\left[1+\frac{k-1}{2}M_1^2\right]^{\left(\frac{k}{k-1}\right)} \qquad ...(iv)$$

$\therefore$ $$p_s=p_1\left[1+\frac{k-1}{2}M_1^2\right]^{\left(\frac{k}{k-1}\right)} \qquad ...(15.21)$$

Equation (15.21) gives the value of stagnation pressure.

In equation (15.21), for $M<1$, the term $\frac{k-1}{2}{M_1}^2$ will be less than 1 and hence the R.H.S. of this equation can be expressed by Binomial theorem as

$$p_s = p_1\left[1+\left(\frac{k}{k-1}\right)\times\left(\frac{k-1}{2}\cdot M_1^2\right)+\left(\frac{k}{k-1}\right)\left(\frac{k}{k-1}-1\right)\left(\frac{k-1}{2}\cdot M_1^2\right)^2\Big/2!\right.$$

$$\left.+\left(\frac{k}{k-1}\right)\left(\frac{k}{k-1}-1\right)\left(\frac{k}{k-1}-2\right)\left(\frac{k-1}{2}\cdot M_1^2\right)^3\Big/\,3!+.....\right]$$

$$= p_1\left[1+\frac{k}{2}M_1^2+\frac{k}{8}M_1^4+\frac{k(2-k)}{48}M_1^6+.....\right]$$

$$= p_1 + p_1\left[\frac{k}{2}M_1^2+\frac{k}{8}M_1^4+\frac{k(2-k)}{48}M_1^6+.....\right]$$

or $$\frac{p_s-p_1}{p_1} = \left[\frac{k}{2}M_1^2+\frac{k}{8}M_1^4+\frac{k(2-k)}{48}M_1^6+.....\right]$$

$$= \frac{k}{2}M_1^2\left[1+\frac{1}{4}M_1^2+\frac{(2-k)}{24}M_1^4+.....\right]$$

$$= \frac{k}{2}\frac{V_1^2}{C_1^2}\left[1+\frac{1}{4}M_1^2+\frac{(2-k)}{24}M_1^4+.....\right]$$

$$= \frac{k}{2}\frac{V_1^2}{\left(k\frac{p_1}{\rho_1}\right)}\left[1+\frac{1}{4}M_1^2+\frac{(2-k)}{24}M_1^4+.....\right] \qquad \left(\because\ C_1=\sqrt{\frac{kp_1}{\rho_1}}\right)$$

$$= \frac{1}{2}.\frac{\rho_1 V_1^2}{p_1}\left[1+\frac{1}{4}M_1^2+\frac{(2-k)}{24}M_1^4+.....\right]$$

or $$(p_s-p_1) = \frac{1}{2}\cdot\rho_1\cdot V_1{}^2\left[1+\frac{1}{4}M_1^2+\frac{(2-k)}{24}M_1^4+.....\right] \qquad ...(15.21A)$$

From the above equation, it is clear that when the approaching velocity V_1 is small compared with the velocity of sound wave C_1, then M_1 will be very small and the term $1+\frac{1}{4}M_1{}^2+\frac{(2-k)}{24}M_1{}^4+.....$ will be nearly equal to 1.

Hence equation (15.21*A*) becomes as

$\therefore$ $$p_s - p_1 = \frac{1}{2}\cdot\rho_1\times V_1{}^2 \text{ or } p_s = p_1+\frac{1}{2}\rho_1 V_1{}^2$$

But when approaching velocity becomes high then M_1 is not small and equation (15.21*A*) is expressed as

$$\frac{(p_s-p_1)}{\frac{1}{2}\times\rho_1\times V_1^2} = 1+\frac{1}{4}M_1^2+\frac{(2-k)}{24}M_1{}^4+..... \qquad ...(15.21B)$$

15.7.2 Expression for Stagnation Density (ρ_s). From equation (*ii*), we have

$$\left(\frac{\rho_1}{\rho_s}\right) = \left(\frac{p_1}{p_s}\right)^{\frac{1}{k}} \text{ or } \left(\frac{\rho_s}{\rho_1}\right) = \left(\frac{p_s}{p_1}\right)^{\frac{1}{k}} \qquad \text{(Taking reciprocal)}$$

$$\therefore \qquad \rho_s = \rho_1 \left[\frac{p_s}{p_1}\right]^{\frac{1}{k}}$$

Substituting the value of $\left(\frac{p_s}{p_1}\right)$ from equation (*iv*),

$$\rho_s = \rho_1 \left[\left(1 + \frac{k-1}{2} M_1^2\right)^{\frac{k}{k-1}}\right]^{\frac{1}{k}} \text{ or } \rho_s = \rho_1 \left[1 + \frac{k-1}{2} M_1^2\right]^{\frac{1}{k-1}} \qquad ...(15.22)$$

15.7.3 Expression for Stagnation Temperature (T_s). Equation of state is given by equation (15.2) as $\frac{p}{\rho} = RT$

For the stagnation point, we have equation of state as $\frac{p_s}{\rho_s} = RT_s$ ...(15.22*A*)

$$\therefore \qquad T_s = \frac{1}{R}\frac{p_s}{\rho_s}$$

Substituting the value of p_s and ρ_s from equations (15.21) and (15.22), we have

$$T_s = \frac{1}{R} \frac{p_1\left[1 + \left(\frac{k-1}{2}\right) M_1^2\right]^{\left(\frac{k}{k-1}\right)}}{\rho_1\left[1 + \left(\frac{k-1}{2}\right) M_1^2\right]^{\frac{1}{k-1}}}$$

$$= \frac{1}{R}\frac{p_1}{\rho_1}\left[1 + \left(\frac{k-1}{2}\right) M_1^2\right]^{\left(\frac{k}{k-1}\right) - \left(\frac{1}{k-1}\right)}$$

$$= \frac{1}{R}\frac{p_1}{\rho_1}\left[1 + \left(\frac{k-1}{2}\right) M_1^2\right]^{\left(\frac{k-1}{k-1}\right)} = \frac{p_1}{\rho_1 R}\left[1 + \left(\frac{k-1}{2}\right) M_1^2\right]$$

$$= T_1\left[1 + \left(\frac{k-1}{2}\right) M_1^2\right] \qquad \left(\because \frac{p_1}{\rho_1} = RT_1\right) \qquad ...(15.23)$$

Problem 15.11 *Find the Mach number when an aeroplane is flying at 1100 km/hour through still air having a pressure of 7 N/cm^2 and temperature – 5°C. Wind velocity may be taken as zero. Take R = 287.14 J/kg K. Calculate the pressure, temperature and density of air at stagnation point on the nose of the plane. Take k = 1.4.*

Solution. Given :

Speed of aeroplane, $V = 1100 \text{ km/hr} = \dfrac{1100 \times 1000}{60 \times 60} = 305.55 \text{ m/s}$

Pressure of air, $p_1 = 7 \text{ N/cm}^2 = 7 \times 10^4 \text{ N/m}^2$

Temperature, $t_1 = -5°\text{C}$

$\therefore$ $T_1 = -5 + 273 = 268°\text{K}$

$R = 287.14 \text{ J/kg K}$

$k = 1.4$

Using relation $C = \sqrt{kRT}$ for velocity of sound for adiabatic process, we have

$$C_1 = \sqrt{1.4 \times 287.14 \times 268} = 328.2 \text{ m/s}$$

$\therefore$ Mach number, $M_1 = \dfrac{V_1}{C_1} = \dfrac{305.55}{328.20} = 0.9309 \simeq \mathbf{0.931.}$ **Ans.**

Stagnation Pressure, p_s. Using equation (15.21) for stagnation pressure,

$$p_s = p_1 \left[1 + \frac{k-1}{2} M_1^2\right]^{\frac{k}{k-1}}$$

$$= 7.0 \times 10^4 \left[1 + \frac{1.4 - 1.0}{2.0} \times (.931)^2\right]^{\frac{1.4}{1.4-1.0}}$$

$$= 7.0 \times 10^4 \, [1 + .1733]^{\frac{1.4}{.4}}$$

$$= 7.0 \times 10^4 \, [1.1733]^{3.5} = 12.24 \times 10^4 \text{ N/m}^2 = \mathbf{12.24 \text{ N/cm}^2.}$$ **Ans.**

Stagnation Temperature, T_s. Using equation (15.23) for stagnation temperature,

$$T_s = T_1 \left[1 + \frac{k-1}{2} M_1^2\right]$$

$$= 268 \left[1 + \frac{1.4 - 1.0}{2.0} \times (.931)^2\right] = 268 \, [1.1733] = 314.44°\text{K}$$

$\therefore$ $t_s = T_s - 273 = 314.43 - 273 = \mathbf{41.44°C.}$ **Ans.**

Stagnation Density, ρ_s. Using equation of state (15.22 A) for stagnation density, $\dfrac{p_s}{\rho_s} = RT_s$

$\therefore$ $$\rho_s = \frac{p_s}{RT_s} \qquad ...(i)$$

In equation (*i*) given above, if *R* is taken as 287.14 J/kg K, then pressure should be taken in N/m^2 so that the value of ρ is in kg/m^3. Hence $p_s = 12.24 \times 10^4$ N/m^2 and $T_s = 314.44°$K.

$\therefore$ $$\rho_s = \frac{12.24 \times 10^4}{287.14 \times 314.44} = \mathbf{1.355 \text{ kg/m}^3.}$$ **Ans.**

Problem 15.12 *Calculate the stagnation pressure, temperature and density at the stagnation point on the nose of a plane, which is flying at 800 km/hour through still air having a pressure 8.0 N/cm^2 (abs.) and temperature – 10°C. Take R = 287 J/kg K and k = 1.4.*

Solution. Given :

Speed of plane, $V = 800 \text{ km/hour} = \dfrac{800 \times 1000}{60 \times 60} = 222.22 \text{ m/s}$

Pressure of air, $p_1 = 8.0 \text{ N/cm}^2 = 8.0 \times 10^4 \text{ N/m}^2$

Temperature, $t_1 = -10°\text{C}$

$\therefore$ $T_1 = -10 + 273 = 263°\text{K}$

$R = 287 \text{ J/kg°K}$

$k = 1.4$

For adiabatic flow, the velocity of sound is given by

$$C = \sqrt{kRT} = \sqrt{1.4 \times 287 \times 263} = 325.07 \text{ m/s}$$

$\therefore$ Mach number, $M = \dfrac{V}{C} = \dfrac{222.22}{325.07} = 0.683.$

This Mach number is the local Mach number and hence equal to M_1.

$\therefore$ $M_1 = 0.683$

Using equation (15.21) for stagnation pressure,

$$p_s = p_1 \left[1 + \frac{k-1}{2} M_1^2\right]^{\left(\frac{k}{k-1}\right)} = 8.0 \times 10^4 \left[1 + \frac{1.4 - 1.0}{2.0} \times (.683)^2\right]^{\left(\frac{1.4}{1.4-1.0}\right)}$$

$$= 8.0 \times 10^4 \, [1.0933]^{3.5} = 10.93 \times 10^4 \text{ N/m}^2 = \mathbf{10.93 \text{ N/cm}^2. \text{ Ans.}}$$

Using equation (15.23) for stagnation temperature,

$$T_s = T_1 \left(1 + \frac{k-1}{2} M_1^2\right) = 263 \left(1 + \frac{1.4 - 1.0}{2.0} \times (.683)^2\right)$$

$$= 263 \, [1.0933] = 287.5 \text{ K}$$

$\therefore$ $t_s = T_s - 273 = 287.5 - 723 = \mathbf{14.5°C. \text{ Ans.}}$

Using equation (15.2), $\dfrac{p}{\rho} = RT$

For stagnation point, $\dfrac{p_s}{\rho_s} = RT_s$ $\quad \therefore \quad \rho_s = \dfrac{p_s}{RT_s}$

As $R = 287$ J/kg K, the value of p_s should be taken in N/m^2 so that the value of ρ_s is obtained in kg/m^3.

$\therefore$ $p_s = 10.93 \times 10^4 \text{ N/m}^2$

$\therefore$ Stagnation density, $\rho_s = \dfrac{10.93 \times 10^4}{287 \times 287.5} = \mathbf{1.324 \text{ kg/m}^3. \text{ Ans.}}$

▶15.8 AREA VELOCITY RELATIONSHIP FOR COMPRESSIBLE FLOW

The area velocity relationship for incompressible fluid is given by the continuity equation as

$$A \times V = \text{Constant}.$$

From the above equation, it is clear that with the increase of area, velocity decreases. But in case of compressible fluid, the continuity equation is given by, $\rho AV = \text{Constant}$. ...(*i*)

From this relation, it is clear that with the change of area, both the velocity and density are affected. Hence to find the relation between area and velocity for compressible fluid we proceed as given below.

Differentiating equation (*i*), we get

$$\rho d(AV) + AVd\rho = 0 \quad \text{or} \quad \rho [AdV + VdA] + AVd\rho = 0$$

or $$\rho AdV + \rho VdA + AVd\rho = 0.$$

Dividing by ρAV, we get $$\frac{dV}{V} + \frac{dA}{A} + \frac{d\rho}{\rho} = 0 \qquad ...(ii)$$

The Euler's equation for compressible fluid is given by equation (15.8), as

$$\frac{dp}{\rho} + VdV + gdz = 0$$

Neglecting the z term, the above equation is written as $\frac{dp}{\rho} + VdV = 0$.

This equation can also be written as $\frac{dp}{\rho} \times \frac{d\rho}{d\rho} + VdV = 0$ (Dividing and multiplying by $d\rho$)

or $$\frac{dp}{d\rho} \times \frac{d\rho}{\rho} + VdV = 0$$

But $$\frac{dp}{d\rho} = C^2 \text{ from equation (15.15)}$$

Hence above equation becomes as

$$C^2 \frac{d\rho}{\rho} + VdV = 0 \quad \text{or} \quad C^2 \frac{d\rho}{\rho} = -VdV \quad \text{or} \quad \frac{d\rho}{\rho} = -\frac{VdV}{C^2}$$

Substituting the value of $\frac{d\rho}{\rho}$ in equation (*ii*), we get

$$\frac{dV}{V} + \frac{dA}{A} - \frac{VdV}{C^2} = 0 \quad \text{or} \quad \frac{dA}{A} = \frac{VdV}{C^2} - \frac{dV}{V} = \frac{dV}{V}\left[\frac{V^2}{C^2} - 1\right]$$

$$\therefore \qquad \frac{dA}{A} = \frac{dV}{V}\left[M^2 - 1\right] \qquad ...(15.24)$$

Equation (15.24) gives the relationship between change of area with change of velocity for different Mach numbers. The following are the important conclusions :

(*i*) For $M < 1$, the flow is sub-sonic and the right-hand side of equation (15.24) is negative as $(M^2 - 1)$ is negative for the values of $M < 1$. Hence $\frac{dA}{A} > 0$, $\frac{dV}{V} < 0$. This means that with the increase of area, the velocity decreases and *vice versa.*

(*ii*) For $M > 1$, the flow is super-sonic. The value of $(M^2 - 1)$ will be positive and hence right-hand side of equation (15.24) will be positive. Hence $\frac{dA}{A} > 0$ and also $\frac{dV}{V} > 0$. This means that with the increase of area, velocity also increases.

(*iii*) For $M = 1$, the flow is called sonic flow. The value of $(M^2 - 1)$ is zero. Hence right-hand side of equation (15.24) will be zero. Hence $\frac{dA}{A} = 0$. This means area is constant.

▶15.9 FLOW OF COMPRESSIBLE FLUID THROUGH ORIFICES AND NOZZLES FITTED TO A LARGE TANK

Consider a compressible fluid filled in a large reservoir or vessel to which a short nozzle is fitted as shown in Fig. 15.5. If the pressure drop of the compressible fluid, flowing through the nozzle from reservoir is small, the process is considered to be isothermal. But if the pressure drop is large, the process is considered to be adiabatic.

Consider two points 1 and 2 inside the tank and at the exit of the nozzle respectively.

Let V_1 = Velocity of fluid in the tank,
p_1 = Pressure of fluid at point 1,
ρ_1 = Density of fluid at point 1,
T_1 = Temperature of fluid at point 1, and
V_2, p_2, ρ_2, T_2 = Corresponding values of velocity, pressure, density and temperature at point 2.

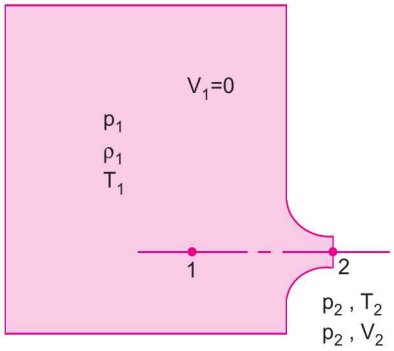


Fig. 15.5 *Pressure tank fitted with a nozzle.*

Considering the process to be adiabatic. Then from Bernoulli's equation for adiabatic flow from equation (15.13), we have

$$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1 g} + \frac{V_1^2}{2g} + Z_1 = \left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2 g} + \frac{V_2^2}{2g} + Z_2.$$

But $Z_1 = Z_2$ and also V_1 = Velocity of fluid in tank = 0

$$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1 g} + 0 = \left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2 g} + \frac{V_2^2}{2g}$$

or $$\left(\frac{k}{k-1}\right)\left[\frac{p_1}{\rho_1 g} - \frac{p_2}{\rho_2 g}\right] = \frac{V_2^2}{2g}$$

or $$\left(\frac{k}{k-1}\right)\left(\frac{p_1}{\rho_1} - \frac{p_2}{\rho_2}\right) = \frac{V_2^2}{2} \qquad \left(\text{Cancelling } \frac{1}{g}\right)$$

or $$V_2 = \sqrt{\frac{2k}{(k-1)}\left[\frac{p_1}{\rho_1} - \frac{p_2}{\rho_2}\right]} = \sqrt{\frac{2k}{(k-1)}\frac{p_1}{\rho_1}\left(1 - \frac{p_2}{\rho_2} \times \frac{\rho_1}{p_1}\right)} \qquad ...(i)$$

But for adiabatic flow from equation (15.4), we have

$$\frac{p_1}{\rho_1^k} = \frac{p_2}{\rho_2^k} \text{ or } \frac{p_1}{p_2} = \left(\frac{\rho_1}{\rho_2}\right)^k$$

$$\therefore \quad \frac{\rho_1}{\rho_2} = \left(\frac{p_1}{p_2}\right)^{\frac{1}{k}} \qquad ...(ii)$$

Substituting the value of $\frac{\rho_1}{\rho_2}$ in equation (*i*), we get

$$V_2 = \sqrt{\frac{2k}{(k-1)}\frac{p_1}{\rho_1}\left[1 - \frac{p_2}{p_1} \times \left(\frac{p_1}{p_2}\right)^{\frac{1}{k}}\right]} = \sqrt{\frac{2k}{(k-1)}\frac{p_1}{\rho_1}\left[1 - \left(\frac{p_2}{p_1}\right)^{1-\frac{1}{k}}\right]}$$

$$= \sqrt{\frac{2k}{(k-1)}\frac{p_1}{\rho_1}\left[1 - \left(\frac{p_2}{p_1}\right)^{\frac{k-1}{k}}\right]} \qquad ...(15.25)$$

Let $\frac{p_2}{p_1} = n$. Then above equation becomes as $V_2 = \sqrt{\frac{2k}{(k-1)}\frac{p_1}{\rho_1}\left[1 - n^{\frac{k-1}{k}}\right]}$ ...(*iii*)

The mass rate of flow of the compressible fluid is given as

$m = \rho_2 A_2 V_2$, where A_2 = Area at the exit of nozzle

$$= \rho_2 A_2 \sqrt{\frac{2k}{(k-1)}\frac{p_1}{\rho_1}\left[1 - n^{\frac{k-1}{k}}\right]} \qquad \text{[Substitute } V_2 \text{ from } (iii)\text{]}$$

$$\Rightarrow \quad = A_2 \sqrt{\frac{2k}{(k-1)}\frac{p_1}{\rho_1} \times \rho_2^2\left[1 - n^{\frac{k-1}{k}}\right]} \qquad \text{(taking } \rho_2 \text{ inside)}$$

But from equation (*ii*), we have $\rho_2 = \frac{\rho_1}{\left(\frac{p_1}{p_2}\right)^{(1/k)}} = \rho_1\left(\frac{p_2}{p_1}\right)^{\frac{1}{k}} = \rho_1 n^{\frac{1}{k}} \qquad \left(\because \frac{p_2}{p_1} = n\right)$

$$\therefore \quad \rho_2^2 = \rho_1^2 n^{\frac{2}{k}}$$

Substituting this value of ρ_2^2 in the above equation, we get

$$m = A_2 \sqrt{\frac{2k}{k-1}\frac{p_1}{\rho_1} \times \rho_1^2 n^{2/k}\left[1 - n^{\frac{k-1}{k}}\right]} = A_2 \sqrt{\frac{2k}{(k-1)} p_1\rho_1\left[n^{\frac{2}{k}} - n^{\frac{k-1}{k}+\frac{2}{k}}\right]}$$

$$= A_2 \sqrt{\frac{2k}{k-1} p_1\rho_1\left[n^{\frac{2}{k}} - n^{\frac{k+1}{k}}\right]} \qquad ...(15.26)$$

The mass rate of flow (m) depends on the value of n for the given values of p_1 and ρ_1 at 1.

15.9.1 Value of n or $\frac{p_2}{p_1}$ for Maximum Value of Mass Rate of Flow. For maximum value of m,

we have $$\frac{\partial m}{\partial n} = 0$$

or $$\frac{\partial}{\partial n}\left[n^{\frac{2}{k}} - n^{\frac{k+1}{k}}\right] = 0 \qquad \left(\because \quad \frac{2k}{k-1} p_1\rho_1 = \text{Constant}\right)$$

or $$\frac{2}{k} n^{\frac{2}{k}-1} - \frac{k+1}{k} n^{\frac{k+1}{k}-1} = 0 \quad \text{or} \quad \frac{2}{k} n^{\frac{2-k}{k}} = \frac{k+1}{k} n^{\frac{k+1-k}{k}} = \frac{k+1}{k} n^{\frac{1}{k}}$$

or $$n^{\frac{2-k}{k}} = \left(\frac{k+1}{k}\right) \times \frac{k}{2} \times n^{\frac{1}{k}} = \frac{k+1}{2} n^{\frac{1}{k}} \quad \text{or} \quad \frac{n^{\frac{2-k}{k}}}{n^{\frac{1}{k}}} = \frac{k+1}{2}$$

$\therefore$ $$n^{\frac{2-k}{k}-\frac{1}{k}} = \frac{k+1}{2} \quad \text{or} \quad n^{\frac{1-k}{k}} = \frac{k+1}{2}$$

$\therefore$ $$n^{\frac{-(k-1)}{k}} = \frac{k+1}{2} \quad \text{or} \quad \frac{1}{n^{\frac{k-1}{k}}} = \frac{k+1}{2} = \frac{1}{\left(\frac{2}{k+1}\right)}$$

$\therefore$ $$n^{\frac{k-1}{k}} = \frac{2}{k+1} \qquad \ldots(15.27)$$

Equation (15.27) is the condition for maximum value of mass rate of flow through the nozzle.

For $k = 1.4$, the value of n can be obtained from equation (15.27) as

$$n^{\frac{1.4-1.0}{1.4}} = \frac{2}{1.4+1} = \frac{2}{2.4} \text{ or } n^{2/7} = \frac{2}{2.4}$$

$\therefore$ $$n = \left(\frac{2}{2.4}\right)^{7/2} = 0.528 \text{ or } \frac{p_2}{p_1} = n = 0.528. \qquad \ldots(15.28)$$

15.9.2 Value of V_2 for Maximum Rate of Flow of Fluid. From equation (15.27), the value of n is given as

$$n = \left(\frac{2}{k+1}\right)^{\frac{k}{k-1}}$$

Substituting this value of n in equation (*iii*), we get

$$V_2 = \sqrt{\frac{2k}{k-1} \frac{p_1}{\rho_1}\left(1 - n^{\frac{k-1}{k}}\right)}$$

$$= \sqrt{\frac{2k}{k-1}\frac{p_1}{\rho_1}\left[1-\left(\frac{2}{k+1}\right)^{\frac{k}{k-1}\times\frac{(k-1)}{k}}\right]}$$

$$= \sqrt{\frac{2k}{k-1}\frac{p_1}{\rho_1}\left(1-\frac{2}{k+1}\right)} = \sqrt{\frac{2k}{k-1}\frac{p_1}{\rho_1}\left[\frac{k+1-2}{k+1}\right]}$$

$$= \sqrt{\frac{2k}{k-1}\frac{p_1}{\rho_1}\left[\frac{k-1}{k+1}\right]} = \sqrt{\frac{2k}{k+1}\frac{p_1}{\rho_1}} \qquad \ldots(15.29)$$

15.9.3 Maximum Rate of Flow of Fluid Through Nozzle. Mass rate of flow of fluid through nozzle is given by equation (15.26) as

$$m = A_2\sqrt{\frac{2k}{(k-1)}p_1\rho_1\left[n^{2/k}-n^{\frac{k+1}{k}}\right]}$$

From maximum rate of flow, from equation (15.27), we have

$$n^{\frac{k-1}{k}} = \frac{2}{k+1}$$

$$\therefore \qquad n = \left(\frac{2}{k+1}\right)^{\frac{k}{k-1}}$$

Substituting the value of n in the above equation, we have

$$m_{max} = A_2\sqrt{\left(\frac{2k}{k-1}\right)p_1\rho_1\left[\left(\frac{2}{k+1}\right)^{\frac{k}{k-1}\times\frac{2}{k}}-\left(\frac{2}{k+1}\right)^{\frac{k}{k-1}\times\frac{k+1}{k}}\right]}$$

$$= A_2\sqrt{\frac{2k}{k-1}p_1\rho_1\left[\left(\frac{2}{k+1}\right)^{\frac{2}{k-1}}-\left(\frac{2}{k+1}\right)^{\frac{k+1}{k-1}}\right]}$$

For air, $k = 1.4$,

$$\therefore \qquad m_{max} = A_2\sqrt{\frac{2\times1.4}{1.4-1.0}p_1\rho_1\left[\left(\frac{2}{1.4+1}\right)^{\frac{2}{1.4-1.0}}-\left(\frac{2}{1.4+1.0}\right)^{\frac{1.4+1.0}{1.4-1.0}}\right]}$$

$$= A_2\sqrt{\frac{2.8}{0.4}p_1\rho_1\left[\left(\frac{2}{2.4}\right)^{2/.4}-\left(\frac{2}{2.4}\right)^{2.4/.4}\right]}$$

$$= A_2\sqrt{7\times p_1\rho_1[.4018-.3348]} = A_2\times0.685\times\sqrt{p_1\rho_1}$$

$$= 0.685\,A_2\sqrt{p_1\rho_1}\,. \qquad \ldots(15.30)$$

15.9.4 Variation of Mass Rate of Flow of Compressible Fluid with Pressure Ratio $\left(\frac{p_2}{p_1}\right)$.

Fig. 15.6 shows the variation of mass rate of flow of compressible fluid with different pressure ratio $\left(\frac{p_2}{p_1}\right)$.

It is seen that when $\frac{p_2}{p_1}$ is less than critical pressure ratio of 0.528, the mass rate of flow is constant and is equal to the mass rate of flow corresponding to pressure ratio = 0.528. But if the pressure ratio is more than 0.528, mass rate of flow decreases as shown by dotted line.

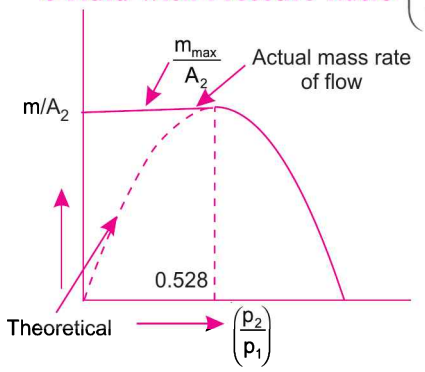


Fig. 15.6

15.9.5 Velocity at Outlet of Nozzle for Maximum Rate of Flow is Equal to Sonic Velocity.

This is proved as given below.

The velocity at the outlet of nozzle for maximum rate of flow is given by equation (15.29) as

$$V_2 = \sqrt{\left(\frac{2k}{k+1}\right)\frac{p_1}{\rho_1}} \qquad ...(i)$$

Now pressure ratio $\frac{p_2}{p_1} = n \quad \therefore \quad p_1 = \frac{p_2}{n}$

Also for adiabatic flow, $\frac{p_1}{\rho_1^k} = \frac{p_2}{\rho_2^k}$ or $\frac{p_1}{p_2} = \left(\frac{\rho_1}{\rho_2}\right)^k$

or $$\frac{\rho_1}{\rho_2} = \left(\frac{p_1}{p_2}\right)^{1/k} = \left(\frac{p_2}{p_1}\right)^{-1/k} = n^{-1/k} \qquad \left(\because \frac{p_2}{p_1} = n\right)$$

$\therefore$ $$\rho_1 = \rho_2 n^{-1/k}$$

Substituting the value of p_1 and ρ_1 in equation (i), we get

$$V_2 = \sqrt{\left(\frac{2k}{k+1}\right)\frac{p_2}{\frac{n}{\rho_2 n^{-1/k}}}} = \sqrt{\frac{2k}{k+1} \times \frac{p_2}{\rho_2} \times \frac{1}{n} \times \frac{1}{n^{-1/k}}}$$

$$= \sqrt{\frac{2k}{k+1} \times \frac{p_2}{\rho_2} \times \frac{1}{n^{(1-1/k)}}} = \sqrt{\frac{2k}{k+1} \times \frac{p_2}{\rho_2} \times \frac{1}{n^{(k-1)/k}}}$$

But $n^{\frac{k-1}{k}} = \frac{2}{k+1}$ from equation (15.27)

$\therefore$ $$V_2 = \sqrt{\frac{2k}{k+1} \times \frac{p_2}{\rho_2} \times \frac{(k+1)}{2}} = \sqrt{\frac{kp_2}{\rho_2}}$$

$$= C_2. \qquad \left(\because \sqrt{\frac{kp_2}{\rho_2}} = C_2\right) \quad ...(15.31)$$

Problem 15.13 *Find the velocity of air flowing at the outlet of a nozzle, fitted to a large vessel which contains air at a pressure of 294.3 N/cm^2 (abs.) and at a temperature of 20°C. The pressure at the outlet of the nozzle is 206 N/cm^2 (abs). Take k = 1.4 and K = 287 J/kg°K.*

Solution. Given :

Pressure inside vessel, $p_1 = 294.3 \text{ N/cm}^2 = 294.3 \times 10^4 \text{ N/m}^2$

Temperature inside vessel, $t_1 = 20°\text{C}$

$\therefore$ $T_1 = 20 + 273 = 293°\text{K}$

Pressure at the nozzle, $p_2 = 206 \text{ N/cm}^2 = 206 \times 10^4 \text{ N/m}^2$

$R = 287 \text{ J/kg°K}$

$k = 1.4$

Using equation (15.25) for the velocity,

$$V_2 = \sqrt{\left(\frac{2k}{k-1}\right)\frac{p_1}{\rho_1}\left[1-\left(\frac{p_2}{p_1}\right)^{\frac{k-1}{k}}\right]}$$

$$= \sqrt{\left(\frac{2\times 1.4}{1.4\times 1.0}\right)\frac{294.3\times 10^4}{\rho_1}\left[1-\left(\frac{206\times 10^4}{294.3\times 10^4}\right)^{\frac{1.4-1.0}{1.4}}\right]}$$

$$= \sqrt{\frac{2.8}{0.4}\times\frac{294.3\times 10^4}{\rho_1}\left[1-0.7^{0.4/1.4}\right]} = \sqrt{\frac{7\times 294.3\times 10^4}{\rho_1}\left[1-.903\right]} \quad ...(i)$$

The value of ρ_1 is calculated from equation of state as

$$\frac{p_1}{\rho_1} = RT_1 \qquad \therefore \quad \rho_1 = \frac{p_1}{RT_1}$$

In this equation if R is taken in J/kg K, p_1 should be in N/m^2. Then ρ_1 will be in kg/m^3.

$\therefore$ $p_1 = 294.3 \times 10^4 \text{ N/m}^2$

$\therefore$ $$\rho_1 = \frac{294.3\times 10^4}{287\times 293} = 34.99 \text{ kg/m}^3.$$

Substituting the value of $\rho_1 = 34.99 \text{ kg/m}^3$ in equation (i),

$$V_2 = \sqrt{7\times\frac{294.3\times 10^4}{34.99}\left[1-.903\right]} = \textbf{239.2 m/s. Ans.}$$

Problem 15.14 *A tank contains air at a temperature of 30°C. Air flows from the tank into atmosphere through a convergent nozzle. The diameter at the outlet of the nozzle is 25 mm. Assuming adiabatic flow, find the mass rate of flow of air through the nozzle when the pressure of air in tank is (i) 3.924 N/cm^2 (gauge), (ii) 33.354 N/cm^2 (gauge). Take k = 1.4, R = 287 J/kg°K and atmospheric pressure = 10.104 N/cm^2 (abs).*

Solution. Given :

Temperature in tank, $t_1 = 30°\text{C}$

$\therefore$ $T_1 = 30 + 273 = 303^\circ K$

Diameter at the nozzle, $D_2 = 25$ mm $= 0.025$ m.

$\therefore$ Area, $A_2 = \frac{\pi}{4} D_2^2 = \frac{\pi}{4} (.025)^2 = 0.0004908$ m^2

$R = 287$ J/kg°K

$k = 1.4$.

(i) Mass rate of flow of air when pressure in tank is 3.924 N/cm^2 (gauge).

$p_1 = 3.924$ N/cm^2 (gauge)

$= 3.924 + 10.104 = 14.028$ N/cm^2 (abs.)

$= 14.028 \times 10^4$ N/cm^2 (abs.)

Pressure at the nozzle, p_2 = Atmospheric pressure

$= 10.104$ N/cm^2 $= 10.104 \times 10^4$ N/m^2

$\therefore$ Pressure ratio, $n = \frac{p_2}{p_1} = \frac{10.104 \times 10^4}{14.028 \times 10^4} = 0.7203.$

This pressure ratio is more than the pressure ratio 0.528 given by equation (15.28), hence mass rate of flow of air is given by equation (15.26), as

$$m = A_2 \sqrt{\frac{2k}{(k-1)} p_1 \rho_1 \left[n^{\frac{2}{k}} - n^{\frac{k+1}{k}} \right]} \qquad ...(i)$$

In this equation if p_1 is taken into N/m^2, then ρ_1 will be in kg/m^3. The value of ρ_1 is obtained from equation of state as

$$\frac{p_1}{\rho_1} = RT_1 \text{ or } \rho_1 = \frac{p_1}{RT_1}, \qquad \text{where } p_1 = 14.028 \text{ N/m}^2$$

$\therefore$ $$\rho_1 = \frac{14.028 \times 10^4}{287 \times 303} = 1.613 \text{ kg/m}^3.$$

Substituting this value of $\rho_1 = 1.613$ kg/m^3 and $p_1 = 14.028 \times 10^4$ N/m^2 in equation (i), we get

$$m = .0004908 \sqrt{\frac{2 \times 1.4}{1.4 - 1.0} \times 14.028 \times 10^4 \times 1.613 \left[.7203^{\frac{2}{1.4}} - .7203^{\frac{1.4+1.0}{1.4}} \right]}$$

$$= .0004908 \sqrt{1583935 \left[.7203^{1.4285} - .7203^{1.7142} \right]}$$

$$= .0004908 \sqrt{1583935 \left[.6258 - .5698 \right]} = \mathbf{0.146 \text{ kg/s. Ans.}}$$

(ii) Mass rate of flow of air when pressure in the tank is 33.354 N/cm^2 (gauge).

$\therefore$ $p_1 = 33.354 + 10.104 = 43.458$ N/cm^2 (abs.) $= 43.458 \times 10^4$ N/m^2 (abs).

p_2 = Atmospheric pressure = 10.104×10^4 N/m^2

$\therefore$ Pressure ratio, $n = \dfrac{p_2}{p_1} = \dfrac{10.104 \times 10^4}{43.458 \times 10^4} = 0.2325.$

This pressure ratio is less than the pressure ratio of 0.528. Hence as mentioned in Art. 15.9.4, the mass rate of flow will be corresponding to the pressure ratio of 0.528. Hence

$$m = A_2 \sqrt{\frac{2k}{k-1} p_1 \rho_1 \left[n^{\frac{2}{k}} - n^{\frac{(k+1)}{k}} \right]}$$

where $n = 0.528$, $p_1 = 43.458 \times 10^4$ N/m^2.

The value of $\rho_1 = \dfrac{p_1}{RT_1} = \dfrac{43.458 \times 10^4}{287 \times 303} = 4.99$ kg/m^3

$\therefore$ $$m = .0004908 \sqrt{\frac{2 \times 1.4}{(1.4 - 1.0)} \times 43.458 \times 10^4 \times 4.99 \left[.528^{\frac{2}{1.4}} - .528^{\frac{1.4+1.0}{1.4}} \right]}$$

$$= .004908 \sqrt{4906875 [.4015 - .3346]} = \textbf{0.494 kg/s. Ans.}$$

Problem 15.15 *A large tank contains air at 28.449 N/cm^2 gauge pressure and 24°C temperature. The air flows from the tank to the atmosphere through an orifice. If the diameter of the orifice is 20 mm, find the maximum rate of flow of air. Tank R = 287 J/kg°K, k = 1.4, atmospheric pressure = 10.104 N/cm^2.*

Solution. Given :

Pressure in tank, $p_1 = 28.449$ N/cm^2 (gauge)

$= 28.449 + 10.104 = 38.553$ N/cm^2 (abs.) $= 38.553 \times 10^4$ N/m^2

Temperature in tank, $t_1 = 24°C$

$\therefore$ $T_1 = 273 + 24 = 297°K$

$R = 287$ J/kg K

$k = 1.4$

Diameter of orifice, $D = 20$ mm $= 0.02$ m

$\therefore$ Area, $A = \dfrac{\pi}{4} (.02)^2 = .0003141$ m^2

Using equation of state, we get $\dfrac{p_1}{\rho_1} = RT_1$ or $\rho_1 = \dfrac{p_1}{RT_1} = \dfrac{38.553 \times 10^4}{287 \times 297} = 4.522$ kg/m^3

Maximum rate of flow of air is given by equation (15.30) as

$m_{max} = 0.685\ A_2 \sqrt{p_1 \rho_1}$ (Here $A_2 = A = 0.0003141$)

$= 0.685 \times .0003141 \sqrt{38.553 \times 10^4 \times 4.522} = \textbf{0.284 kg/s. Ans.}$

▶ 15.10 MASS RATE OF FLOW OF COMPRESSIBLE FLUID THROUGH VENTURIMETER

Consider a compressible fluid flowing through the horizontal venturimeter. Let the conditions of flow is represented by suffix 1 at the inlet of venturimeter and by suffix 2 at the throat of the venturimeter. Considering the flow as adiabatic, we have from Bernoulli's equation at sections 1 and 2 from equation (15.13), as

$$\left(\frac{k}{k-1}\right)\frac{p_1}{\rho_1 g}+\frac{V_1^2}{2g}=\left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2 g}+\frac{V_2^2}{2g} \qquad (\because \quad z_1 = z_2)$$

or $$\frac{k}{(k-1)}\frac{p_1}{\rho_1}+\frac{V_1^2}{2}=\left(\frac{k}{k-1}\right)\frac{p_2}{\rho_2}+\frac{V_2^2}{2} \qquad \left(\text{Cancelling } \frac{1}{g}\right)$$

or $$\frac{k}{(k-1)}\left[\frac{p_1}{\rho_1}-\frac{p_2}{\rho_2}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2}$$

$$\frac{k}{(k-1)}\frac{p_1}{\rho_1}\left[1-\frac{p_2}{\rho_2}\times\frac{\rho_1}{p_1}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2} \qquad ...(i)$$

For adiabatic flow, $$\frac{p_1}{\rho_1^k}=\frac{p_2}{\rho_2^k} \quad \therefore \quad \frac{p_1}{p_2}=\left(\frac{\rho_1}{\rho_2}\right)^k$$

or $$\frac{\rho_1}{\rho_2}=\left(\frac{p_1}{p_2}\right)^{1/k}=\left(\frac{p_2}{p_1}\right)^{-1/k} \qquad ...(ii)$$

Substituting this value of $\frac{\rho_1}{\rho_2}$ in equation (*i*), we get

$$\frac{k}{k-1}\frac{p_1}{\rho_1}\left[1-\frac{p_2}{p_1}\times\left(\frac{p_2}{p_1}\right)^{-1/k}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2}$$

or $$\frac{k}{k-1}\frac{p_1}{\rho_1}\left[1-\left(\frac{p_2}{p_1}\right)^{1-1/k}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2}$$

or $$\frac{k}{k-1}\frac{p_1}{\rho_1}\left[1-\left(\frac{p_2}{p_1}\right)^{k-1/k}\right]=\frac{V_2^2}{2}-\frac{V_1^2}{2} \qquad ...(iii)$$

Applying continuity for sections 1 and 2, we have

$$\rho_1 A_1 V_1 = \rho_2 A_2 V_2 \qquad \therefore \quad V_1 = \frac{\rho_2 A_2 V_2}{\rho_1 A_1}$$

Substituting the value of V_1 in equation (*iii*), we get

$$\frac{k}{(k-1)}\frac{p_1}{\rho_1}\left[1-\left(\frac{p_2}{p_1}\right)^{k-1/k}\right]=\frac{V_2^2}{2}-\left(\frac{\rho_2 A_2 V_2}{\rho_1 A_1}\right)^2\times\frac{1}{2}=\frac{V_2^2}{2}\left[1-\frac{\rho_2^2 A_2^2}{\rho_1^2 A_1^2}\right] \quad ...(iv)$$

But from equation (*ii*), we have

$$\frac{\rho_1}{\rho_2}=\left(\frac{p_1}{p_2}\right)^{1/k} \quad \text{or} \quad \frac{\rho_2}{\rho_1}=\left(\frac{p_2}{p_1}\right)^{1/k} \quad \therefore \left(\frac{\rho_2}{\rho_1}\right)^2=\left(\frac{p_2}{p_1}\right)^{2/k}$$

Substituting this value in equation (*iv*), we get

$$\frac{k}{(k-1)}\frac{p_1}{\rho_1}\left[1-\left(\frac{p_2}{p_1}\right)^{k-1/k}\right]=\frac{V_2^2}{2}\left[1-\left(\frac{p_2}{p_1}\right)^{2/k}\times\frac{A_2^2}{A_1^2}\right]$$

$$V_2{}^2=\frac{\dfrac{2k}{(k-1)}\dfrac{p_1}{\rho_1}\left[1-\left(\dfrac{p_2}{p_1}\right)^{k-1/k}\right]}{\left[1-\left(\dfrac{p_2}{p_1}\right)^{2/k}\times\dfrac{A_2^2}{A_1^2}\right]}$$

$\therefore$

$$V_2=\sqrt{\frac{\dfrac{2k}{k-1}\dfrac{p_1}{\rho_1}\left[1-\left(\dfrac{p_2}{p_1}\right)^{\frac{k-1}{k}}\right]}{1-\left(\dfrac{p_2}{p_1}\right)^{2/k}\times\dfrac{A_2^2}{A_1^2}}}$$

$\therefore$ Mass rate of flow through venturimeter,

$$m=\rho_2 A_2 V_2=\rho_2 A_2\sqrt{\frac{\dfrac{2k}{k-1}\dfrac{p_1}{\rho_1}\left[1-\left(\dfrac{p_2}{p_1}\right)^{k-1/k}\right]}{\left[1-\left(\dfrac{A_2^2}{A_1^2}\right)\left(\dfrac{p_2}{p_1}\right)^{2/k}\right]}} \quad ...(15.32)$$

The only condition for equation (15.32) is that the pressure ratio $\left(\frac{p_2}{p_1}\right)$ should be more than the pressure ratio 0.528

Problem 15.16 *Find the mass rate of flow of air though a venturimeter having inlet diameter as 300 mm and throat diameter 150 mm. The pressure at the inlet of venturimeter is 1.4 kgf/cm^2 (1.4 ×9.81 N/cm^2) absolute and temperature of air at inlet is 15°C. The pressure at the throat is given as 1.3 kgf/cm^2 (1.3 ×9.81 N/cm^2) absolute. Take R = 287 J/kg°K and k = 1.4.*

Solution. Given :

Diameter at inlet, $D_1 = 300 \text{ mm} = 0.30 \text{ m}$

$\therefore$ Area, $A_1=\frac{\pi}{4}D_1{}^2=\frac{\pi}{4}(.30)^2=.07068 \text{ m}^2$

Diameter at throat, $D_2 = 150 \text{ mm} = 0.15 \text{ m}$

$\therefore$ Area, $A_2 = \frac{\pi}{4}(.15)^2 = 0.01767 \text{ m}^2$

Pressure of air at inlet, $p_1 = 1.4 \text{ kgf/cm}^2 = 1.4 \times 10^4 \text{ kgf/m}^2 = 1.4 \times 10^4 \times 9.81 \text{ N/m}^2$

Throat pressure, $p_2 = 1.3 \text{ kgf/cm}^2 = 1.3 \times 10^4 \times 9.81 \text{ N/m}^2$

$R = 287 \text{ J/kg°K}$

$k = 1.4$

Temperature at inlet, $t_1 = 15°\text{C}$

The pressure ratio, $\frac{p_2}{p_1} = \frac{1.3 \times 10^4 \times 9.81}{1.4 \times 10^4 \times 9.81} = 0.9285$

Density of gas at inlet is obtained from equation of state,

$$\frac{p_1}{\rho_1} = RT_1 \text{ or } \rho_1 = \frac{p_1}{RT_1} = \frac{1.4 \times 10^4 \times 9.81}{287 \times (273 + 15)}$$

where $T_1 = t_1 + 273 = 15 + 273 = 288°\text{K}$

$\therefore$ $$\rho_1 = \frac{1.4 \times 10^4 \times 9.81}{287 \times 288} = 1.66 \text{ kg/cm}^3$$

For adiabatic process, we have $\frac{p_1}{\rho_1^k} = \frac{p_2}{\rho_2^k}$ or $\left(\frac{\rho_2}{\rho_1}\right)^k = \frac{p_2}{p_1}$ or $\frac{\rho_2}{\rho_1} = \left(\frac{p_2}{p_1}\right)^{1/k}$

$\therefore$ $$\rho_2 = \rho_1 \left(\frac{p_2}{p_1}\right)^{1/k} = 1.66 \times (.9285)^{1/1.4} \qquad \left(\because \frac{p_2}{p_1} = .9285\right)$$

$= 1.574 \text{ kg/m}^3$.

Using equation (15.32) for the mass rate of flow through venturimeter, we get

$$m = \rho_2 A_2 \sqrt{\frac{\frac{2k}{k-1}\frac{p_1}{\rho_1}\left[1 - \left(\frac{p_2}{p_1}\right)^{k-1/k}\right]}{\left[1 - \left(\frac{A_2^2}{A_1^2}\right)\left(\frac{p_2}{p_1}\right)^{2/k}\right]}}$$

$$= 1.574 \times .01767 \sqrt{\frac{\frac{2 \times 1.4}{1.4 - 1} \times \frac{1.4 \times 10^4 \times 9.81}{1.66}\left[1 - .9285^{\frac{1.4 - 1.0}{1.4}}\right]}{\left[1 - \left(\frac{.01767^2}{.07068^2}\right)\left(.9285^{2/1.4}\right)\right]}}$$

$$= .0278 \sqrt{\frac{579144\left[1 - .9285^{.2857}\right]}{1 - .0625 \times .899}} = .0278 \times \sqrt{\frac{12145.57}{.9438}}$$

= 315 kg/s. Ans.

▶ 15.11 PITOT-STATIC TUBE IN A COMPRESSIBLE FLOW

The pitot-static tube, when used for determining the velocity at any point in a compressible fluid, gives only the difference between the stagnation head and static head. From this difference, the velocity of the incompressible fluid at that point is obtained from the relation

$$V = \sqrt{2gh}\text{, where } h = \text{Difference in two heads.}$$

But when the pitot-static tube is used for finding velocity at any point in a compressible fluid, the actual pressure difference shown by the gauges of the pitot-tube should be multiplied by a factor, for obtaining correct velocity at that point. The value of the factor depends upon the Mach number of the flow. Let us find an expression for the correction factor for *sub-sonic* flow.

At a point in pitot-static tube, the pressure becomes stagnation pressure, denoted by p_s. The expression for stagnation pressure, p_s is given by equation (15.21), as

$$p_s = p_1\left[1 + \frac{k-1}{2} M_1^2\right]^{\frac{k}{k-1}} \qquad ...(i)$$

where p_1 = Pressure of fluid far away from stagnation point,
M_1 = Mach number at point 1 far away from stagnation point.

For $M < 1$, the term $\frac{k-1}{2} M_1^2$ will be less than 1 and hence the right-hand side of equation (i) can be expanded by Binomial theorem* as

$$p_s = \left[1 + \frac{k-1}{2}M_1^2 \times \frac{k}{k-1} \times \frac{\left(\frac{k}{k-1}\right)\left(\frac{k}{k-1}-1\right)}{2!} \times \left(\frac{k-1}{2}M_1^2\right)^2 + \frac{\left(\frac{k}{k-1}\right)\left(\frac{k}{k-1}-1\right)\left(\frac{k}{k-1}-2\right)\left(\frac{k-1}{2}M_1^2\right)^3}{3!} + \ldots\right]$$

$$= p_1\left[1 + \frac{k}{2}M_1^2 + \frac{k}{8}M_1^4 + \frac{k(2-k)}{48}M_1^6 + \ldots\right]$$

$$= p_1 + p_1\left[\frac{k}{2}M_1^2 + \frac{k}{8}M_1^4 + \frac{k(2-k)}{48}M_1^6 + \ldots\right]$$

$$p_s - p_1 = p_1 \times \frac{k}{2} M_1^2\left[1 + \frac{M_1^2}{4} + \frac{(2-k)}{24}M_1^4 + \ldots\right] \qquad ...(ii)$$

But $$M_1^2 = \frac{V_1^2}{C_1^2}, \text{ where } C_1^2 = \frac{kp_1}{\rho_1}$$

---

* Binomial Theorem $(1+x)^n = 1 + nx + \frac{n(n-1)x^2}{2!} + \frac{n(n-1)(n-2)x^3}{3!} + \frac{n(n-1)(n-2)(n-3)x^4}{4!} \ldots\ldots$

$$= \frac{V_1^2}{\frac{kp_1}{\rho_1}} = \frac{V_1^2 \rho_1}{kp_1}$$

Substituting the value of M_1^2 in equation (*ii*), we get

$$p_s - p_1 = p_1 \times \frac{k}{2} \times \frac{V_1^2 \rho_1}{kp_1} \left[1 + \frac{M_1^2}{4} + \frac{(2-k)}{24} M_1^4 + \ldots\right]$$

$$= \frac{\rho_1 V_1^2}{2} \left[1 + \frac{M_1^2}{4} + \frac{(2-k)}{24} M_1^4 + \ldots\right]$$

The term $\left[1 + \frac{M_1^2}{4} + \frac{(2-k)}{24} M_1^4 + \ldots\right]$ is known as **Compressibility Correction Factor.** And $\frac{\rho_1 V_1^2}{2}$ is the reading of the pitot-static tube. Thus the reading of the pitot- tube must be multiplied by a correction factor given below for correct value of velocity measured by the pitot-tube.

$$\text{Compressibility Correction Factor, C.C.F.} = \left[1 + \frac{M_1^2}{4} + \frac{2-k}{24} M_1^4 + \ldots\right] \quad \ldots(15.33)$$

Problem 15.17 *Calculate the numerical factor by which the actual pressure difference shown by the gauge of a pitot-static tube must be multiplied to allow for compressibility when the value of the Mach number is 0.9. Take k = 1.4.*

Solution. Given :

Mach number, $M_1 = 0.9$

$k = 1.4$

Using equation (15.33), Compressibility Correction Factor is

$$\text{C.C.F.} = \left[1 + \frac{M_1^2}{4} + \frac{2-k}{24} M_1^4 + \ldots\right] = 1 + \frac{.9^2}{4} + \frac{2-1.4}{24} (.9)^4 + \ldots$$

$$= 1.0 + .2025 + .0164 + \ldots\ldots = \mathbf{1.2189.}\ \textbf{Ans.}$$

$\therefore$ Numerical factor by which the actual pressure difference is to be multiplied = 1.2189.

HIGHLIGHTS

1. A flow in which density does not remain constant during flow, is called compressible flow.
2. Equation of state is given by, $\frac{p}{\rho} = RT$

   where p = Absolute pressure in kgf/m^2 or N/m^2
   T = Absolute temperature = 273 + t°C
   R = Gas constant in J/kg K or m/kg
   ρ = Density of gas.

   If the value of p is taken in N/m^2, R is J/kg °K and T in °K, the value of density is given in kg/m^3

3. The pressure density of a gas are related as

$$\frac{p}{\rho} = \text{Constant.... for isothermal process}$$

$$\frac{p}{\rho^k} = \text{Constant....for adiabatic process}$$

where k = Ratio of specific heats = 1.4 for air.

4. Continuity equation for compressible flow is given as ρAV = Constant.

And in differential form, continuity equation is $\frac{d\rho}{\rho} + \frac{dA}{A} + \frac{dV}{V} = 0$.

5. Bernoulli's equation for compressible fluids is given as

$$\frac{p}{\rho g}\log_e p + \frac{V^2}{2g} + Z = \text{Constant.....for isothermal process.}$$

$$\frac{k}{(k-1)}\frac{p}{\rho g} + \frac{V^2}{2g} + Z = \text{Constant.....for adiabatic process.}$$

6. Velocity of sound wave is given by

$$C = \sqrt{\frac{dp}{d\rho}} = \sqrt{\frac{K}{\rho}} \qquad \text{.....in term of Bulk modulus}$$

$$C = \sqrt{\frac{p}{\rho}} = \sqrt{RT} \qquad \text{.....for isothermal process}$$

$$= \sqrt{\frac{kp}{\rho}} = \sqrt{kRT} \qquad \text{.....for adiabatic process.}$$

7. Mach number, M is given as $M = \frac{V}{C}$

If $M < 1$ .....flow is sub-sonic flow,
$M > 1$ .....flow is super-sonic flow,
$M = 1$ .....flow is sonic flow.

8. In sub-sonic flow, the disturbance always moves ahead of the projectile. In sonic flow, the disturbance moves along the projectile while in super-sonic flow, the projectile always moves ahead of the disturbance .

9. Mach angle is given by $\sin\alpha = \frac{C}{V} = \frac{1}{M}$.

10. The pressure, temperature and density at a point where velocity is zero are called stagnation pressure, temperature and stagnation density. Their values are given as

$$p_s = p_1\left[1 + \frac{k-1}{2}M_1^2\right]^{\frac{k}{k-1}}; \rho_s = \rho_1\left[1 + \frac{k-1}{2}M_1^2\right]^{\frac{1}{k-1}}$$

$$T_s = T_1\left[1 + \frac{k-1}{2}M_1^2\right] \text{ also } = \frac{1}{R}\frac{p_s}{\rho_s}.$$

11. Area velocity relationship for compressible fluid is given as $\frac{dA}{A} = \frac{dV}{V}\ [M^2 - 1]$

If $M < 1, \frac{dV}{V} < 0$ and $\frac{dA}{A} > 0$ which means with the increase of area, velocity decreases.

If $M > 1, \frac{dV}{V} > 0$ and $\frac{dA}{A} > 0$ which means with the increase of area, velocity also increases.

If $M = 1, \frac{dA}{A} = 0$ means area is constant.

12. Velocity through a nozzle or orifice fitted to a large tank is

$$V_2 = \sqrt{\frac{2k}{k-1}\frac{p_1}{\rho_1}\left[1 - \left(\frac{p_2}{\rho_1}\right)^{\frac{k-1}{k}}\right]}$$

where p_2 = Pressure at the outlet of nozzle or orifice, p_1 = Pressure in the tank.

13. The mass rate of flow of compressible fluid though orifice or nozzle fitted to the tank is

$$m = A_2\sqrt{\frac{2k}{k-1}p_1\rho_1\left[n^{2/k} - n^{\frac{k+1}{k}}\right]}$$

where n = Pressure ratio = $\frac{p_2}{p_1}$.

14. For maximum flow through orifice or nozzle fitted to the tank, pressure ratio = $\frac{p_2}{p_1} = n = 0.528$

Also $n^{\frac{k-1}{k}} = \frac{2}{k+1}$.

And velocity at the outlet of the orifice or nozzle is $V_2 = \sqrt{\frac{2k}{k+1}\frac{p_1}{\rho_1}} = C_2$.

And mass rate of flow of fluid is given by $m_{max} = 0.685\ A_2\sqrt{p_1\rho_1}$.

15. If the pressure ratio for the nozzle or orifice fitted to the large tank is less than 0.528, the mass rate of flow of the fluid is always corresponding to the pressure ratio of 0.528. But if the pressure ratio is more than 0.528, the mass rate of flow of fluid is corresponding to the given pressure ratio.

16. Mass rate of flow of compressible fluid through venturimeter is given by

$$m = \rho_2 A_2\sqrt{\frac{\frac{2k}{k-1}\frac{p_1}{\rho_1}\left[1 - \left(\frac{p_2}{p_1}\right)^{k-1/k}\right]}{\left[1 - \left(\frac{A_2}{A_1}\right)^2\left(\frac{p_2}{p_1}\right)^{2/k}\right]}}$$

where A_2 = Area at the throat, A_1 = Area at inlet.

17. The compressibility correction factor is given by C.C.F = $\left[1 + \frac{M_1^2}{4} + \frac{2-k}{24}M_1^4 + ...\right]$

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define compressible and incompressible flow.
2. What is the relation between pressure and density of a compressible fluid for
   (*a*) isothermal process and (*b*) adiabatic process ?
3. Derive the continuity equation for one-dimensional compressible flow in differential form.
4. State the Bernoulli's theorem for compressible flow. Derive an expression for Bernoulli's equation when the process is (*i*) isothermal and (*ii*) adiabatic.
5. Write an expression for momentum equation for compressible fluid.
   (*a*) isothermal process and (*b*) adiabatic process ?
6. (*a*) Obtain an expression for velocity of the sound wave in a compressible fluid in terms of change of pressure and change of density.
   (*b*) Show that the velocity of propagation of the pressure wave in a compressible fluid is given by $C = \sqrt{E/\rho}$, where E is volume modulus of elasticity of fluid.
7. Prove that the velocity of sound wave in a compressible fluid is given by $C = \sqrt{\frac{K}{\rho}}$
   where K = Bulk modulus of fluid, ρ = Density of fluid.
8. Derive an expression for the velocity of sound wave for compressible fluid when the process is assumed as (*i*) isothermal and(*ii*) adiabatic.
9. Define Mach number. What is the significance of Mach number in compressible fluid flows ?
10. Define the terms: Sub-sonic flow, super-sonic flow, sonic flow, Mach angle and Mach cone.
11. Show by means of diagrams the nature of propagation of disturbance in compressible flow when Mach number is less than one, is equal to one and is more than one. *(Delhi University, Dec. 2002)*
12. What do you understand by stagnation pressure? Obtain an expression for stagnation pressure of a compressible fluid in terms of approaching Mach number and pressure.
13. Prove that stagnation temperature and stagnation density are given as

$$T_s = T_1\left[1 + \frac{k-1}{2} M_1^2\right] \text{ and } \rho_s = \rho_1\left[1 + \frac{k-1}{2} M_1^2\right]^{\frac{1}{k-1}}$$

14. Derive an expression for area velocity relationship for a compressible fluid in the form.

$$\frac{dA}{A} = \frac{dV}{V}\,[M^2 - 1]$$ *(Delhi University, Dec. 2002)*

15. Find an expression for mass rate of flow of compressible fluid through an orifice or nozzle fitted to a large tank. What is condition for maximum rate of flow?
16. What do you mean by compressibility correction factor ? Find an expression for compressibility factor.
17. Derive an expression for velocity of sound for an adiabatic process.
18. What do you mean by sub-sonic, sonic and super-sonic flows?
19. For frictionless adiabatic flow(*i.e.*, isentropic flow) show that the stagnation pressure at a given point is given by

$$\frac{p_s}{p_1} = 1 + \frac{1}{4} M_0^2 + \frac{(2-k)}{24} M_0^4 + \ldots$$

where p_s = stagnation pressure, p_0 = pressure in ambient flow and $M_0 = U_0/\sqrt{E/\rho}$.

20. Differentiate between isentropic and adiabatic processes.

21. (*a*) Derive an expression for the velocity of sound waves moving in a compressible fluid.
(*b*) Define and explain the terms :
Mach number, Froude number, Reynolds number, Mach cone and Mach angle

22. State the Bernoulli's theorem for compressible flow. Derive an expression for Bernoulli's equation when the process is adiabatic. *(R.G.P.V., Summer, 2002)*

(B) NUMERICAL PROBLEMS

1. A gas is flowing through a horizontal pipe of cross-sectional area of 30 cm^2. At a point the pressure is 30 N per cm^2 (gauge) and temperature 20°C. At another section the area of cross-section is 15 cm^2 and pressure is 25 N/cm^2 gauge. If the mass rate of flow of gas is 0.15 kg/s, find the velocities of the gas at these two sections, assuming an isothermal change. Take R = 287 J/kg K, and atmospheric pressure 10 N/cm^2.
[**Ans.** V_1 = 10.71 m/s ; V_2 = 24.5 m/s]

2. A gas with a velocity of 350 m/s is flowing through a horizontal pipe at a section where pressure is 8 N/cm^2 (absolute) and temperature is 30°C. The pipe changes in diameter and at this section the pressure is 12 N/cm^2 (absolute). Find the velocity of the gas at this section if the flow of the gas is adiabatic. Take R = 287 J/kg K and k = 1.4. [**Ans.** 218.63 m/s]

3. Find the speed of the sound wave in air at sea-level where the pressure and temperature are 9.81 N/cm^2 (abs.) and 20°C respectively. Take R = 287 J/kg K and k = 1.4. [**Ans.** 343.11 m/s]

4. Calculate the Mach number at a point on a jet propelled aircraft which is flying at 900 km/hour at sea-level where air temperature is 15°C. Take k = 1.4 and R = 287 J/kg K. [**Ans.** 0.735]

5. An aeroplane is flying at an height of 20 km, where the temperature is – 40°C. The speed of the plane is corresponding to M = 1.8. Assuming k = 1.4 and R = 287 J/kg K, find the speed of the plane.
[**Ans.** 1982.66 m/hr]

6. A projectile is travelling in air having pressure and temperature as 8.829 N/cm^2 and – 5°C. If the Mach angle is 30°, find the velocity of the projectile. Take k = 1.4 and R = 287 J/kg K. [**Ans.** 656.30 m/s]

7. A projectile travels in air of pressure 8.829 N/cm^2 at –10°C at a speed of 1200 km/hour. Find the Mach number and the Mach angle. Take k = 1.4 and R = 287 J/kg K. [**Ans.** 1.025, θ = 77.2°]

8. Find the Mach number when an aeroplane is flying at 900 km/hour through still air having a pressure of 8.0 N/cm^2 and temperature –15°C. Take k = 1.4 and R = 287 J/kg K. Calculate the pressure, temperature and density of air at the stagnation point on the nose of the plane.
[**Ans.** 0.776, 11.9 N/cm^2 , 16.06°C, 1.434 kg/m^3]

9. Find the velocity of air flowing at the outlet of a nozzle, fitted to a large vessel which contains air at a pressure of 294.3 N/cm^2 (abs.) and at a temperature of 30°C. The pressure at the outlet of the nozzle is 137.34 N/cm^2 (abs.) Take k = 1.4 and R = 287 J/kg K. [**Ans.** 242.98 m/s]

10. A nozzle of diameter 20 mm is fitted to a large tank which contains air at 20°C. The air flows from the tank into atmosphere. For adiabatic flow, find the mass rate of flow of air through the nozzle when pressure of air in tank is (*i*) 5.886 N/cm^2 (gauge) and (*ii*) 29.43 N/cm^2 (gauge). Take k = 1.4 and R = 287 J/kg K and atmospheric pressure = 9.81 N/cm^2. [**Ans.** (*i*) 0.114 kg/s, (*ii*) 0.291 kg/s]

11. Find the mass rate of flow of air through a venturimeter having inlet diameter as 400 mm and throat diameter 200 mm. The pressure at the inlet of the venturimeter is 27.468 N/cm^2 (abs.) and temperature of air at inlet is 20°C. The pressure at the throat is given as 25.506 N/cm^2 absolute. Take k = 1.4 and R = 287 J/kg K. [**Ans.** 11.13 kg/s]

12. Calculate the numerical factor by which the actual pressure difference shown by the gauge of a pitot-tube should be multiplied to allow for compressibility when the value of the Mach number is 0.7. Take k = 1.4.
[**Ans.** 1.1285]

13. Find the Mach number when an aeroplane is flying at 1000 km/hour through still air having pressure of 7 N/cm^2 and temperature of –5°C. Take $R = 287.14$ J/kg K. Calculate the pressure and temperature of air at stagnation point. Take $k = 1.4$. *(Delhi University, Dec. 2002)*

[**Hint.** $V = 1000$ km/hour $= \dfrac{1000 \times 1000}{60 \times 60} = 277.77$ m/s ; $p_1 = 7$ N/m$^2 = 7 \times 10^4$ N/m^2, $t_1 = -5$°C

$\therefore \quad T_1 = -5 + 273 = 268°$ K ; $R = 287.14$ J/kg K, $k = 1.4$

Now $\quad C_1 = \sqrt{kRT_1} = \sqrt{1.4 \times 287.14 \times 268} = 328.2$ m/s $\qquad \therefore M = M_1 = \dfrac{V}{C} = \dfrac{V_1}{C_1} = \dfrac{277.77}{328.2}$

$\therefore \quad M_1 = 0.846 \qquad \therefore p_s = p_1\left[1 + \dfrac{k-1}{2} M_1^2\right]^{\frac{k}{k-1}} = 7 \times 10^4 \left[1 + \dfrac{1.4-1}{2} \times (0.846)^2\right]^{\frac{1.4}{1.4-1}}$

$= 7 \times 10^4 \left[1 + 0.2 \times 0.846^2\right]^{\frac{1.4}{0.4}} = 11.18 \times 10^4$ N/m^2 = **11.18 N/cm^2.**

$T_s = T_1\left[1 + \dfrac{k-1}{2} M_1^2\right] = 268\left[1 + \dfrac{1.4-1}{2} \times (0.846)^2\right] = 306.38°$ K $= 306.38 - 273 =$ **33.38°C.]**

16 CHAPTER

FLOW IN OPEN CHANNELS

▶ 16.1 INTRODUCTION

Flow in open channels is defined as the flow of a liquid with a free surface. A free surface is a surface having constant pressure such as atmospheric pressure. Thus a liquid flowing at atmospheric pressure through a passage is known as flow in open channels. In most of cases, the liquid is taken as water. Hence flow of water through a passage under atmospheric pressure is called flow in open channels. The flow of water through pipes at atmospheric pressure or when the level of water in the pipe is below the top of the pipe, is also classified as open channel flow.

In case of open channel flow, as the pressure is atmospheric, the flow takes place under the force of gravity which means the flow takes place due to the slope of the bed of the channel only. The hydraulic gradient line coincides with the free surface of water.

▶ 16.2 CLASSIFICATION OF FLOW IN CHANNELS

The flow in open channel is classified into the following types :

1. Steady flow and unsteady flow,
2. Uniform flow and non-uniform flow,
3. Laminar flow and turbulent flow, and
4. Sub-critical, critical and super critical flow.

16.2.1 Steady Flow and Unsteady Flow. If the flow characteristics such as depth of flow, velocity of flow, rate of flow at any point in open channel flow do not change with respect to time, the flow is said to be steady flow. Mathematically, steady flow is expressed as

$$\frac{\partial V}{\partial t} = 0, \frac{\partial Q}{\partial t} = 0 \quad \text{or} \quad \frac{\partial y}{\partial t} = 0 \qquad \ldots(16.1)$$

where V = velocity, Q = rate of flow and y = depth of flow.

If at any point in open channel flow, the velocity of flow, depth of flow or rate of flow changes with respect to time, the flow is said to be unsteady flow. Mathematically, unsteady flow means

$$\frac{\partial V}{\partial t} \neq 0 \quad \text{or} \quad \frac{\partial y}{\partial t} \neq 0 \quad \text{or} \quad \frac{\partial Q}{\partial t} \neq 0.$$

16.2.2 Uniform Flow and Non-uniform Flow. If for a given length of the channel, the velocity of flow, depth of flow, slope of the channel and cross-section remain constant, the flow is

said to be uniform. On the other hand, if for a given length of the channel, the velocity of flow, depth of flow etc., do not remain constant, the flow is said to be non-uniform flow. Mathematically, uniform and non-uniform flow are written as :

$$\frac{\partial y}{\partial S} = 0, \frac{\partial V}{\partial S} = 0 \text{ for uniform flow}$$

and

$$\frac{\partial y}{\partial S} \neq 0, \frac{\partial V}{\partial S} \neq 0 \text{ for non-uniform flow.}$$

Non-uniform flow in open channels is also called varied flow, which is classified in the following two types as :

(*i*) Rapidly Varied Flow (R.V.F.), and

(*ii*) Gradually Varied Flow (G.V.F.).

(*i*) **Rapidly Varied Flow (R.V.F.).** Rapidly varied flow is defined as that flow in which depth of flow changes abruptly over a small length of the channel. As shown in Fig. 16.1 when there is any obstruction in the path of flow of water, the level of water rises above the obstruction and then falls and again rises over a small length of channel. Thus the depth of flow changes rapidly over a short length of the channel. For this short length of the channel the flow is called rapidly varied flow (R.V.F.).

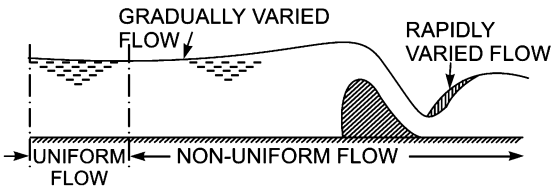


Fig. 16.1 *Uniform and non-uniform flow.*

(*ii*) **Gradually Varied Flow (G.V.F.).** If the depth of flow in a channel changes gradually over a long length of the channel, the flow is said to be gradually varied flow and is denoted by G.V.F.

16.2.3 Laminar Flow and Turbulent Flow. The flow in open channel is said to be laminar if the Reynold number (R_e) is less than 500 or 600. Reynold number in case of open channels is defined as :

$$R_e = \frac{\rho VR}{\mu} \qquad ...(16.2)$$

where V = Mean velocity of flow of water

R = Hydraulic radius or Hydraulic mean depth

$$= \frac{\text{Cross-section area of flow normal to the direction of flow}}{\text{Wetted perimeter}}$$

ρ and μ = Density and viscosity of water.

If the Reynold number is more than 2000, the flow is said to be turbulent in open channel flow. If R_e lies between 500 to 2000, the flow is considered to be in transition state.

16.2.4 Sub-critical, Critical and Super Critical Flow. The flow in open channel is said to be sub-critical if the Froude number (F_e) is less than 1.0. The Froude number is defined as :

$$F_e = \frac{V}{\sqrt{gD}} \qquad ...(16.3)$$

where V = Mean velocity of flow

D = Hydraulic depth of channel and is equal to the ratio of wetted area to the top width of channel

$= \frac{A}{T}$, where T = Top width of channel.

Sub-critical flow is also called tranquil or streaming flow. For sub-critical flow, $F_e < 1.0$.

The flow is called critical if $F_e = 1.0$. And if $F_e > 1.0$, the flow is called super critical or shooting or rapid or torrential.

▶ 16.3 DISCHARGE THROUGH OPEN CHANNEL BY CHEZY'S FORMULA

Consider uniform flow of water in a channel as shown in Fig. 16.2. As the flow is uniform, it means the velocity, depth of flow and area of flow will be constant for a given length of the channel. Consider sections 1-1 and 2-2.

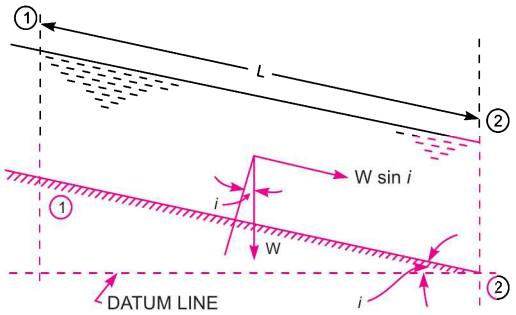


Fig. 16.2 *Uniform flow in open channel.*

Let L = Length of channel,

A = Area of flow of water,

i = Slope of the bed,

V = Mean velocity of flow of water,

P = Wetted perimeter of the cross-section,

f = Frictional resistance per unit velocity per unit area.

The weight of water between sections 1-1 and 2-2.

W = Specific weight of water × volume of water

$= w \times A \times L$

Component of W along direction of flow $= W \times \sin i = wAL \sin i$ ...(*i*)

Frictional resistance against motion of water = $f \times$ surface area × (velocity $)^n$

The value of n is found experimentally equal to 2 and surface area = $P \times L$

$\therefore$ Frictional resistance against motion = $f \times P \times L \times V^2$ ...(*ii*)

The forces acting on the water between sections 1-1 and 2-2 are:

1. Component of weight of water along the direction of flow,
2. Friction resistance against flow of water,
3. Pressure force at section 1-1,
4. Pressure force at section 2-2.

As the depths of water at the sections 1-1 and 2-2 are the same, the pressure forces on these two sections are same and acting in the opposite direction. Hence they cancel each other. In case of uniform flow, the velocity of flow is constant for the given length of the channel. Hence there is no acceleration acting on the water. Hence the resultant force acting in the direction of flow must be zero.

∴ Resolving all forces in the direction of flow, we get

$$wAL \sin i - f \times P \times L \times V^2 = 0$$

or $$wAL \sin i = f \times P \times L \times V^2$$

$$V^2 = \frac{wAL \sin i}{f \times P \times L} = \frac{w}{f} \times \frac{A}{P} \times \sin i$$

or $$V = \sqrt{\frac{w}{f}} \times \sqrt{\frac{A}{P} \times \sin i} \qquad ...(iii)$$

But $$\frac{A}{P} = m$$

= hydraulic mean depth or hydraulic radius,

$$\sqrt{\frac{w}{f}} = C = \text{Chezy's constant}$$

Substituting the values of $\frac{A}{P}$ and $\sqrt{\frac{w}{f}}$ in equation (*iii*), $V = C\sqrt{m \sin i}$

For small values of i, $\sin i \simeq \tan i \approx i \quad \therefore \quad V = C\sqrt{mi}$ ...(16.4)

∴ Discharge, Q = Area × Velocity = $A \times V$

$$= A \times C\sqrt{mi} \qquad ...(16.5)$$

Problem 16.1 *Find the velocity of flow and rate of flow of water through a rectangular channel of 6 m wide and 3 m deep, when it is running full. The channel is having bed slope as 1 in 2000. Take Chezy's constant C = 55.*

Solution. Given :

Width of rectangular channel, $b = 6$ m

Depth of channel, $d = 3$ m

∴ Area, $A = 6 \times 3 = 18 \text{ m}^2$

Bed slope, $i = 1 \text{ in } 2000 = \frac{1}{2000}$

Chezy's constant, $C = 55$

Perimeter, $P = b + 2d = 6 + 2 \times 3 = 12$ m

∴ Hydraulic mean depth, $m = \frac{A}{P} = \frac{18}{12} = 1.5$ m

Velocity of flow is given by equation (16.4) as,

$$V = C\sqrt{mi} = 55\sqrt{1.5 \times \frac{1}{2000}} = \mathbf{1.506 \text{ m/s. Ans.}}$$

Rate of flow, $Q = V \times \text{Area} = V \times A = 1.506 \times 18 = \mathbf{27.108 \text{ m}^3\text{/s. Ans.}}$

Problem 16.2 *Find the slope of the bed of a rectangular channel of width 5 m when depth of water is 2 m and rate of flow is given as 20 m^3/s. Take Chezy's constant, C = 50.*

Solution. Given :

Width of channel, $b = 5$ m

Depth of water, $d = 2$ m

Rate of flow, $Q = 20 \text{ m}^3\text{/s}$

Chezy's constant $C = 50$

Let the bed slope $= i$

Using equation (16.5), we have $Q = AC\sqrt{mi}$

where A = Area = $b \times d = 5 \times 2 = 10\ m^2$

$$m = \frac{A}{P} = \frac{10}{b+2d} = \frac{10}{5+2\times 2} = \frac{10}{5+4} = \frac{10}{9}\ m$$

$$\therefore \qquad 20.0 = 10 \times 50 \times \sqrt{\frac{10}{9} \times i} \quad \text{or} \quad \sqrt{\frac{10}{9}}\, i = \frac{20.0}{500} = \frac{2}{50}$$

Squaring both sides, we have $\frac{10}{9} i = \frac{4}{2500}$

$$\therefore \qquad i = \frac{4}{2500} \times \frac{9}{10} = \frac{36}{25000} = \frac{1}{\frac{25000}{36}} = \frac{1}{\mathbf{694.44}}\ \textbf{. Ans.}$$

$\therefore$ Bed slope is 1 in 694.44.

Problem 16.3 *A flow of water of 100 litres per second flows down in a rectangular flume of width 600 mm and having adjustable bottom slope. If Chezy's constant C is 56, find the bottom slope necessary for uniform flow with a depth of flow of 300 mm. Also find the conveyance K of the flume.*

Solution. Given :

Discharge, $Q = 100$ litres/s $= \frac{100}{1000} = 0.10\ m^3/s$

Width of channel, $b = 600$ mm $= 0.60$ m

Depth of flow, $d = 300$ mm $= 0.30$ m

$\therefore$ Area of flow, $A = b \times d = 0.6 \times 0.3 = 0.18\ m^2$

Chezy's constant, $C = 56$

Let the slope of bed $= i$

Hydraulic mean depth, $m = \frac{A}{P} = \frac{0.18}{b+2d} = \frac{0.18}{0.6+2\times 0.30} = \frac{0.18}{1.2} = 0.15$ m

Using equation (16.5), we have $Q = AC\sqrt{mi}$

or $$0.10 = 0.18 \times 56 \times \sqrt{0.15 \times i} \quad \text{or} \quad \sqrt{0.15i} = \frac{0.10}{0.18 \times 56}$$

Squaring both sides, we have $0.15\, i = \left(\frac{0.10}{0.18 \times 56}\right)^2 = .000098418$

$$\therefore \qquad i = \frac{.000098418}{0.15} = .0006512 = \frac{1}{\frac{1}{.0006512}} = \frac{1}{\mathbf{1524}}\ \textbf{. Ans.}$$

$\therefore$ Slope of the bed is 1 in 1524.

Conveyance K of the channel

Equation (16.5) is given as $Q = AC\sqrt{mi}$

which can be written as $Q = K\sqrt{i}$

where $K = AC\sqrt{m}$ and K is called conveyance of the channel section.

$\therefore$ $K = AC\sqrt{m} = 0.18 \times 56 \times \sqrt{0.15} = \mathbf{3.9039\ m^3/s.\ Ans.}$

Problem 16.4 *Find the discharge through a trapezoidal channel of width 8 m and side slope of 1 horizontal to 3 vertical. The depth of flow of water is 2.4 m and value of Chezy's constant, C = 50. The slope of the bed of the channel is given 1 in 4000.*

Solution. Given :

Width, $b = 8$ m

Side slope = 1 hor. to 3 vertical

Depth, $d = 2.4$ m

Chezy's constant, $C = 50$

Bed slope, $i = \dfrac{1}{4000}$

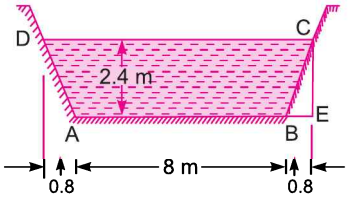


Fig. 16.3

From Fig. 16.3 when depth , $CE = 2.4$,

the horizontal distance $BE = 2.4 \times \dfrac{1}{3} = 0.8$ m

$\therefore$ Top width of the channel,

$CD = AB + 2 \times BE = 8.0 + 2 \times 0.8 = 9.6$ m

$\therefore$ Area of trapezoidal channel, $ABCD$ is given as,

$$A = (AB + CD) \times \frac{CE}{2} = (8 + 9.6) \times \frac{2.4}{2} = 17.6 \times 1.2 = 21.12\ \text{m}^2$$

Wetted perimeter , $P = AB + BC + AD = AB + 2BC$ ($\because$ $BC = AD$)

But $BC = \sqrt{BE^2 + CE^2} = \sqrt{(0.8)^2 + (2.4)^2} = 2.529$ m

$\therefore$ $P = 8.0 + 2 \times 2.529 = 13.058$ m

Hydraulic mean depth, $m = \dfrac{A}{P} = \dfrac{12.12}{13.058} = 1.617$ m

The discharge, Q is given by equation (16.5) as

$$Q = AC\sqrt{mi} = 21.12 \times 50\sqrt{1.617 \times \frac{1}{4000}} = \mathbf{21.23\ m^3/s.\ Ans.}$$

Problem 16.5 *Find the bed slope of trapezoidal channel of bed width 6 m, depth of water 3 m and side slope of 3 horizontal to 4 vertical, when the discharge through the channel is 30 m^3/s. Take Chezy's constant, C = 70.*

Solution. Given :

Bed width, $b = 6.0$ m

Depth of flow, $d = 3.0$ m

Side slope = 3 horizontal to 4 vertical

Discharge, $Q = 30\ \text{m}^3/\text{s}$

Chezy's constant, $C = 70$

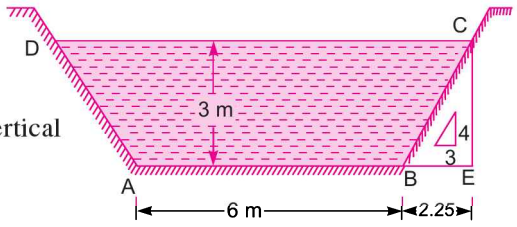


Fig. 16.4

From Fig. 16.4, for depth of flow

= 3 m = CE

Distance, $BE = 3 \times \dfrac{3}{4} = \dfrac{9}{4} = 2.25$ m

$\therefore$ Top width, $CD = AB + 2 \times BE = 6.0 + 2 \times 2.25 = 10.50$ m

Wetted perimeter, $P = AD + AB + BC = AP + 2BC$ ($\because BC = AD$)

$$= AB + 2\sqrt{BE^2 + CE^2} = 6.0 + 2\sqrt{(2.25)^2 + (3)^2} = 13.5 \text{ m}$$

Area of flow, A = Area of trapezoidal $ABCD$

$$= \frac{(AB + CD) \times CE}{2} = \frac{(6 + 10.50)}{2} \times 3.0 = 24.75 \text{ m}^2$$

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{24.75}{13.50} = 1.833$

Using equation (16.5), $Q = AC\sqrt{mi}$

or $$30.0 = 24.75 \times 70 \times \sqrt{1.833 \times i} = 2345.6\sqrt{i}$$

$$i = \left(\frac{30}{2345.6}\right)^2 = \frac{1}{\left(\frac{2345.6}{30}\right)^2} = \mathbf{\frac{1}{6133}}. \textbf{ Ans.}$$

Problem 16.6 *Find the discharge of water through the channel shown in Fig. 16.5. Take the value of Chezy's constant = 60 and slope of the bed as 1 in 2000.*

Solution. Given :

Chezy's constant, $C = 60$

Bed slope, $i = \frac{1}{2000}$

From Fig.16.5, Area, A = Area $ABCD$ + Area BEC

$$= (1.2 \times 3.0) + \frac{\pi R^2}{2}$$

$$= 3.6 + \frac{(1.5)^2}{2} = 7.134 \text{ m}^2$$

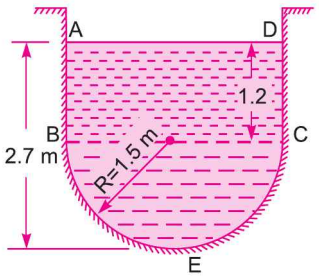


Fig. 16.5

Wetted perimeter, $P = AB + BEC + CD$

$$= 1.2 + \pi R + 1.2 = 1.2 + \pi \times 1.5 + 1.2 = 7.1124 \text{ m}$$

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{7.134}{7.1124} = 1.003$

The discharge, Q is given by equation (16.5) as

$$Q = AC\sqrt{mi}$$

$$= 7.134 \times 60 \times \sqrt{1.003 \times \frac{1}{2000}} = \mathbf{9.585 \text{ m}^3\text{/s. Ans.}}$$

Problem 16.7 *Find the rate of flow of water through a V-shaped channel as shown in Fig. 16.6. Take the value of C = 55 and slope of the bed 1 in 2000.*

Solution. Given :

Chezy's constant, $C = 55$

Bed slope, $i = \frac{1}{1000}$

Depth of flow, $d = 4.0$ m

Angle made by each side with vertical,

i.e., $\angle ABD = \angle CBD = 30°$

From Fig. 16.6, we have

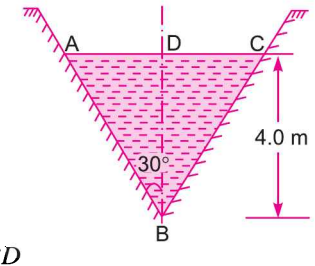


Fig. 16.6

Area, A = Area of ABC

$$= 2 \times \text{Area } ABD = \frac{2 \times AD \times BD}{2} = AD \times BD$$

$$= BD \tan 30° \times BD \qquad \left(\because \tan 30° = \frac{AD}{BD},\ AD = BD \tan 30°\right)$$

$$= 4 \tan 30° \times 4 = 9.2376 \text{ m}^2$$

Wetted perimeter, $P = AB + BC = 2AB \qquad (\because AB = BC)$

$$= 2\sqrt{BD^2 + AD^2} = 2\sqrt{4.0^2 + (4 \tan 30°)^2}$$

$$= 2\sqrt{16.0 + 5.333} = 9.2375 \text{ m}.$$

$\therefore$ Hydraulic mean depth, $m = \dfrac{A}{P} = \dfrac{9.2376}{9.2375} = 1.0$ m

Using equation (16.5) for discharge,

$$Q = AC\sqrt{mi} = 9.2376 \times 55 \times \sqrt{1 \times \frac{1}{1000}} = \mathbf{16.066\ m^3/s.\ Ans.}$$

▶ 16.4 EMPIRICAL FORMULAE FOR THE VALUE OF CHEZY'S CONSTANT

Equation (16.4) is known Chezy's formula after the name of a French Engineer, Antoine Chezy who developed this formula in 1975. In this equation C is known as Chezy's constant, which is not a dimensionless co-efficient. The dimension of C is

$$= \frac{V}{\sqrt{mi}} = \frac{L/T}{\sqrt{\frac{A}{P} i}} = \frac{L/T}{\sqrt{\frac{L^2}{L} i}} = \frac{L}{T\sqrt{Li}} = \frac{\sqrt{L}}{T}$$

$$= L^{1/2}T^{-1} \qquad \{ i \text{ is dimensionless}\}$$

Hence the value of C depends upon the system of units. The following are the empirical formulae, after the name of their inventors, used to determine the value of C:

1. Bazin formula (In MKS units) : $C = \dfrac{157.6}{1.81 + \dfrac{K}{\sqrt{m}}}$ ...(16.6)

where K = Bazin's constant and depends upon the roughness of the surface of channel, whose values are given in Table 16.1.

m = Hydraulic mean depth or hydraulic radius.

2. Ganguillet-Kutter Formula. The value of C is given in MKS unit as

$$C = \frac{23 + \frac{0.00155}{i} + \frac{1}{N}}{1 + \left(23 + \frac{0.00155}{i}\right)\frac{N}{\sqrt{m}}} \quad ...(16.7)$$

where N = Roughness co-efficient which is known as Kutter's constant, whose value for different surfaces are given in Table 16.2

i = Slope of the bed

m = Hydraulic mean depth.

Table 16.1 Values of K in the Bazin's Formula

S. No.	*Nature of Channel inside surface*	*Value of K*
1.	Smooth cemented or planned wood	0.11
2.	Brick or concrete or unplanned wood	0.21
3.	Rubble masonary or Ashlar or poor brick work	0.83
4.	Earthen channel of very good surface	1.54
5.	Earthen channel of ordinary surface	2.36
6.	Earthen channel of rough surface	3.17

Table 16.2 Value of N in the Ganguillet-Kutter Formula

S. No.	*Nature of Channel inside surface*	*Value of N*
1.	Very smooth surface of glass, plastic or brass	0.010
2.	Smooth surface of concrete	0.012
3.	Rubble masonary or poor brick work	0.017
4.	Earthen channels neatly excavated	0.018
5.	Earthen channels of ordinary surface	0.027
6.	Earthen channels of rough surface	0.030
7.	Natural streams, clean and straight	0.030
8.	Natural streams with weeds, duppools etc.	0.075 to .15

3. Manning's Formula. The value of C according to this formula is given as

$$C = \frac{1}{N} m^{1/6} \quad ...(16.8)$$

where m = Hydraulic mean depth

N = Manning's constant which is having same value as Kutter's constant for the normal range of slope and hydraulic mean depth. The values of N are given in Table 16.2.

Problem 16.8 *Find the discharge through a rectangular channel 2.5 m wide, having depth of water 1.5 m and bed slope as 1 in 2000. Take the value of k = 2.36 in Bazin's formula.*

Solution. Given :

Width of channel, $b = 2.5$ m

Depth of flow, $d = 1.5$ m

$\therefore$ Area, $A = b \times d = 2.5 \times 1.5 = 3.75$ m^2

Wetted Perimeter, $P = d + b + d = 1.5 + 2.5 + 1.5 = 5.5$ m

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{3.75}{5.50} = 0.682$

Bed slope, $i = \frac{1}{2000}$

Bazin's constant, $K = 2.36$

Using Bazin's formula given by equation (16.6), as

$$C = \frac{157.6}{1.81 + \frac{K}{\sqrt{m}}} = \frac{157.6}{1.81 + \frac{2.36}{\sqrt{0.682}}} = 33.76$$

Discharge, Q is given by equation (16.5), as

$$Q = AC\sqrt{mi}$$

$$= 3.75 \times 33.76 \times \sqrt{0.682 \times \frac{1}{2000}} = \mathbf{2.337\ m^3/s.\ Ans.}$$

Problem 16.9 *Find the discharge through a rectangular channel 14 m wide, having depth of water 3 m and bed slope 1 in 1500. Take the value of N = 0.03 in the Kutter's formula.*

Solution. Given :

Width of channel, $b = 4$ m

Depth of water, $d = 3$ m

Bed slope, $i = \frac{1}{1500} = 0.000667$

Kutter's constant, $N = 0.03$

Area of flow, $A = b \times d = 4 \times 3 = 12\ m^2$

Wetted perimeter, $P = d + b + d = 3 + 4 + 3 = 10$ m

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{.12}{10} = 1.2$ m

Using Kutter's formula given by equation (16.7), as

$$C = \frac{23 + \frac{.00155}{i} + \frac{1}{N}}{1 + \left(23 + \frac{.00155}{i}\right) \times \frac{N}{\sqrt{m}}} = \frac{23 + \frac{.00155}{.000667} + \frac{1}{.03}}{1 + \left(23 + \frac{.00155}{.000667}\right) \times \frac{.03}{\sqrt{1.20}}}$$

$$= \frac{23 + 2.3238 + 33.33}{1 + (23 + 2.3238) \times .03286} = \frac{58.633}{1.832} = 32.01$$

Discharge, Q is given by equation (16.5), as

$$Q = AC\sqrt{mi} = 12 \times 32.01 \times \sqrt{12 \times .000667} = \mathbf{10.867\ m^3/s.\ Ans.}$$

Problem 16.10 *Find the discharge through a rectangular channel of width 2 m, having a bed slope of 4 in 8000. The depth of flow is 1.5 m and take the value of N in Manning's formula as 0.012.*

Solution. Given :

Width of the channel, $b = 2$ m

Depth of the flow, $d = 1.5$ m

$\therefore$ Area of flow, $A = b \times d = 2 \times 1.5 = 3.0\ m^2$

Wetted perimeter, $P = b + d + d = 2 + 1.5 + 1.5 = 5.0$ m

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{3.0}{5.0} = 0.6$

Bed slope , $i = 4 \text{ in } 8000 = \frac{4}{8000} = \frac{1}{2000}$

Value of N $= 0.012$

Using Manning's formula, given by equation (16.8), as

$$C = \frac{1}{N} m^{1/6} = \frac{1}{0.012} \times 0.6^{1/6} = 76.54$$

Discharge, Q is given by equation (16.5), as

$$Q = AC\sqrt{mi}$$

$$= 3.0 \times 76.54 \sqrt{0.6 \times \frac{1}{2000}} \text{ m}^2/\text{s} = \mathbf{3.977\ m^3/s.\ Ans.}$$

Problem 16.11 *Find the bed slope of trapezoidal channel of bed width 4 m, depth of water 3 m and side slope of 2 horizontal to 3 vertical, when the discharge through the channel is 20 m^3/s.*

Take Manning's $N = 0.03$ in Manning's formula $C = \frac{1}{N} m^{1/6}$.

Solution. Given :

Bed width, $b = 4$ cm

Depth of flow, $d = 3$ m

Side slope $= 2$ hor. to 3 vert.

Discharge, $Q = 20.0 \text{ m}^3/\text{s}$

Manning's, $N = 0.03$

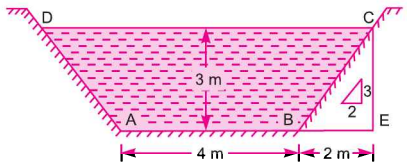


Fig. 16.7

From Fig. 16.7, we have

Distance, $BE = d \times \frac{2}{3} = 3 \times \frac{2}{3} = 2$ m

$\therefore$ Top width, $CD = AB + 2BE$

$= 4 + 2 \times 2 = 8.0$ m

$\therefore$ Area of flow, $A =$ Area of trapezoidal section $ABCD$

$$= \frac{(AB + CD)}{2} \times d = \frac{(4 + 8)}{2} \times 3 = 18 \text{ m}^2$$

Wetted perimeter, $P = AD + AB + BC = AB + 2BC$ ($\because$ $AD = BC$)

$$= 4.0 + 2\sqrt{BE^2 + EC^2} = 4.0 + 2\sqrt{2^2 + 3^2} = 4.0 + 2 \times \sqrt{13} = 11.21 \text{ m}$$

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{18}{11.21} = 1.6057$

Using Manning's formula, $C = \frac{1}{N} m^{1/6} = \frac{1}{0.03} \times (1.6057)^{1/6} = 36.07$

Using equation (16.5) for discharge,

$$Q = AC\sqrt{mi} = 18 \times 36.07 \times \sqrt{1.6057 \times i} \quad \text{or } 20.0 = 822.71\sqrt{i}$$

$$\therefore \quad i = \left(\frac{20.0}{822.71}\right)^2 = 0.0005909 = \frac{\mathbf{1}}{\mathbf{1692}}. \textbf{ Ans.}$$

Problem 16.12 *Find the diameter of a circular sewer pipe which is laid at a slope of 1 in 8000 and carries a discharge of 800 litres/s when flowing half full. Take the value of Manning's N = 0.020.*

Solution. Given :

Slope of pipe, $i = \frac{1}{8000}$

Discharge, $Q = 800$ litres/s $= 0.8$ m^3/s

Manning's, $N = 0.020$

Let the dia. of sewer pipe, $= D$

Depth of flow, $d = \frac{D}{2}$

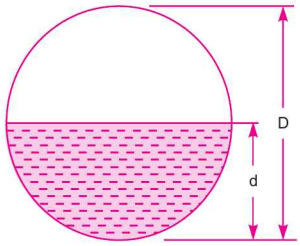


Fig. 16.8

$\therefore$ Area of flow, $A = \frac{\pi}{4}D^2 \times \frac{1}{2} = \frac{\pi D^2}{8}$

Wetted perimeter, $P = \frac{\pi D}{2}$

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{\frac{\pi D^2}{8}}{\pi D/2} = \frac{D}{4}$

Using Manning's formula given by equation (16.8), $C = \frac{1}{N} m^{1/6}$

The discharge, Q through pipe is given by equation (16.6), as

$$Q = AC\sqrt{mi}$$

$$= \frac{\pi D^2}{8} \times \frac{1}{N} m^{1/6}\sqrt{mi}$$

or $$0.80 = \frac{\pi}{8}D^2 \times \frac{1}{.020} \times m^{1/6} \times m^{1/2} \times \sqrt{i}$$

$$= \frac{\pi}{8}D^2 \times \frac{1}{.020} m^{(1/6 + 1/2)} \times \sqrt{\frac{1}{8000}} = \frac{\pi}{8}D^2 \times \frac{1}{.020} \times m^{2/3} \times 0.01118$$

$$= 0.2195 \times D^2 \times \left(\frac{D}{4}\right)^{2/3} \qquad \left(\because \quad m = \frac{D}{4}\right)$$

$$= \frac{.2195}{4^{2/3}} \times D^2 \times D^{2/3} = 0.0871\, D^{8/3}$$

or $$D^{8/3} = \frac{0.80}{.0871} = 9.1848$$

$\therefore$ $$D = (9.1848)^{3/8} = (9.1848)^{0.375} = \textbf{2.296 m. Ans.}$$

▶ 16.5 MOST ECONOMICAL SECTION OF CHANNELS

A section of a channel is said to be most economical when the cost of construction of the channel is minimum. But the cost of construction of a channel depends upon the excavation and the lining. To keep the cost down or minimum, the wetted perimeter, for a given discharge, should be minimum. This condition is utilized for determining the dimensions of a economical sections of different form of channels.

Most economical section is also called the best section or most efficient section as the discharge, passing through a most economical section of channel for a given cross-sectional area (A), slope of the bed (i) and a resistance co-efficient, is maximum. But the discharge, Q is given by equation (16.5) as

$$Q = AC\sqrt{mi} = AC\sqrt{\frac{A \times i}{P}} \qquad \left(\because \quad m = \frac{A}{P}\right)$$

For a given A, i and resistance co-efficient C, the above equation is written as

$$Q = K\frac{1}{\sqrt{P}}, \quad \text{where } K = AC\sqrt{Ai} = \text{constant}$$

Hence the discharge, Q will be maximum, when the wetted perimeter P is minimum. This condition will be used for determining the best section of a channel *i.e.*, best dimensions of a channel for a given area.

The conditions to be most economical for the following shapes of the channels will be considered :

1. Rectangular Channel, 2. Trapezoidal Channel, and 3. Circular Channel.

16.5.1 Most Economical Rectangular Channel. The condition for most economical section, is that for a given area, the perimeter should be minimum. Consider a rectangular channel as shown in Fig. 16.9

Let b = width of channel,

d = depth of the flow,

$\therefore$ Area of flow, $A = b \times d$ ...(*i*)

Wetted perimeter, $P = d + b + d = b + 2d$ ...(*ii*)

From equation (*i*), $b = \dfrac{A}{d}$

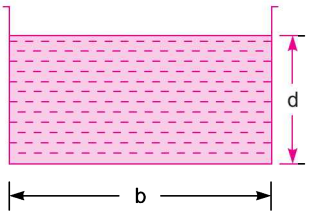


Fig. 16.9 *Rectangular channel.*

Substituting the value of b in (*ii*),

$$P = b + 2d = \frac{A}{d} + 2d \qquad ...(iii)$$

For most economical section, P should be minimum for a given area.

or $$\frac{dP}{d(d)} = 0$$

Differentiating the equation (*iii*) with respect to d and equating the same to zero, we get

$$\frac{d}{d(d)}\left[\frac{A}{d} + 2d\right] = 0 \quad \text{or} \quad -\frac{A}{d^2} + 2 = 0 \quad \text{or} \quad A = 2d^2$$

But $A = b \times d$, $\therefore$ $b \times d = 2d^2$ or $b = 2d$ ...(16.9)

Now hydraulic mean depth, $m = \dfrac{A}{P} = \dfrac{b \times d}{b + 2d}$ $(\because \quad A = bd,\ P = b + 2d)$

$$= \frac{2d \times d}{2d + 2d} \qquad (\because \quad b = 2d)$$

$$= \frac{2d^2}{4d} = \frac{d}{2} \qquad \ldots(16.10)$$

From equations (16.9) and (16.10), it is clear that rectangular channel will be most economical when:
(*i*) Either $b = 2d$ means width is two times depth of flow.

(*ii*) Or $m = \frac{d}{2}$ means hydraulic depth is half the depth of flow.

Problem 16.13 *A rectangular channel of width, 4 m is having a bed slope of 1 in 1500. Find the maximum discharge through the channel. Take value of C = 50.*

Solution. Given :

Width of channel, $b = 4$ m

Bed slope, $i = \frac{1}{1500}$

Chezy's constant, $C = 50$

Discharge will be maximum, when the channel is most economical. The conditions for most economical rectangular channel are :

(*i*) $b = 2d$ or $d = \frac{b}{2} = \frac{4}{2} = 2.0$ m

(*ii*) $m = \frac{d}{2} = \frac{2}{2} = 1.0$ m

$\therefore$ Area of most economical rectangular channel, $A = b \times d = 4.0 \times 2.0 = 8$ m^2

Using equation (16.5) for discharge as

$$Q = AC\sqrt{mi} = 8.0 \times 50 \times \sqrt{1.0 \times \frac{1}{1500}} = \mathbf{10.328\ m^3/s.\ Ans.}$$

Problem 16.14 *A rectangular channel carries water at the rate of 400 litres/s when bed slope is 1 in 2000. Find the most economical dimensions of the channel if C = 50.*

Solution. Given :

Discharge, $Q = 400$ litres/s $= 0.4$ m^3/s

Bed slope, $i = \frac{1}{2000}$

Chezy's constant, $C = 50$

For the rectangular channel to be most economical,

(*i*) Width, $b = 2d$

(*ii*) Hydraulic mean depth, $m = \frac{d}{2}$

$\therefore$ Area of flow, $A = b \times d = 2d \times d = 2d^2$

Using equation (16.5) for discharge,

$$Q = AC\sqrt{mi}$$

or $$0.4 = 2d^2 \times 50 \times \sqrt{\frac{d}{2} \times \frac{1}{2000}} = 2 \times 50 \times \sqrt{\frac{1}{2 \times 2000}}\, d^{5/2} = 1.581\, d^{5/2}$$

$$\therefore \quad d^{5/2} = \frac{0.4}{1.581} = 0.253$$

$$\therefore \quad d = (.253)^{2/5} = \mathbf{0.577\ m.\ Ans.}$$

$$b = 2d = 2 \times .577 = \mathbf{1.154\ m.\ Ans.}$$

Problem 16.15 *A rectangular channel 4 m wide has depth of water 1.5 m. The slope of the bed of the channel is 1 in 1000 and value of Chezy's constant C = 55. It is desired to increase the discharge to a maximum by changing the dimensions of the section for constant area of cross-section, slope of the bed and roughness of the channel. Find the new dimensions of the channel and increase in discharge.*

Solution. Given :

Width of channel, $b = 4.0$ m

Depth of flow, $d = 1.5$ m

$\therefore$ Area of flow, $A = b \times d = 4 \times 1.5 = 6.0\ \text{m}^2$

Slope of bed, $i = \frac{1}{1000}$

Chezy's constant, $C = 55$

Wetted perimeter, $P = d + b + d = 1.5 + 4 + 1.5 = 7.0$ m

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{4.0}{7.0} = 0.857$

The discharge, Q is given by $Q = AC\sqrt{mi} = 6.0 \times 55\sqrt{0.857 \times \frac{1}{1000}} = 9.66\ \text{m}^3/\text{s}$ ...(*i*)

For maximum discharge for a given area, slope of bed and roughness we proceed as :

Let b' = new width of channel

d' = new depth of flow

Then, Area, $A = b' \times d'$, where A = constant = 6.0 m^2

$\therefore \quad b' \times d' = 6.0$ ...(*ii*)

Also for maximum discharge $b' = 2d'$ ...(*iii*)

Substituting the value of b' in equation (*ii*), we have

$$2d' \times d' = 6.0 \text{ or } d'^2 = \frac{6.0}{2} = 3.0$$

$$\therefore \quad d' = \sqrt{3} = 1.732$$

Substituting the value of d' in (*iii*), we get

$$b' = 2 \times 1.732 = 3.464$$

$\therefore$ New dimensions of the channel are

Width, $b' = \mathbf{3.464\ m.\ Ans.}$

Depth, $d' = \mathbf{1.732\ m.\ Ans.}$

Wetted perimeter, $P' = d' + b' + d' = 1.732 + 3.464 + 1.732 = 6.928$

$\therefore$ Hydraulic mean depth, $m' = \frac{A}{P'} = \frac{6.0}{6.928} = 0.866$ m

(New hydraulic mean depth, m' corresponds to the condition of maximum discharge. And hence also equal to

$$\frac{d'}{2} = \frac{1.732}{2} = 0.866 \text{ m})$$

Max. discharge, Q', is given by $Q' = AC\sqrt{m'i} = 6.0 \times 55 \times \sqrt{0.866 \times \frac{1}{1000}} = 9.71 \text{ m}^3/\text{s}$ ...(*iv*)

∴ Increase in discharge $= Q' - Q = 9.71 - 9.66 =$ **0.05 m³/s. Ans.**

16.5.2 Most Economical Trapezoidal Channel. The trapezoidal section of a channel will be most economical, when its wetted perimeter is minimum. Consider a trapezoidal section of channel as shown in Fig. 16.10.

Let b = width of channel at bottom,
d = depth of flow,
θ = angle made by the sides with horizontal,

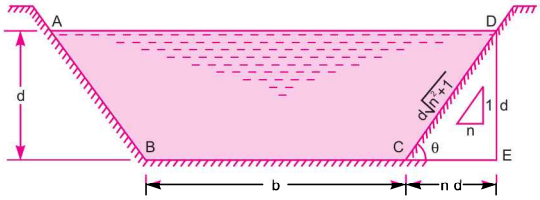


Fig. 16.10 *Trapezoidal section.*

(*i*) The side slope is given as 1 vertical to n horizontal.

∴ Area of flow, $A = \frac{(BC + AD)}{2} \times d = \frac{b + (b + 2nd)}{2} \times d$ ($\because$ $AD = b + 2nd$)

$$= \frac{2b + 2nd}{2} \times d = (b + nd) \times d \quad ...(i)$$

$$\therefore \quad \frac{A}{d} = b + nd$$

$$\therefore \quad b = \frac{A}{d} - nd \quad ...(ii)$$

Now wetted perimeter, $P = AB + BC + CD = BC + 2CD$ ($\because$ $AB = CD$)

$$= b + 2\sqrt{CE^2 + DE^2} = b + 2\sqrt{n^2d^2 + d^2} = b + 2d\sqrt{n^2 + 1} \quad ...(iia)$$

Substituting the value of b from equation (*ii*), we get

$$P = \frac{A}{d} - nd + 2d\sqrt{n^2 + 1} \quad ...(iii)$$

For most economical section, P should be minimum or $\frac{dP}{d(d)} = 0$

∴ Differentiating equation (*iii*) with respect to d and equating it equal to zero, we get

$$\frac{d}{d(d)}\left[\frac{A}{d} - nd + 2d\sqrt{n^2 + 1}\right] = 0$$

or $$-\frac{A}{d^2} - n + 2\sqrt{n^2 + 1} = 0 \quad (\because \ n \text{ is constant})$$

or $$\frac{A}{d^2} + n = 2\sqrt{n^2+1}$$

Substituting the value of A from equation (i) in the above equation,

$$\frac{(b+nd)d}{d^2} + n = 2\sqrt{n^2+1} \quad \text{or} \quad \frac{b+nd}{d} + n = 2\sqrt{n^2+1}$$

or $$\frac{b+nd+nd}{d} = \frac{b+2nd}{d} = 2\sqrt{n^2+1} \quad \text{or} \quad \frac{b+2nd}{2} = d\sqrt{n^2+1} \qquad ...(16.11)$$

But from Fig. 16.10, $\frac{b+2nd}{2}$ = Half of top width

and $d\sqrt{n^2+1} = CD$ = one of the sloping side

Equation (16.11) is the required condition for a trapezoidal section to be most economical, which can be expressed as half of the top width must be equal to one of the sloping sides of the channel.

(*ii*) **Hydraulic mean depth**

Hydraulic mean depth, $m = \frac{A}{P}$

Value of A from (i), $A = (b + nd) \times d$

Value of P from (iia), $P = b + 2d\sqrt{n^2+1} = b + (b + 2nd)$ ($\because$ From equation (16.11) $b + 2nd = 2d\sqrt{n^2+1}$)

$= 2b + 2nd = 2(b + nd)$

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{(b+nd)d}{2(b+nd)} = \frac{d}{2}$ ...(16.12)

Hence for a trapezoidal section to be most economical hydraulic mean depth must be equal to half the depth of flow,

(*iii*) The three sides of the trapezoidal section of most economical section are tangential to the semi-circle described on the water line. This is proved as :

Let Fig. 16.11 shows the trapezoidal channel of most economical section.

Let θ = angle made by the sloping side with horizontal, and

O = the centre of the top width, AD.

Draw OF perpendicular to the sloping side AB.

ΔOAF is a right-angled triangle and angle $OAF = \theta$

$\therefore$ $\sin\theta = \frac{OF}{OA}$ $\therefore$ $OF = AO \sin\theta$ ...(iv)

In ΔAEB, $\sin\theta = \frac{AE}{AB} = \frac{d}{\sqrt{d^2 + n^2d^2}}$

$= \frac{d}{d\sqrt{1+n^2}} = \frac{1}{\sqrt{1+n^2}}$

Substituting $\sin\theta = \frac{1}{\sqrt{1+n^2}}$ in equation (iv), we get

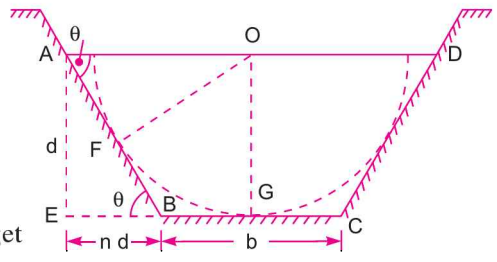


Fig. 16.11

$$OF = AO \times \frac{1}{\sqrt{1+n^2}} \qquad ...(v)$$

But AO = half of top width

$$= \frac{b+2nd}{2} = d\sqrt{n^2+1} \text{ from equation (16.11)}$$

Substituting this value of AO in equation (v),

$$OF = \frac{d\sqrt{n^2+1}}{\sqrt{n^2+1}} = d \text{ depth of flow} \qquad ...(16.13)$$

Thus, if a semi-circle is drawn with O as centre and radius equal to the depth of flow d, the three sides of most economical trapezoidal section will be tangential to the semi-circle.

Hence the conditions for the most economical trapezoidal section are:

1. $\frac{b+2nd}{2} = d\sqrt{n^2+1}$

2. $m = \frac{d}{2}$

3. A semi-circle drawn from O with radius equal to depth of flow will touch the three sides of the channel.

Problem 16.16 *A trapezoidal channel has side slopes of 1 horizontal to 2 vertical and the slope of the bed is 1 in 1500. The area of the section is 40 m^2. Find the dimensions of the section if it is most economical. Determine the discharge of the most economical section if C = 50.*

Solution. Given :

Side slope, $n = \frac{\text{Horizontal}}{\text{Vertical}} = \frac{1}{2}$

Bed slope, $i = \frac{1}{1500}$

Area of section, $A = 40 \text{ m}^2$

Chezy's constant, $C = 50$

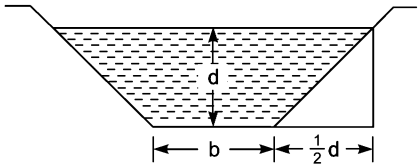


Fig. 16.12

For the most economical section, using equation (16.11)

$$\frac{b+2nd}{2} = d\sqrt{n^2+1} \quad \text{or} \quad \frac{b+2\times\frac{1}{2}\times d}{2} = d\sqrt{\left(\frac{1}{2}\right)^2+1}$$

or $$\frac{b+d}{2} = d\sqrt{\frac{1}{4}+1} = 1.118\, d$$

or $$b = 2 \times 1.118d - d = 1.236\, d \qquad ...(i)$$

But area of trapezoidal section, $A = \frac{b+(b+2nd)}{2} \times d = (b + nd)\, d$

$$= (1.236\, d + \tfrac{1}{2}\, d)\, d \qquad (\because \quad b = 1.236\, d \text{ and } n = \tfrac{1}{2})$$

$$= 1.736\, d^2$$

But $A = 40 \text{ m}^2$ (given)

$\therefore$ $40 = 1.736\, d^2$

$\therefore$ $$d = \sqrt{\frac{40}{1.736}} = \mathbf{4.80\ m.\ Ans.}$$

Substituting the value of d in equation (i), we get

$$b = 1.236 \times 4.80 = \mathbf{5.933\ m.\ Ans.}$$

Discharge for most economical section. Hydraulic mean depth for most economical section is

$$m = \frac{d}{2} = \frac{4.80}{2} = 2.40\ \text{m}$$

$\therefore$ Discharge $$Q = AC\sqrt{mi} = 40 \times 50 \times \sqrt{2.40 \times \frac{1}{1500}}$$

$$= \mathbf{80\ m^3/s.\ Ans.}$$

Problem 16.17 *A trapezoidal channel has side slopes of 3 horizontal to 4 vertical and slope of its bed is 1 in 2000. Determine the optimum dimensions of the channel, if it is to carry water at 0.5 m^3/s. Take Chezy's constant as 80.*

Solution. Given :

Side slopes $$n = \frac{\text{Horizontal}}{\text{Vertical}} = \frac{3}{4}$$

Slope of bed, $$i = \frac{1}{2000}$$

Discharge, $$Q = 0.5\ \text{m}^3/\text{s}$$

Chezy's constant, $$C = 80$$

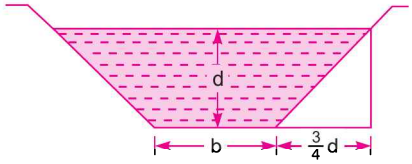


Fig. 16.13

For the most economical section, the condition is given by equation (16.11) as

$$\frac{b + 2nd}{2} = d\sqrt{n^2 + 1}, \text{ where } b = \text{width of section}, d = \text{depth of flow}$$

or $$\frac{b + 2 \times \frac{3}{4}d}{2} = d\sqrt{\left(\frac{3}{4}\right)^2 + 1} = \frac{5}{4}d \quad \text{or} \quad \frac{b + 1.5d}{2} = 1.25\ d$$

or $$b = 2 \times 1.25\ d - 1.5\ d = d \qquad ...(i)$$

For the discharge, Q, using equation (16.5) as

$$Q = AC\sqrt{mi} \qquad ...(ii)$$

But for most economical section, hydraulic mean depth $m = \frac{d}{2}$

Substituting the value of m and other known values in equation (ii)

$$0.50 = A \times 80 \times \sqrt{\frac{d}{2} \times \frac{1}{2000}} \qquad ...(iii)$$

But area of trapezoidal section is given as

$$A = (b + nd) \times d = (d + \tfrac{3}{4}d) \times d \quad (\because \text{ From } (i)\ b = d \text{ and } n = \tfrac{3}{4})$$

$$= \frac{7}{4}d^2 = 1.75\ d^2$$

Substituting the value of A in equation (iii), we get

$$0.50 = 1.75\, d^2 \times 80 \times \sqrt{\frac{d}{2} \times \frac{1}{2000}} = 2.2135\, d^{5/2}$$

$$\therefore \qquad d = \left(\frac{0.50}{2.2135}\right)^{2/5} = \textbf{0.55 m. Ans.}$$

From equation (*i*), $b = d = \textbf{0.55 m. Ans.}$

$\therefore$ Optimum dimensions of the channel are width = depth = 0.55 m.

Problem 16.18 *A trapezoidal channel with side slopes of 1 to 1 has to be designed to convey 10 m^3/s at a velocity of 2 m/s so that the amount of concrete lining for the bed and sides is the minimum. Calculate the area of lining required for one metre length of canal.*

Solution. Given :

Side slope, $n = \dfrac{\text{Horizontal}}{\text{Vertical}} = 1$

Discharge $Q = 10\ m^3/s$

Velocity, $V = 2.0$ m/s

$\therefore$ Area of flow, $A = \dfrac{\text{Discharge}}{\text{Velocity}} = \dfrac{10.0}{2.0} = 5\ m^2$ ...(*i*)

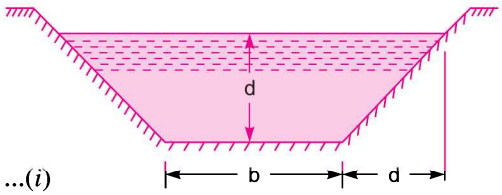


Fig. 16.14

Let b = Width of the channel

d = Depth of flow

For the amount of concrete lining for the bed and sides to be minimum the section should be most economical. But for the most economical trapezoidal section, the condition is from equation (16.11) as

Half of the top width = one of the sloping side

i.e., $$\frac{b + 2nd}{2} = d\sqrt{n^2 + 1}$$

For $n = 1$, the condition becomes

$$\frac{b + 2 \times 1d}{2} = d\sqrt{1^2 + 1} = 1.414\, d$$

or $$b = 2 \times 1.414d - 2d = 0.828\, d \qquad \text{...(}ii\text{)}$$

But area, $A = (b + nd)\, d = (0.828d + 1 \times d)\, d \qquad (\because\ b = 0.828\, d,\ n = 1)$

$= 1.828\, d^2$

Also from equation (*i*), $A = 5\ m^2$

Equating the two values of A, we get

$$5 = 1.828\, d^2 \quad \text{or} \quad d = \sqrt{\frac{5}{1.828}} = 1.6538 \simeq 1.654 \text{ m}$$

From equation (*ii*), $b = 0.828\, d = 0.828 \times 1.654 = 1.369$ m

Area of lining required for one metre length of canal

= Wetted perimeter × length of canal

$= P \times 1$

where $P = b + 2d\sqrt{n^2 + 1} = 1.369 + 2 \times 1.654\sqrt{1^2 + 1} = 6.047$ m

$\therefore$ Area of lining $= 6.047 \times 1 = \textbf{6.047 m}^2\textbf{. Ans.}$

Problem 16.19 *A trapezoidal channel has side slopes 1 to 1. It is required to discharge 13.75 m^3/s of water with a bed gradient of 1 in 1000. If unlined the value of Chezy's C is 44. If lined with concrete, its value is 60. The cost per m^3 of excavation is four times the cost per m^2 of lining. The channel is to be the most efficient one. Find whether the lined canal or the unlined canal will be cheaper. What will be the dimensions of that economical canal ?*

Solution. Given :

Side slope, $n = \frac{1}{1} = 1$

Discharge, $Q = 13.75$ m^3/s

Slope of bed, $i = \frac{1}{1000}$

For unlined, $C = 44$

For lined $C = 60$

Cost per m^3 of excavation $= 4 \times$ cost per m^2 of lining

Let the cost per m^2 of lining $= x$

Then cost per m^3 of excavation $= 4x$

As the channel is most efficient,

$\therefore$ Hydraulic mean depth, $m = \frac{d}{2}$, where d = depth of channel

Let b = width of channel

Also for the most efficient trapezoidal channel, from equation (16.11), we have

Half of top width = length of sloping side

or $$\frac{b + 2nd}{2} = d\sqrt{n^2 + 1}$$

or $$\frac{b + 2 \times 1 \times d}{2} = d\sqrt{1^2 + 1} = \sqrt{2}d$$

$\therefore$ $$b = 2 \times \sqrt{2d} - 2d = 0.828\ d \qquad ...(i)$$

Area, $$A = (b + nd) \times d = (0.828\ d + 1 \times d) \times d$$
$$= 1.828\ d^2 \qquad ...(ii)$$

1. **For unlined channel**

Value of $C = 44$

The discharge, Q is given by, $Q = A \times V = A \times C\sqrt{mi}$

or $$13.75 = 1.828\ d^2 \times 44 \times \sqrt{\frac{d}{2} \times \frac{1}{1000}} \qquad \left(\because\ A = 1.828\ d^2,\ m = \frac{d}{2}\right)$$

$$= \frac{1.828 \times 44}{\sqrt{2000}} \times d^{5/2}$$

$$d^{5/2} = \frac{13.75 \times \sqrt{2000}}{1.828 \times 44} = 7.6452$$

$\therefore$ $$d = (7.6452)^{2/5} = 2.256 \text{ m}$$

Substituting this value in equation (i), we get

$b = 0.828 \times 2.256 = 1.868$ m.

Now cost of excavation per running metre length of unlined channel

= Volume of channel × cost per m^3 of excavation

= (Area of channel × 1) × $4x$ = $[(b + nd) \times d \times 1] \times 4x$

= $(1.868 + 1 \times 2.256) \times 2.256 \times 1 \times 4x = 37.215\ x$ ...(*iii*)

2. **For lined channels**

Value of $C = 60$

The discharge is given by the equation, $Q = A \times C \times \sqrt{mi}$

Substituting the value of A from equation (*ii*) and value of $m = \frac{d}{2}$, we get

$$13.75 = 1.828\ d^2 \times 60 \times \sqrt{\frac{d}{2} \times \frac{1}{1000}} \qquad (\because \quad Q = 13.75)$$

$$= 1.828 \times 60 \times \frac{1}{\sqrt{2000}} \times d^{5/2}$$

$$\therefore \qquad d^{5/2} = \frac{13.75 \times \sqrt{2000}}{1.828 \times 60} = 5.606$$

$$\therefore \qquad d = (5.606)^{2/5} = 1.992 \text{ m}$$

Substituting this value in equation (*i*), we get

$b = 0.828 \times 1.992 = 1.649$ m

In case of lined channel, the cost of lining as well as cost of excavation is to be considered.

Now cost of excavation = (Volume of channel) × cost per m^3 of excavation

= $(b + nd) \times d \times 1 \times 4x$

= $(1.649 + 1 \times 1.992) \times 1.992 \times 1 \times 4x = 29.01\ x$

Cost of lining = Area of lining × cost per m^2 of lining

= (Perimeter of lining × 1) × x

= $(b + 2d\sqrt{1+n^2}) \times 1 \times x = (1.649 + 2 \times 1.992\sqrt{1+1^2}) \times 1 \times x$

= $(1.649 + 2 \times 1.992 \times \sqrt{2})\ x = 7.283\ x$

$\therefore$ Total cost = $29.01x + 7.283x = 36.293x$

The total cost of lined channel is $36.293x$ whereas the total cost of unlined channel is $37.215x$. Hence lined channel will be cheaper. The dimensions are $b = 1.649$ m and $d = 1.992$ m. **Ans.**

Problem 16.20 *An open channel of most economical section, having the form of a half hexagon with horizontal bottom is required to give a maximum discharge of 20.2 m^3/s of water. The slope of the channel bottom is 1 in 2500. Taking Chezy's constant, C = 60 in Chezy's equation, determine the dimensions of the cross-section.*

Solution. Given :

Maximum discharge, $Q = 20.2\ m^3/s$

Bed slope, $i = \frac{1}{2500}$

Chezy's constant, $C = 60$

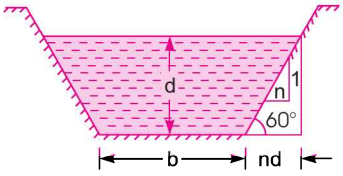


Fig. 16.15

Channel is the form of a half hexagon as shown in Fig. 16.15. This means that the angle made by the sloping side with horizontal will be 60°.

$$\therefore \qquad \tan\theta = \tan 60° = \sqrt{3} = \frac{1}{n}$$

$$\therefore \qquad n = \frac{1}{\sqrt{3}}$$

Let b = width of the channel, d = depth of the flow.

As the channel given is of most economical section, hence the condition given by equations (16.11) and (16.12) should be satisfied *i.e.*,

Half of the top width = one of the sloping side

And hydraulic mean depth = half of depth of flow

From equation (16.11), $\dfrac{b + 2nd}{2} = d\sqrt{n^2 + 1}$

For $n = \dfrac{1}{\sqrt{3}}$, $\qquad \dfrac{b + 2 \times \dfrac{1d}{\sqrt{3}}}{2} = d\sqrt{\left(\dfrac{1}{\sqrt{3}}\right)^2 + 1} = \dfrac{2d}{\sqrt{3}}$

or $\qquad \dfrac{\sqrt{3}b + 2d}{2\sqrt{3}} = \dfrac{2d}{\sqrt{3}} \quad \text{or} \quad \dfrac{\sqrt{3}b + 2d}{2} = 2d$

$$\therefore \qquad b = \frac{2 \times 2d - 2d}{\sqrt{3}} = \frac{2d}{\sqrt{3}} \qquad \ldots(i)$$

Area of flow, $\qquad A = (b + nd)\, d = \left(\dfrac{2}{\sqrt{3}}d + \dfrac{d}{\sqrt{3}}\right)d \qquad \left(\because\ n = \dfrac{1}{\sqrt{3}}, b = \dfrac{2d}{\sqrt{3}}\right)$

$$= \frac{3}{\sqrt{3}}d^2 = \sqrt{3}d^2$$

From equation (16.12) $\qquad m = \dfrac{d}{2}$

Using equation (16.5) for discharge Q as

$$Q = AC\sqrt{mi} \quad \text{or} \quad 20.2 = \sqrt{3}\ d^2 \times 60 \times \sqrt{\frac{d}{2} \times \frac{1}{2500}} = 1.4694\ d^{5/2}$$

$$\therefore \qquad d^{5/2} = \frac{20.2}{1.4696} = 13.745$$

$$\therefore \qquad d = (13.745)^{2/5} = \mathbf{2.852\ m.\ Ans.}$$

Substituting this value in equation (i), we get

$$b = \frac{2d}{\sqrt{3}} = \frac{2}{\sqrt{3}} \times 2.852 = \mathbf{3.293\ m.\ Ans.}$$

Problem 16.21 *A trapezoidal channel to carry 142 m^3/minute of water is designed to have a minimum cross-section. Find the bottom width and depth if the bed slope is 1 in 1200, the side slopes at 45° and Chezy's co-efficient = 55.*

Solution. Given : Discharge, $Q = 142 \text{ m}^3/\text{min.} = \frac{142}{60} = 2.367 \text{ m}^3/\text{s}$

Bed slope, $i = 1 \text{ in } 1200 = \frac{1}{1200}$

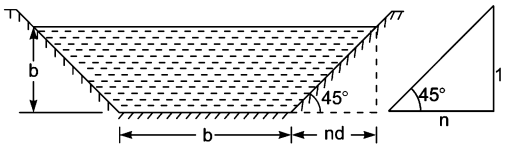


Fig. 16.16

Side slope, $\theta = 45°$

$\therefore \quad \tan\theta = \frac{1}{n}$ or $\tan 45° = \frac{1}{n}$

$\therefore \quad 1 = \frac{1}{n}$ or $n = 1$

Chezy's constant, $C = 55$

Let b = Width of the channel, d = Depth of the flow.

As the channel is to be designed for a minimum cross-section (*i.e.*, channel is of most economical section), the conditions given by equations (16.11) and (16.12) should be satisfied *i.e.*,

(*i*) Half of top with = Length of sloping side

(*ii*) Hydraulic mean depth = Half of depth of flow

From equation (16.11), $\frac{b + 2nd}{2} = d\sqrt{n^2 + 1}$

or $\frac{b + 2 \times 1 \times d}{2} = d\sqrt{1^2 + 1}$ ($\because \ n = 1$)

or $b + 2d = 2\sqrt{2d} = 2 \times 1.414\, d = 2.828\, d$

$\therefore \quad b = 2.828\, d - 2d = 0.828\, d$ ...(*i*)

Now using equation (16.5) for discharge Q, we get

$$Q = A \,.\, C \,.\, \sqrt{mi}$$

or $2.367 = (b + nd)\, d \times 55\sqrt{\frac{d}{2} \times \frac{1}{1200}}$ $\left(\because \ A = (b + nd) \times d \text{ and } m = \frac{d}{2}\right)$

$= (0.828d \times 1 \times d)\, d \times 55\sqrt{\frac{d}{2400}}$ ($\because \ b = 0.828d$)

$= (1.828d) \times d \times 55\sqrt{\frac{d}{2400}} = 2.052\, d^{5/2}$

$\therefore \quad d = \left(\frac{2.367}{2.052}\right)^{2/5} = 1.058 \simeq \mathbf{1.06 \text{ m. Ans.}}$

Substituting this value in equation (*i*), we get

$b = 0.828 \times 1.06 = \mathbf{0.877 \text{ m. Ans.}}$

Problem 16.22 *A trapezoidal channel with side slopes of 3 horizontal to 2 vertical has to be designed to convey 10 m^3/s at a velocity of 1.5 m/s, so that the amount of concrete lining for the bed and sides is minimum. Find*

(i) the wetted perimeter, and

(ii) slope of the bed if Manning's N = 0.014 in the formula $C = \frac{1}{N} \times m^{1/6}$

Solution. (*i*) Given :

Side slope, $n = \frac{\text{Horizontal}}{\text{Vertical}} = \frac{3}{2} = 1.5$

Discharge, $Q = 10\ m^3/s$

Velocity, $V = 1.5$ m/s

Manning's constant, $N = .014$

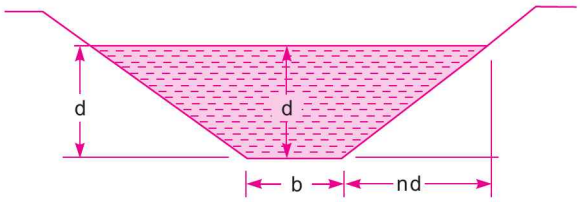


Fig. 16.17

Let b = width of the channel, d = depth of the flow.

The amount of concrete lining for the bed and sides will be minimum, when the section is most economical. For most economical trapezoidal section, the condition is given by equation (16.11) as

$$\frac{b + 2nd}{2} = d\sqrt{n^2 + 1}$$

For $n = 1.5$, $\frac{b + 2 \times 1.5 \times d}{2} = d\sqrt{1.5^2 + 1} = \sqrt{3.25}\, d = 1.8\, d$ or $\frac{b + 3d}{2} = 1.8\, d$

$\therefore$ $b = 2 \times 1.8 - 3d = 0.6\, d$ ...(*i*)

But area of trapezoidal section, $A = (b + nd)d = (0.6d + 1.5d)d$ ($\because$ $b = 0.6d$, $n = 1.5$)

$= 2.1\, d^2$

Also area, $A = \frac{\text{Discharge}}{\text{Velocity}} = \frac{Q}{V} = \frac{10.0}{1.5} = 6.67\ m^2$

Equating the two values of A, we have $2.1\, d^2 = 6.67$

$\therefore$ $d = \sqrt{\frac{6.67}{2.1}} = 1.78$ m

From equation (*i*), $b = 0.6d = 0.6 \times 1.78 = 1.068 \simeq 1.07$ m

Hence wetted perimeter, $P = b + 2d\sqrt{n^2 + 1} = 1.07 + 2 \times 1.78\sqrt{1.5^2 + 1}$ = **7.48 m. Ans.**

(*ii*) Slope of the bed when $N = 0.014$ in the formula, $C = \frac{1}{N} m^{1/6}$

For the most economical trapezoidal section, hydraulic mean depth m, is given by equation (16.12) as

$$m = \frac{d}{2} = \frac{1.78}{2} = 0.89 \text{ m}$$

$$C = \frac{1}{0.014} \times (.89)^{1/6} = 66.09$$

Using equation (16.5), $Q = AC\sqrt{mi}$

or $$10.0 = 6.67 \times 66.09\sqrt{0.89} \times i = 415.86\sqrt{i}$$

$\therefore$ $$i = \left(\frac{10}{415.86}\right)^2 = \mathbf{\frac{1}{1729.4}}. \quad \textbf{Ans.}$$

Hence slope of the bed is 1 in 1729.4.

16.5.3 Best Side Slope for Most Economical Trapezoidal Section.

Area of trapezoidal section, $A = (b + nd)d$ ...(*i*)

where b = width of trapezoidal channel, d = depth of flow, and

n = slope of the side of the channel

From equation (*i*), $b = \frac{A}{d} - nd$ ...(*ii*)

Perimeter (wetted) of channel, $P = b + 2d\sqrt{n^2 + 1}$

Substituting the value of b from equation (*ii*), perimeter becomes

$$P = \frac{A}{d} - nd + 2d\sqrt{n^2 + 1} \quad ...(iii)$$

For the most economical trapezoidal section, the depth of flow, d and area A are constant. Then n is the only variable. Best side slope will be when section is most economical or in other words, P is minimum. For P to be minimum, we must have $\frac{dP}{dn} = 0$

Hence differentiating equation (*iii*) with respect to n,

$$\frac{d}{dn}\left[\frac{A}{d} - nd + 2d\sqrt{n^2 + 1}\right] = 0$$

or $$-d + 2d \times \frac{1}{2} \times (n^2 + 1)^{1/2-1} \times 2n = 0 \text{ or } -d + 2nd \times \frac{1}{\sqrt{n^2 + 1}} = 0$$

Cancelling d and re-arranging, we get $2n = \sqrt{n^2 + 1}$

Squaring to both sides,

$$4n^2 = n^2 + 1 \text{ or } 3n^2 = 1 \text{ or } n = \sqrt{\frac{1}{3}} = \frac{1}{\sqrt{3}} \quad ...(16.14)$$

If the sloping side makes an angle θ, with the horizontal, then we have

$$\tan\theta = \frac{1}{n} = \frac{1}{\frac{1}{\sqrt{3}}} = \sqrt{3} = \tan 60°$$

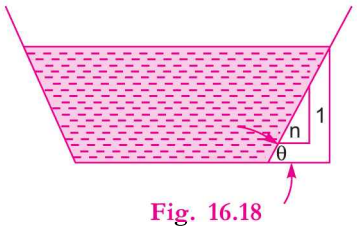


Fig. 16.18

$$\therefore \qquad \theta = 60° \qquad \ldots(16.15)$$

Hence best side slope is at 60° to the horizontal or the value of n for the best side slope is given by equation (16.14).

For the most economical trapezoidal section, we have

Half of top width = length of one sloping side

or
$$\frac{b + 2nd}{2} = d\sqrt{n^2 + 1}$$

Substituting the value of n from equation (16.14), we have

$$\frac{b + 2 \times \frac{1}{\sqrt{3}} \times d}{2} = d\sqrt{\left(\frac{1}{\sqrt{3}}\right)^2 + 1} = \frac{2d}{\sqrt{3}} \quad \text{or} \quad \frac{\sqrt{3}b + 2d}{2 \times \sqrt{3}} = \frac{2d}{\sqrt{3}}$$

or
$$\sqrt{3}b + 2d = 2 \times \sqrt{3} \times \frac{2d}{\sqrt{3}} = 4d$$

$$\therefore \qquad b = \frac{4d - 2d}{\sqrt{3}} = \frac{2d}{\sqrt{3}} \qquad \ldots(iv)$$

Now, wetted perimeter, $P = b + 2d\sqrt{n^2 + 1}$

$$= \frac{2d}{\sqrt{3}} + 2d\sqrt{\left(\frac{1}{\sqrt{3}}\right)^2 + 1} \qquad \left(\because\ b = \frac{2d}{\sqrt{3}}, n = \frac{1}{\sqrt{3}}\right)$$

$$= \frac{2d}{\sqrt{3}} + 2d \times \frac{2}{\sqrt{3}} = \frac{2d}{\sqrt{3}} + \frac{4d}{\sqrt{3}}$$

or
$$P = \frac{6d}{\sqrt{3}} = 3 \times \frac{2d}{\sqrt{3}} = 3 \times b \qquad \left(\because\ \text{From } (iv), \frac{2d}{\sqrt{3}} = b\right)$$

For a slope of 60°, the length of sloping side is equal to the width of the trapezoidal section.

Problem 16.23 *A power canal of trapezoidal section has to be excavated through hard clay at the least cost. Determine the dimensions of the channel given, discharge equal to 14 m^3/s, bed slope 1 : 2500 and Manning's N = 0.020.*

Solution. Given :

Discharge, $Q = 14\ m^3/s$

Bed slope, $i = \frac{1}{2500}$

Manning's, $N = 0.020$

For excavation of the canal at the least cost, the trapezoidal section should be most economical. Here side slope (*i.e.*, value of n) is not given. Hence the best side slope for most economical trapezoidal section (*i.e.*, the value of n) is given by equation (16.14) as $n = \frac{1}{\sqrt{3}}$

Let b = width of channel, d = depth of flow

For most economical section,

Half of top width = length of one of sloping side

or $$\frac{b + 2nd}{2} = d\sqrt{n^2 + 1}$$

For $n = \frac{1}{\sqrt{3}}$, $$\frac{b + 2 \times \frac{1}{\sqrt{3}} d}{2} = d\sqrt{\left(\frac{1}{\sqrt{3}}\right)^2 + 1} = \frac{2d}{\sqrt{3}}$$

or $$b = \frac{2 \times 2d}{\sqrt{3}} - \frac{2d}{\sqrt{3}} = \frac{2d}{\sqrt{3}} \qquad ...(i)$$

Area of trapezoidal section, $A = (b + nd) \times d = \left(\frac{2d}{\sqrt{3}} + \frac{1}{\sqrt{3}} d\right) \times d \qquad \left(\because\ b = \frac{2d}{\sqrt{3}}, n = \frac{1}{\sqrt{3}}\right)$

$$= \sqrt{3} d^2$$

Hydraulic mean depth for most economical section, $m = \frac{d}{2}$

Now discharge, Q is given by $Q = AC\sqrt{mi}$, where $C = \frac{1}{N} m^{1/6}$

$\therefore$ $$Q = \sqrt{3} d^2 \times \frac{1}{N} m^{1/6} \sqrt{m \times \frac{1}{2500}}$$

$$= \sqrt{3} d^2 \times \frac{1}{0.020} \times m^{1/6 + 1/2} \times \sqrt{\frac{1}{2500}} = 1.732\ d^2 \times m^{2/3}$$

or $$14.0 = 1.732\ d^2 \times \left(\frac{d}{2}\right)^{2/3} = \frac{1.732}{2^{2/3}} d^{8/3} = 1.09\ d^{8/3}$$

$\therefore$ $$d^{8/3} = \frac{14.0}{1.09} = 12.844$$

$\therefore$ $$d = (12.844)^{3/8} = (12.844)^{0.375} = \mathbf{2.605\ m.\ Ans.}$$

From equation (*i*), $$b = \frac{2d}{\sqrt{3}} = \frac{2 \times 2.605}{1.732} = \mathbf{3.008\ m.\ Ans.}$$

Problem 16.24 *For a trapezoidal channel with bottom width 40 m and side slopes 2H : 1 V, Manning's N is 0.015 and bottom slope is 0.0002. If it carries 60 m^3/s discharge, determine the normal depth.*

Solution. Given :

Bottom width, $b = 40$ m

Side slopes 2 horizontal to 1 vertical *i.e.*, $n = 2$

$\therefore$ Manning's constant, $N = 0.015$

Bed slope, $i = 0.0002$

Discharge, $Q = 60$ m^3/s

Let d = Normal depth.

Now $A = (b + nd) \times d = (40 + 2d) \times d$

$$P = b + 2d\sqrt{1 + n^2} = 40 + 2d\sqrt{1 + 2^2} = 40 + 2 \times \sqrt{5}\, d = 40 + 4.472\, d$$

$\therefore$ $$m = \frac{A}{P} = \frac{(40 + 2d) \times d}{40 + 4.472d}$$

The discharge is given by, Q = Area × Velocity

$$= A \times \frac{1}{N} m^{2/3}\, i^{1/2} = \frac{A}{N} \times m^{2/3} \times i^{1/2}$$

$$60 = \frac{(40 + 2d) \times d}{0.015} \times \left[\frac{(40 + 2d) \times d}{40 + 4.472d}\right]^{2/3} \times 0.0002^{1/2}$$

$$= \frac{[(40 + 2d) \times d]^{5/3}}{0.015 \times (40 + 4.472d)^{2/3}} \times 0.01414$$

$\therefore$ $$\frac{60 \times 0.015 \times (40 + 4.472d)^{2/3}}{0.01414} = [(40 + 2d) \times d]^{5/3}$$

$$63.65(40 + 4.472\, d)^{2/3} = (40d + 2d^2)^{5/3}$$

$$(40d + 2d^2)^{5/3} - 63.65\,(40 + 4.472d)^{2/3} = 0 \qquad ...(i)$$

The above equation will be solved by Hit and Trial method.

(*i*) Assume $d = 1$ m, then L.H.S of equation (*i*) will as

$$\text{L.H.S.} = (40 + 2)^{5/3} - 63.65\,(40 + 4.472)^{2/3}$$
$$= 42^{5/3} - 63.65 \times 44.472^{2/3} = 513.838 - 808.4 = -294.56$$

(*ii*) Assume $d = 2$ m, then L.H.S. of equation (*i*) will be as

$$\text{L.H.S.} = (40 \times 2 + 2 \times 2^2)^{5/3} - 63.65(40 + 4.47 \times 2)^{2/3}$$
$$= 88^{5/3} - 63.65 \times 48.944^{2/3} = 1767.2 - 862.77 = 904.43$$

where $d = 1$ m, L.H.S. is – ve. But when $d = 2$ m, L.H.S. is +ve. Hence value of d lies between 1 and 2.

(*iii*) Assume $d = 1.3$ m, then L.H.S. of equation (*i*) will be as

$$\text{L.H.S.} = (40 \times 1.3 + 2 \times 1.3^2)^{5/3} - 63.65(40 + 4.472 \times 1.3)^{2/3}$$
$$= 55.38^{5/3} - 63.65 \times 45.8136^{2/3} = 815.45 - 825.4 = -9.95$$

(*iv*) Assume $d = 1.31$ m, then L.H.S. of equation (*i*) will be

$$\text{L.H.S.} = (40 \times 1.31 + 2 \times 1.31^2)^{5/3} - 63.65(40 + 4.472 \times 1.31)^{2/3}$$
$$= 55.8322^{5/3} - 63.65 \times 45.8583^{2/3} = 826.6 - 825.9 = 0.7$$

The value of L.H.S. = 0.7 is negligible in comparison to the value of 904.43.

$\therefore$ Value of d = **1.31 m. Ans.**

16.5.4 Flow Through Circular Channel. The flow of a liquid through a circular pipe, when the level of liquid in the pipe is below the top of the pipe is classified as an open channel flow. The rate of flow through circular channel is determined from the depth of flow and angle subtended by the liquid surface at the centre of the circular channel.

Fig.16.19 shows a circular channel through which water is flowing.

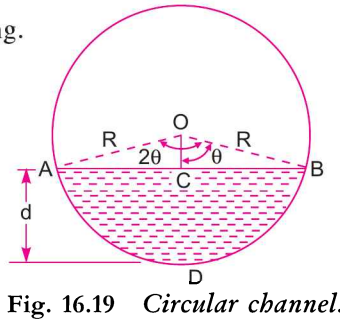


Fig. 16.19 *Circular channel.*

Let d = depth of water,

2θ = angle subtended by water surface AB at the centre in radians,

R = radius of the channel,

Then the wetted perimeter and wetted area is determine as :

Wetted perimeter, $P = \dfrac{2\pi R}{2\pi} \times 2\theta = 2R\theta$ ...(16.16)

Wetted area, A = Area $ADBA$

= Area of sector $OADBO$ – Area of ΔABO

$$= \frac{\pi R^2}{2\pi} \times 2\theta - \frac{AB \times CO}{2} = R^2\theta - \frac{2BC \times CO}{2} \qquad (\because \ AB = 2BC)$$

$$= R^2\theta - \frac{2 \times R \sin\theta \times R \cos\theta}{2} \qquad (\because \ BC = R \sin\theta, \ CO = R \cos\theta)$$

$$= R^2\theta - \frac{R^2 \times 2 \sin\theta \cos\theta}{2} = R^2\theta - \frac{R^2 \sin 2\theta}{2} \qquad (\because \ 2 \sin\theta \cos\theta = \sin 2\theta)$$

$$= R^2\left(\theta - \frac{\sin 2\theta}{2}\right) \qquad ...(16.17)$$

Then hydraulic mean depth, $m = \dfrac{A}{P} = \dfrac{R^2\left(\theta - \dfrac{\sin 2\theta}{2}\right)}{2R\theta} = \dfrac{R}{2\theta}\left(\theta - \dfrac{\sin 2\theta}{2}\right)$

And discharge, Q is given by, $Q = AC\sqrt{mi}$.

Problem 16.25 *Find the discharge through a circular pipe of diameter 3.0 m, if the depth of water in the pipe is 1.0 m and the pipe is laid at a slope of 1 in 1000. Take the value of Chezy's constant as 70.*

Solution. Given :

Dia. of pipe, $D = 3.0$

$\therefore$ Radius, $R = \dfrac{D}{2} = \dfrac{3.0}{2} = 1.50$ m

Depth of water in pipe, $d = 1.0$ m

Bed slope, $i = \dfrac{1}{1000}$

Chezy's constant, $C = 70$

From Fig. 16.20, we have $OC = OD - CD = R - 1.0$

$= 1.5 - 1.0 = 0.5$ m

$AO = R = 1.5$ m

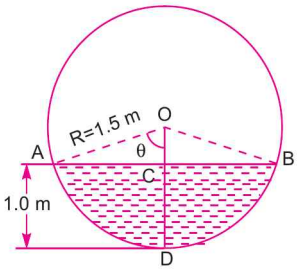


Fig. 16.20

Also $\cos\theta = \frac{OC}{AO} = \frac{0.5}{1.5} = \frac{1}{3}$

$\therefore$ $\theta = 70.53° = 70.53 \times \frac{\pi}{180} = 1.23$ radians ($\because$ 180° = π radians)

Wetted perimeter is given by equation (16.16) as

$P = 2R\theta = 2 \times 1.5 \times 1.23$ (θ should be in radians)

$= 3.69$ m

Wetted area is given by equation (16.17) as

$$A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right) = 1.5^2\left(1.23 - \frac{\sin(2 \times 70.53°)}{2}\right)$$

$$= 2.25\left[1.23 - \frac{\sin(141.08°)}{2}\right] = 2.25\left[1.23 - \frac{\sin(180° - 141.08°)}{2}\right]$$

$$= 2.25\left[1.23 - \frac{\sin 38.94°}{2}\right] = 2.06 \text{ m}^2$$

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{2.06}{3.69} = .5582$

The discharge is given by, $Q = AC\sqrt{mi} = 2.06 \times 70 \times \sqrt{0.5582 \times \frac{1}{1000}}$ = **3.407 m²/s. Ans.**

Problem 16.26 *If in the problem 16.25, the depth of water in the pipe is 2.5 m, find the rate of flow through the pipe.*

Solution. Given :

Dia. of pipe = 3.0 m

$\therefore$ Radius, $R = 1.5$ m

Depth of water, $d = 2.5$ m

$i = \frac{1}{1000}$ and $C = 70$

From Fig. 16.21, $OC = CD - OD = 2.5 - R = 2.5 - 1.5 = 1.0$ m

$OA = R = 1.5$ m

From ΔAOC, $\cos\alpha = \frac{OC}{OA} = \frac{1.0}{1.5} = 0.667$

$\therefore$ $\alpha = 48.16°$

$\theta = 180° - \alpha = 180° - 48.16° = 131.84°$

$= 131.84 \times \frac{\pi}{180} = 2.30$ radians

Now wetted perimeter is given by equation (16.16) as

$P = 2R\theta = 2 \times 1.5 \times 2.30 = 6.90$ m

And wetted area is given by equation (16.17) as

$$A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right) = 1.5^2\left(2.30 - \frac{\sin(2\times 131.84°)}{2}\right)$$

$$= 2.25\left(2.30 - \frac{\sin 263.68°}{2}\right)$$

$$= 2.25\left[2.30 - \frac{\sin(180° + 83.68°)}{2}\right]$$

$$= 2.25\left[2.30 - \frac{(-\sin 83.58°)}{2}\right]$$

$$= 2.25\left[2.30 + \frac{\sin 83.68°}{2}\right] = 6.293 \text{ m}^2$$

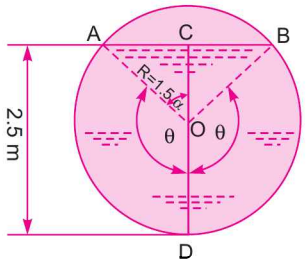


Fig. 16.21

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{6.293}{6.90} = 0.912$ m

Discharge, Q is given by, $Q = AC\sqrt{mi} = 6.293 \times 70 \times \sqrt{0.912 \times \frac{1}{1000}}$ = **13.303 m^3/s. Ans.**

Problem 16.27 *Calculate the quantity of water that will be discharged at a uniform depth of 0.9 m in a 1.2 m diameter pipe which is laid at a slope 1 in 1000. Assume Chezy's C = 58.*

Solution. Given :

Dia. of pipe = 1.2 m

$\therefore$ Radius, $R = \frac{1.2}{2} = 0.6$ m

Depth of water, $d = 0.9$ m

Slope, $i = \frac{1}{1000}$

Chezy's, $C = 58$

From Fig. 16.22, we have $OC = CD - OD$

$= 0.9 - R = 0.9 - 0.6 = 0.3$ m

$OA = R = 0.6$ m

Now in triangle AOC,

$$\cos\alpha = \frac{OC}{OA} = \frac{0.3}{0.6} = \frac{1}{2}$$

$\therefore$ $\alpha = \cos^{-1}\left(\frac{1}{2}\right) = 60°$

$\therefore$ θ = Angle $DOA = 180° - \alpha$

$= 180° - 60° = 120° = 120 \times \frac{\pi}{180}$ radians

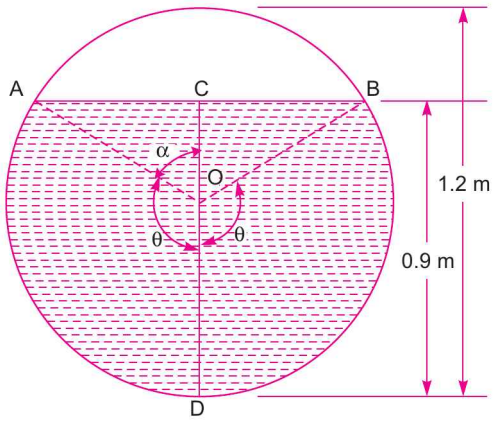


Fig. 16.22

$= 0.667\ \pi$ radians

Now wetted perimeter is given by equation (16.16) as

$$P = 2R\theta = 2 \times 0.6 \times 0.667\ \pi = 2.526 \text{ m}$$

And area of flow is given by equation (16.17) as

$$A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right)$$

$$= 0.6^2\left[0.667\pi - \frac{\sin(2\times 120°)}{2}\right] = 0.36\left[0.667\pi - \frac{\sin 240°}{2}\right]$$

$$= 0.36\left[0.667\pi - \frac{(-0.866)}{2}\right] = 0.36\ [0.667\ \pi + 0.433] = 0.913 \text{ m}^2$$

Now discharge is given by, $Q = A \times V = A \times C\sqrt{mi} = 0.913 \times 58\sqrt{\frac{A}{P} \times \frac{1}{1000}}$ $\left(\because\ m = \frac{A}{P}\right)$

$$= 0.913 \times 58\sqrt{\frac{0.913}{2.526} \times \frac{1}{1000}} = \mathbf{1.007\ m^3/s.\ Ans.}$$

Problem 16.28 *Water is flowing through a circular channel at the rate of 400 litres/s, when the channel is having a bed slope of 1 in 9000. The depth of water in the channel is 8.0 times the diameter. Find the diameter of the circular channel if the value of Manning's N = 0.015.*

Solution. Given :

Discharge, $Q = 400$ litres/s $= 0.4$ m^3/s

Bed slope, $i = \frac{1}{9000}$

Manning's, $N = 0.015$

Let the diameter of channel $= D$

Then depth of flow, $d = 0.8\ D$

From Fig. 16.23, we have

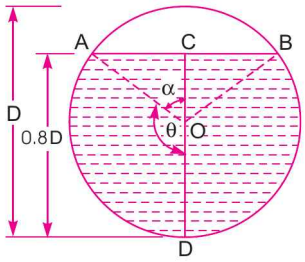


Fig. 16.23

$$OC = CD - OD = 0.8\ D - \frac{D}{2}$$

$$= (0.8 - 0.5)\ D = 0.3\ D$$

And $AO = R = \frac{D}{2} = 0.5\ D$

$\therefore$ $\cos\alpha = \frac{OC}{AO} = \frac{0.3\ D}{0.5\ D} = 0.6$

$\therefore$ $\alpha = 53.13°$

And $\theta = 180° - 53.13 = 126.87° = 126.87 \times \frac{\pi}{180} = 2.214$ radians.

From equation (16.16), wetted perimeter,

$$P = 2R\theta = 2 \times \frac{D}{2} \times 2.214 = 2.214\ D \text{ m.}$$

From equation (16.17), wetted area,

$$A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right) = \left(\frac{D}{2}\right)^2 \left[2.214 - \frac{\sin (2 \times 126.87°)}{2}\right]$$

$$= \frac{D^2}{4}\left[2.214 - \frac{\sin 253.74°}{2}\right] = \frac{D^2}{4}\left[2.214 - \frac{\sin (180° + 73.74°)}{2}\right]$$

$$= \frac{D^2}{4}\left[2.214 - \left(\frac{-\sin 73.74°}{2}\right)\right] = \frac{D^2}{4}\left[2.214 + \frac{\sin 73.74°}{2}\right]$$

$$= \frac{D^2}{4}\ [2.214 + .48] = 0.6735\ D^2$$

$\therefore$ Hydraulic mean depth, $m = \dfrac{A}{P} = \dfrac{0.6735\ D^2}{2.214\ D} = 0.3042\ D$

Discharge by Manning's formula is given by,

$$Q = \frac{1}{N} \times A \times m^{2/3} \times i^{1/2}$$

or
$$0.4 = \frac{1}{.015} \times 0.6735\ D^2 \times (.3042\ D)^{2/3} \times \left(\frac{1}{9000}\right)^{1/2}$$

$$= \frac{0.6735}{0.015}\ D^2 \times 34521 \times D^{2/3} \times 0.0105 = 0.213\ D^{8/3}$$

$\therefore$
$$D^{8/3} = \frac{0.40}{0.213} = 1.8779$$

$\therefore$
$$D = (1.8779)^{3/8} = (1.8779)^{0.375} = \textbf{1.266 m. Ans.}$$

Problem 16.29 *A sewer pipe is to be laid at a slope of 1 in 8100 to carry a maximum discharge of 600 litres/s, when the depth of water is 75% of the vertical diameter. Find the diameter of this pipe if the value of Manning's N is 0.025.*

Solution. Given :

Discharge, $Q = 600$ litres/s $= 0.6$ m^3/s

Bed slope, $i = \dfrac{1}{8100}$

Manning's, $N = 0.025$

Depth of water = 75% of dia. of pipe = 0.75 dia. of pipe

Let d = depth of water, D = Dia. of pipe

Then $d = 0.75\ D$

From Fig. 16.23 (*a*), we have $OC = CD - OD = 0.75\ D - 0.5\ D = 0.25\ D$

$$AO = R = 0.5\ D$$

In triangle AOC, $$\cos\ \alpha = \frac{OC}{AO} = \frac{0.25\ D}{0.5\ D} = 0.5$$

$$\therefore \quad \alpha = \cos^{-1} 0.5 = 60°$$

And $$\theta = 180° - \alpha = 180° - 60° = 120°$$

$$= 120 \times \frac{\pi}{180} \text{ radians} = 2.0946 \text{ radians.}$$

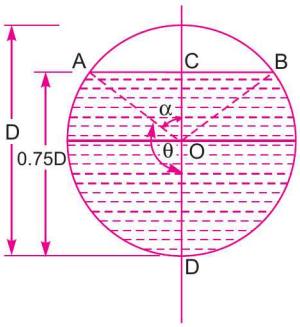


Fig. 16.23 (*a*)

From equation (16.16), wetted perimeter

$$P = 2R\theta = 2 \times 0.5\ D \times 2.0496 \quad (\because \ R = 0.5\ D)$$

$$= 2.0496\ D$$

And from equation (16.17), the area of flow,

$$A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right)$$

$$= (0.5\ D)^2 \left[2.0946 - \frac{\sin\ (2 \times 120°)}{2}\right]$$

$$= 0.25\ D^2 \left[2.0946 - \left(\frac{-.866}{2}\right)\right] = 0.25\ D^2\ [2.0946 + 0.433]$$

$$= 0.6319\ D^2$$

$$\therefore \quad m = \frac{A}{P} = \frac{0.6319\ D^2}{2.0496\ D} = 0.308\ D$$

Discharge by Manning's formula is given by

$$Q = \frac{1}{N} \times A \times m^{2/3} \times i^{1/2}$$

or $$0.6 = \frac{1}{0.025} \times 0.6319\ D^2 \times (0.308\ D)^{2/3} \times \left(\frac{1}{8100}\right)^{1/2} = 0.128 \times D^{8/3}$$

$$\therefore \quad D^{8/3} = \frac{0.6}{0.128} = 4.6875$$

$$\therefore \quad D = (4.6875)^{3/8} = \mathbf{1.785\ m.\ Ans.}$$

16.5.5 Most Economical Circular Section. We have discussed in Art. 16.5 that for a most economical section the discharge for a constant cross-sectional area, slope of bed and resistance co-efficient, is maximum. But in case of circular channels, the area of flow cannot be maintained constant. With the change of depth of flow in a circular channel of any radius, the wetted area and wetted perimeter changes. Thus in case of circular channels, for most economical section, two separate conditions are obtained. They are :

1. Condition for maximum velocity, and
2. Condition for maximum discharge.

1. Condition for Maximum Velocity for Circular Section. Fig. 16.24 shows a circular channel through which water is flowing.

Let d = depth of water,
2θ = angle subtended at the centre by water surface,
R = radius of channel, and
i = slope of the bed,

The velocity of flow according to Chezy's formula is given as

$$V = C\sqrt{mi} = C\sqrt{\frac{A}{P}\,i} \quad \left(\because \quad m = \frac{A}{P}\right)$$

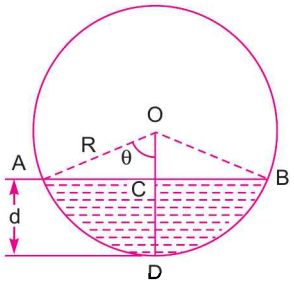


Fig. 16.24

The velocity of flow through a circular channel will be maximum when the hydraulic mean depth m or A/P is maximum for a given value of C and i. In case of circular pipe, the variable is θ only. Hence for maximum value of A/P we have the condition,

$$\frac{d\left(\frac{A}{P}\right)}{d\theta} = 0 \qquad \ldots(i)$$

where A and P both are functions of θ.

The value of wetted area, A is given by equation (16.17) as

$$A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right) \qquad \ldots(ii)$$

The value of wetted perimeter, P is given by equation (16.16) as

$$P = 2R\theta \qquad \ldots(iii)$$

Differentiating equation (i) with respect to θ, we have

$$\frac{P\frac{dA}{d\theta} - A\frac{dP}{d\theta}}{P^2} = 0 \quad \text{or} \quad P\frac{dA}{d\theta} - A\frac{dP}{d\theta} = 0 \qquad \ldots(iv)$$

From equation (ii), $\quad \dfrac{dA}{d\theta} = R^2\left(1 - \dfrac{\cos 2\theta}{2} \times 2\right) = R^2(1 - \cos 2\theta)$

From equation (iii), $\quad \dfrac{dP}{d\theta} = 2R$

Substituting the values of A, $P\dfrac{dA}{d\theta}$ and $\dfrac{dP}{d\theta}$ in equation (iv),

$$2R\theta\left[R^2(1 - \cos 2\theta)\right] - R^2\left(\theta - \frac{\sin 2\theta}{2}\right)(2R) = 0$$

or $$2R^3\theta(1 - \cos 2\theta) - 2R^3\left(\theta - \frac{\sin 2\theta}{2}\right) = 0$$

or $$\theta(1 - \cos 2\theta) - \left(\theta - \frac{\sin 2\theta}{2}\right) = 0 \qquad \text{(Cancelling } 2R^3\text{)}$$

$$\theta - \theta \cos 2\theta - \theta + \frac{\sin 2\theta}{2} = 0$$

or $$\theta \cos 2\theta = \frac{\sin 2\theta}{2} \quad \text{or} \quad \frac{\sin 2\theta}{\cos 2\theta} = 2\theta$$

$\therefore$ $$\tan 2\theta = 2\theta$$

The solution of this equation by hit and trial, gives

$$2\theta = 257^\circ\ 30' \qquad \text{(approximately)}$$

or $$\theta = 128^\circ\ 45'$$

The depth of flow for maximum velocity from Fig. 16.24, is

$$d = OD - OC = R - R\cos\theta$$
$$= R[1 - \cos\theta] = R[1 - \cos 128^\circ\ 45'] = R[1 - \cos(180^\circ - 51^\circ\ 15')]$$
$$= R[1 - (-\cos 51^\circ\ 15')] = R[1 + \cos 51^\circ\ 15']$$
$$= R[1 + 0.62] = 1.62\ R = 1.62 \times \frac{D}{2} = 0.81\ D \qquad ...(16.18)$$

where D = diameter of the circular channel.

Thus for maximum velocity of flow, the depth of water in the circular channel should be equal to 0.81 times the diameter of the channel.

Hydraulic mean depth for maximum velocity is

$$m = \frac{A}{P} = \frac{R^2\left(\theta - \frac{\sin 2\theta}{2}\right)}{2R\theta} = \frac{R}{2\theta}\left[\theta - \frac{\sin 2\theta}{2}\right]$$

where $\theta = 128^\circ\ 45' = 128.75^\circ$

$$= 128.75 \times \frac{\pi}{180} = 2.247 \text{ radians}$$

$\therefore$ $$m = \frac{R}{2 \times 2.247}\left[2.247 - \frac{\sin 257^\circ\ 30'}{2}\right] = \frac{R}{4.494}\left[2.247 - \frac{\sin(180^\circ + 87.5^\circ)}{2}\right]$$

$$= \frac{R}{4.494}\left[2.247 + \frac{\sin 87.5^\circ}{2}\right] = 0.611\ R$$

$$= 0.611 \times \frac{D}{2} = 0.3055\ D = 0.3\ D \qquad ...(16.19)$$

Thus for maximum velocity, the hydraulic mean depth is equal to 0.3 times the diameter of circular channel.

2. Condition for Maximum Discharge for Circular Section. The discharge through a channel is given by

$$Q = AC\sqrt{mi} = AC\sqrt{\frac{A}{P}i} \qquad \left(\because\ m = \frac{A}{P}\right)$$

$$= C\sqrt{\frac{A^3}{P}i}$$

The discharge will be maximum for constant values of C and i, when $\frac{A^3}{P}$ is maximum. $\frac{A^3}{P}$ will be maximum when $\frac{d}{d\theta}\left(\frac{A^3}{P}\right) = 0$.

Differentiating this equation with respect to θ and equation the same to zero, we get

$$\frac{P \times 3A^2 \frac{dA}{d\theta} - A^3 \frac{dP}{d\theta}}{P^2} = 0 \quad \text{or} \quad 3PA^2 \frac{dA}{d\theta} - A^3 \frac{dP}{d\theta} = 0$$

Dividing by A^2, $3P \frac{dA}{d\theta} - A \frac{dP}{d\theta} = 0$ ...(*i*)

But from equation (16.16), $P = 2R\theta$

$$\therefore \quad \frac{dP}{d\theta} = 2R$$

From equation (16.17), $A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right)$

$$\therefore \quad \frac{dA}{d\theta} = R^2 (1 - \cos 2\theta)$$

Substituting the values of P, A, $\frac{dP}{d\theta}$ and $\frac{dA}{d\theta}$ in equation (*i*)

$$3 \times 2R\theta \times R^2 (1 - \cos 2\theta) - R^2\left(\theta - \frac{\sin 2\theta}{2}\right) \times 2R = 0$$

$$6R^3\theta (1 - \cos 2\theta) - 2R^3\left(\theta - \frac{\sin 2\theta}{2}\right) = 0$$

Dividing by $2R^3$, we get

$$3\theta (1 - \cos 2\theta) - \left(\theta - \frac{\sin 2\theta}{2}\right) = 0 \quad \text{or} \quad 3\theta - 3\theta \cos 2\theta - \theta + \frac{\sin 2\theta}{2} = 0$$

or $$2\theta - 3\theta \cos 2\theta + \frac{\sin 2\theta}{2} = 0 \quad \text{or} \quad 4\theta - 6\theta \cos 2\theta + \sin 2\theta = 0$$

The solution of this equation by hit and trial, gives

$$2\theta = 308° \qquad \text{(approximately)}$$

$$\therefore \quad \theta = \frac{308°}{2} = 154°$$

Depth of flow for maximum discharge [See Fig. 16.24]

$$d = OD - OC = R - R \cos \theta$$
$$= R[1 - \cos \theta] = R[1 - \cos 154°]$$
$$= R[1 - \cos (180° - 26°)] = R[1 + \cos 26°] = 1.898\, R$$
$$= 1.898 \times \frac{D}{2} = 0.948\, D \simeq 0.95\, D \qquad ...(16.20)$$

where D = Diameter of the circular channel.

Thus for maximum discharge through a circular channel the depth of flow is equal to 0.95 times its diameter.

Problem 16.30 *The rate of flow of water through a circular channel of diameter 0.6 m is 150 litres/s. Find the slope of the bed of the channel for maximum velocity. Take C = 60*

Solution. Given :

Discharge, $Q = 150$ litres/s $= 0.15$ m^3/s

Dia. of channel, $D = 0.6$ m

Value of $C = 60$

Let the slope of the bed of channel for maximum velocity $= i$

For maximum velocity through a circular channel, depth of flow is given by equation (16.18) as

$$d = 0.81 \times D = 0.81 \times 0.6 = .486 \text{ m}$$

and $$\theta = 128° 45' \quad \text{or} \quad 128.75 \times \frac{\pi}{180} = 2.247 \text{ radians}$$

From equation (16.19), hydraulic mean depth for maximum velocity is given as

$$m = 0.3 \times D = 0.3 \times 0.6 = 0.18 \text{ m}$$

Wetted perimeter for circular pipe is given by equation (16.16),

$$P = 2R\theta = D \times \theta = 0.6 \times 2.247 = 1.3482 \text{ m}$$

But $$m = \frac{A}{P} = 0.18 \text{ m}$$

$\therefore$ Area, $$A = 0.18 \times P = 0.18 \times 1.3482 = 0.2426 \text{ m}^2$$

For discharge, using the relation

$$Q = AC\sqrt{mi} \quad \text{or} \quad 0.15 = 0.2426 \times 60 \times \sqrt{0.81 \times i} = 6.175\sqrt{i}$$

$\therefore$ $$i = \left(\frac{0.15}{6.175}\right)^2 = \frac{\mathbf{1}}{\mathbf{1694.7}} \textbf{ . Ans.}$$

$\therefore$ Bed slope is 1 in 1694.7.

Problem 16.31 *Determine the maximum discharge of water through a circular channel of diameter 1.5 m when the bed slope of the channel is 1 in 1000. Take C = 60.*

Solution. Given :

Dia. of channel, $D = 1.5$ m

$\therefore$ $$R = \frac{1.5}{2} = 0.75 \text{ m}$$

Bed slope, $$i = \frac{1}{1000}$$

Value of $C = 60$

For maximum discharge, $$\theta = 154° \text{ or } \frac{154 \times \pi}{180} = 2.6878 \text{ radians.}$$

Wetted perimeter for a circular channel is given by equation (16.16) as

$$P = 2R\theta = 2 \times \frac{D}{2} \times 2.6878 = 2 \times \frac{1.5}{2} \times 2.6878 = 4.0317 \text{ m}$$

Wetted area A is given by equation (16.17) as

$$A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right) = 0.75^2\left[2.6878 - \frac{\sin (2 \times 154)°}{2}\right]$$

$$= 0.75^2\left[2.6878 - \frac{\sin 308°}{2}\right] = .75^2\left[2.6878 - \frac{\sin (360° - 52°)}{2}\right]$$

$$= 0.75^2\left[2.6878 + \frac{\sin 52°}{2}\right] = 1.7335$$

$\therefore$ Hydraulic mean depth, $m = \frac{A}{P} = \frac{1.7335}{4.0317} = 0.4299$

Maximum discharge is given by $Q = AC\sqrt{mi} = 1.7335 \times 60 \times \sqrt{0.4299 \times \frac{1}{1000}}$

$= \mathbf{2.1565\ m^3/s.\ Ans.}$

Problem 16.32 *A concrete lined circular channel of diameter 3 m has a bed slope of 1 in 500. Work out the velocity and flow rate for the conditions of (i) maximum velocity and (ii) maximum discharge. Assume Chezy's C = 50.*

Solution. Given :

Dia of channel, $D = 3$ m

Bed slope, $i = \frac{1}{500}$

Value of $C = 50$

(*i*) *Velocity and discharge for maximum velocity*

For maximum velocity, $\theta = 128° 45' = 128.75°$

$= 128.75 \times \frac{\pi}{180}$ radians $= 2.247$ radians

$\therefore$ Wetted perimeter, $P = 2 \times R \times \theta$

$= 2 \times 1.5 \times 2.247 \qquad \left(\because\ R = \frac{D}{2} = \frac{3}{2} = 1.5\right)$

$= 6.741$ m

Area of flow, $A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right) = 1.5^2\left[2.247 - \frac{\sin (2 \times 128.75°)}{2}\right]$

$= 2.25\ [2.247 - (-0.488)] = 6.1537\ m^2$

$\therefore$ Hydraulic mean depth $m^* = \frac{A}{P} = \frac{6.1537}{6.741} = 0.912$

Now velocity, $V = C\sqrt{m \times i} = 50 \times \sqrt{\frac{0.912 \times 1}{500}} = \mathbf{2.135\ m/s.\ Ans.}$

and discharge, $Q = A \times V = 6.1537 \times 2.135 = \mathbf{13.138\ m^3/s.}$ **Ans.**

(*ii*) *Velocity and discharge for maximum discharge*

For maximum discharge, $\theta = 154° = \frac{154 \times \pi}{180}$ radians $= 2.6878$ radians

---

* From equation (16.19), m is also equal to $0.3055\ D$.

Hence $m = 0.3055 \times 3 = 0.9165$

$$\therefore \quad A = R^2\left[\theta - \frac{\sin 2\theta}{2}\right] = 1.5^2\left[2.6878 - \frac{\sin(2\times 154)}{2}\right]$$

$$= 2.25\ [2.6878 - (-0.394)] = 6.934$$

$$P = 2R \times \theta = 2 \times 1.5 \times 2.6878 = 8.0634$$

and $$m = \frac{A}{P} = \frac{6.934}{8.0634} = 0.8599$$

Now velocity $$V = C\sqrt{mi} = 50 \times \sqrt{0.8599 \times \frac{1}{500}} = \mathbf{2.0735\ m/s.\ Ans.}$$

and discharge, $$Q = A \times V = 6.934 \times 2.0735 = \mathbf{14.377\ m^3/s.\ Ans.}$$

▶ 16.6 NON-UNIFORM FLOW THROUGH OPEN CHANNELS

We have defined uniform flow and non-uniform flow in Art. 16.2.2. A flow is said to be uniform if the velocity of flow, depth of flow, slope of the bed of the channel and area of cross-section remain constant for a given length of the channel. On the other hand, if velocity of flow, depth of flow, area of cross-section and slope of the bed of channel do not remain constant for a given length of pipe, the flow is said to be non-uniform.

Non-uniform is further divided into Rapidly Varied Flow (R.V.F.), and Gradually Varied Flow (G.V.F.) depending upon the change of depth of flow over the length of the channel. If the depth of flow changes abruptly over a small length of the channel, the flow is said as rapidly varied flow. And if the depth of flow in a channel changes gradually over a long length of channel, the flow is said to be gradually varied flow.

▶ 16.7 SPECIFIC ENERGY AND SPECIFIC ENERGY CURVE

The total energy of a flowing liquid per unit weight is given by,

Total Energy $$= z + h + \frac{V^2}{2g}$$

where z = Height of the bottom of channel above datum,
h = Depth of liquid, and V = Mean velocity of flow.

If the channel bottom is taken as the datum as shown in Fig. 16.25, then the total energy per unit weight of liquid will be,

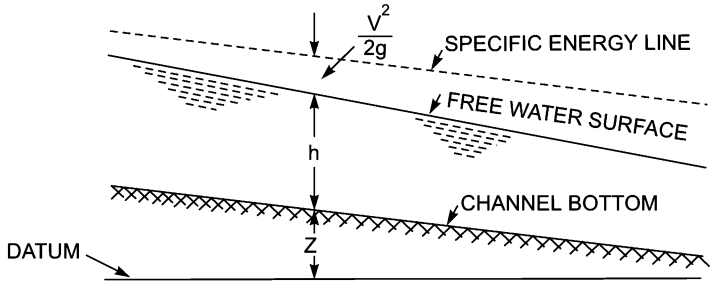


Fig. 16.25 *Specific energy.*

$$E = h + \frac{V^2}{2g} \qquad ...(16.21)$$

The energy given by equation (16.21) is known as **specific energy**. Hence specific energy of a flowing liquid is defined as energy per unit weight of the liquid with respect to the bottom of the channel.

Specific Energy Curve. It is defined as the curve which shows the variation of specific energy with depth of flow. It is obtained as :

From equation (16.21), the specific energy of a flowing liquid

$$E = h + \frac{V^2}{2g} = E_p + E_k$$

where E_p = Potential energy of flow = h

E_k = Kinetic energy of flow $= \frac{V^2}{2g}$

Consider a rectangular channel in which a steady but non-uniform flow is taking place.

Let Q = discharge through the channel,
b = width of the channel,
h = depth of flow, and
q = discharge per unit width.

Then $q = \frac{Q}{\text{width}} = \frac{Q}{b}$ = constant ($\because$ Q and b are constant)

Velocity of flow, $V = \frac{\text{Discharge}}{\text{Area}} = \frac{Q}{b \times h} = \frac{q}{h}$ $\left(\because \frac{Q}{b} = q\right)$

Substituting the values of V in equation (16.21), we get

$$E = h + \frac{q^2}{2gh^2} = E_p + E_k \qquad ...(16.22)$$

Equation (16.22), gives the variation of specific energy (E) with the depth of flow (h). Hence for a given discharge Q, for different values of depth of flow, the corresponding values of E may be obtained. Then a graph between specific energy (along X-X axis) and depth of flow, h (along Y-Y axis) may be plotted.

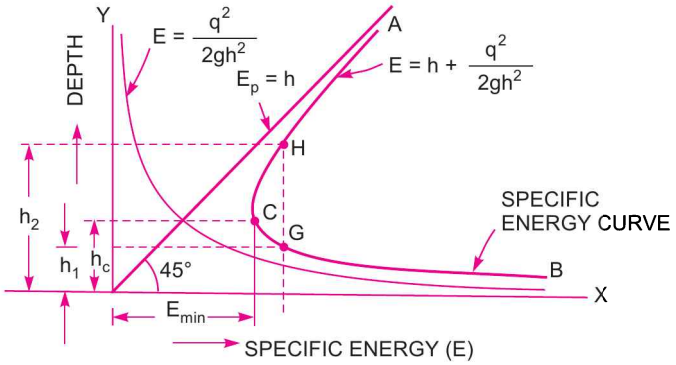


Fig. 16.26 *Specific energy curve.*

The specific energy curve may also be obtained by first drawing a curve for potential energy (*i.e.*, $E_p = h$), which will be a straight line passing through the origin, making an angle of 45° with the X - axis as shown in Fig. 16.26. Then drawing another curve for kinetic energy $\left(i.e., E_k = \frac{q^2}{2gh^2} \text{ or } E_k = \frac{K}{h^2}, \right.$ $\left. \text{where } K = \frac{q^2}{2g} = \text{constant} \right)$ which will be a parabola as shown in Fig. 16.26. By combining these two curves, we can obtain the specific energy curve. In Fig. 16.26, curve ACB denotes the specific energy curve.

16.7.1 Critical Depth (h_c). Critical depth is defined as that depth of flow of water at which the specific energy is minimum. This is denoted by 'h_c' . In Fig. 16.26, curve ACB is a specific energy curve and point C corresponds to the minimum specific energy. The depth of flow of water at C is known as critical depth. The mathematical expression for critical depth is obtained by differentiating the specific energy equation (16.22) with respect to depth of flow and equating the same to zero.

or $$\frac{dE}{dh} = 0, \qquad \text{where } E = h + \frac{q^2}{2gh^2} \text{ from equation (16.22)}$$

or $$\frac{d}{dh}\left[h + \frac{q^2}{2gh^2}\right] = 0 \quad \text{or} \quad 1 + \frac{q^2}{2g}\left(\frac{-2}{h^3}\right) = 0 \qquad \left(\because \frac{q^2}{2g} \text{ is constant}\right)$$

or $$1 - \frac{q^2}{gh^3} = 0 \quad \text{or} \quad 1 = \frac{q^2}{gh^3} \quad \text{or} \quad h^3 = \frac{q^2}{g}$$

$\therefore$ $$h = \left(\frac{q^2}{g}\right)^{1/3}$$

But when specific energy is minimum, depth is critical and it is denoted by h_c. Hence critical depth is

$$h_c = \left(\frac{q^2}{g}\right)^{1/3} \qquad ...(16.23)$$

16.7.2 Critical Velocity (V_c). The velocity of flow at the critical depth is known as critical velocity. It is denoted by V_c. The mathematical expression for critical velocity is obtained from equation (16.23) as

$$h_c = \left(\frac{q^2}{g}\right)^{1/3}$$

Taking cube to both sides, we get $h_c^3 = \frac{q^2}{g}$ or $gh_c^3 = q^2$ ...(i)

But $$q = \text{Discharge per unit width} = \frac{Q}{b}$$

$$= \frac{\text{Area} \times V}{b} = \frac{b \times h \times V}{b} = h \times V = h_c \times V_c$$

Substituting this value of q in (i),

$\therefore \quad gh_c^3 = (h_c \times V_c)^2$

or $\quad gh_c^3 = h_c^2 \times V_c^2$ or $gh_c = V_c^2$ [Dividing by h_c^2]

or $\quad V_c = \sqrt{g \times h_c}$ ...(16.24)

16.7.3 Minimum Specific Energy in Terms of Critical Depth. Specific energy equation is given by equation (16.22)

$$E = h + \frac{q^2}{2gh^2}$$

When specific energy is minimum, depth of flow is critical and hence above equation becomes as

$$E_{min} = h_c + \frac{q^2}{2gh_c^2} \qquad ...(ii)$$

But from equation (16.23), $h_c = \left(\frac{q^2}{g}\right)^{1/3}$ or $h_c^3 = \frac{q^2}{g}$

Substituting the value of $\frac{q^2}{g} = h_c^3$ in equation (ii), we get

$$E_{min} = h_c + \frac{h_c^3}{2h_c^2} = h_c + \frac{h_c}{2} = \frac{3h_c}{2}. \qquad ...(16.25)$$

Problem 16.33 *Find the specific energy of flowing water through a rectangular channel of width 5 m when the discharge is 10 m^3/s and depth of water is 3 m.*

Solution. Given :

Width of rectangular channel, $b = 5$ m

Discharge, $Q = 10$ m^3/s

Depth of water, $h = 3$ m

Specific energy is given by equation (16.21), as

$$E = h + \frac{V^2}{2g}, \quad \text{where } V = \frac{Q}{\text{Area}} = \frac{10}{b \times h} = \frac{10}{5 \times 3} = \frac{2}{3}$$

$\therefore$

$$E = 3 + \frac{\left(\frac{2}{3}\right)^2}{2 \times 9.81} = 3 + .0226 = \mathbf{3.0226\ m.\ Ans.}$$

Problem 16.34 *Find the critical depth and critical velocity of the water flowing through a rectangular channel of width 5 m, when discharge is 15 m^3/s.*

Solution. Given :

Width of channel, $b = 5$ m

Discharge, $Q = 15$ m^3/s

$\therefore$ Discharge per unit width, $q = \frac{Q}{b} = \frac{15}{5} = 3$ m^2/s

Critical depth is given by equation (16.23) as

$$h_c = \left(\frac{q^2}{g}\right)^{1/3} = \left(\frac{3^2}{9.81}\right)^{1/3} = \left(\frac{9}{9.81}\right)^{1/3} = \mathbf{0.972\ m.\ Ans.}$$

Critical velocity is given by equation (16.24) as

$$V_c = \sqrt{g \times h_c} = \sqrt{9.81 \times 0.972} = \mathbf{3.088\ m/s.\ Ans.}$$

Problem 16.35 *The discharge of water through a rectangular channel of width 8 m, is 15 m^3/s when depth of flow of water is 1.2 m. Calculate :*

(i) Specific energy of the flowing water, (ii) Critical depth and critical velocity,

(iii) Value of minimum specific energy.

Solution. Given :

Discharge, $Q = 15\ m^3/s$

Width, $b = 8\ m$

Depth, $h = 1.2\ m$

$\therefore$ Discharge per unit width, $q = \frac{Q}{b} = \frac{15}{8} = 1.875\ m^2/s$

Velocity of flow, $V = \frac{Q}{\text{Area}} = \frac{15}{b \times h} = \frac{15.0}{8 \times 1.2} = 1.5625\ m/s$

(*i*) Specific energy (E) is given by equation (16.21) as

$$E = h + \frac{V^2}{2g} = 1.2 + \frac{1.5625^2}{8 \times 9.81} = 1.20 + 0.124 = \mathbf{1.324\ m.\ Ans.}$$

(*ii*) Critical depth (h_c) is given by equation (16.23) as

$$h_c = \left(\frac{q^2}{g}\right)^{1/3} = \left(\frac{1.875^2}{9.81}\right)^{1/3} = \mathbf{0.71\ m.\ Ans.}$$

Critical velocity (V_c) is given by equation (16.24) as

$$V_c = \sqrt{g \times h_c} = \sqrt{9.81 \times 0.71} = \mathbf{2.639\ m/s.\ Ans.}$$

(*iii*) Minimum specific energy (E_{min}) is given by equation (16.25)

$$E_{min} = \frac{3h_c}{2} = \frac{3 \times 0.71}{2} = \mathbf{1.065\ m.\ Ans.}$$

16.7.4 Critical Flow. It is defined as that flow at which the specific energy is minimum or the flow corresponding to critical depth is defined as critical flow. Equation (16.24) gives the relation for critical velocity in terms of critical depth as

$$V_c = \sqrt{g \times h_c} \quad \text{or} \quad \frac{V_c}{\sqrt{g h_c}} = 1 \qquad \left| \text{where } \frac{V_c}{\sqrt{g h_c}} = \text{Froude number} \right.$$

$\therefore$ Froude number, $F_e = 1.0$ for critical flow.

16.7.5 Streaming Flow or Sub-critical Flow or Tranquil Flow. When the depth of flow in a channel is greater than the critical depth (h_c), the flow is said to be sub-critical flow or streaming flow or tranquil flow. For this type of flow the Froude number is less than one *i.e.*, $F_e < 1.0$.

16.7.6 Super-critical Flow or Shooting Flow or Torrential Flow. When the depth of flow in a channel is less than the critical depth (h_c), the flow is said to be super-critical flow or shooting flow or torrential flow. For this type of flow the Froude number is greater than one *i.e.*, $F_e > 1.0$.

16.7.7 Alternate Depths. In the specific energy curve shown in Fig. 16.26, the point C corresponds to the minimum specific energy and the depth of flow at C is called critical depth. For any other value of the specific energy, there are two depths, one greater than the critical depth and other smaller than the critical depth. These two depths for a given specific energy are called the alternate depths. These depths are shown as h_1 and h_2 in Fig. 16.26. Or the depths corresponding to points G and H in Fig. 16.26 are called alternate depths.

16.7.8 Condition for Maximum Discharge for a Given Value of Specific Energy. The specific energy (E) at any section of a channel is given by equation (16.21) as

$$E = h + \frac{V^2}{2g}, \text{ where } V = \frac{Q}{A} = \frac{Q}{b \times h}$$

$$\therefore \quad E = h + \frac{Q^2}{b^2 \times h^2} \times \frac{1}{2g} = h + \frac{Q^2}{2gb^2h^2}$$

or $$Q^2 = (E - h)\, 2gb^2h^2 \quad \text{or} \quad Q = \sqrt{(E-h)2gb^2h^2} = b\sqrt{2g\left(Eh^2 - h^3\right)}$$

For maximum discharge, Q, the expression $(Eh^2 - h^3)$ should be maximum. Or in other words,

$$\frac{d}{dh}(Eh^2 - h^3) = 0 \quad \text{or} \quad 2Eh - 3h^2 = 0 \qquad (\because \ E \text{ is constant})$$

or $$2E - 3h = 0 \qquad \text{(Dividing by } h\text{)}$$

or $$h = \tfrac{2}{3}E \qquad ...(16.26)$$

or $$E = \frac{3h}{2} \qquad ...(i)$$

But from equation (16.25), specific energy is minimum when it is equal to $\frac{3}{2}$ times the value of depth of critical flow. Here in equation (i), the specific energy (E) is equal to $\frac{3}{2}$ times the depth of flow. Thus equation (i) represents the minimum specific energy and h is the critical depth. Hence the condition for maximum discharge for given value of specific energy is that the depth of flow should be critical.

Problem 16.36 *The specific energy for a 3 m wide channel is to be 3 kg-m/kg. What would be the maximum possible discharge ?*

Solution. Given :

Width of channel, $b = 3$ m

Specific energy, $E = 3$ kg-m/kg $= 3$ m

For the given value of specific energy, the discharge will be maximum, when depth of flow is critical. Hence from equation (16.26) for maximum discharge.

$$h_c = h = \tfrac{2}{3}\text{E} = \tfrac{2}{3} \times 3.0 = 2.0 \text{ m}$$

$\therefore$ Maximum discharge, Q_{max} is given by

$$Q_{max} = \text{Area} \times \text{Velocity} = (b \times \text{depth of flow}) \times \text{Velocity}$$
$$= (b \times h_c) \times V_c \quad (\because \text{ At critical depth, Velocity will be critical})$$

where V_c is critical velocity and it is given by equation (16.24),

$$V_c = \sqrt{g \times h_c} = \sqrt{9.81 \times 2.0} = 4.4249 \text{ m/s}$$

$\therefore$ $$Q_{max} = (b \times h_c) \times V_c = (3 \times 2) \times 4.4249 = \mathbf{26.576\ m^3/s.\ Ans.}$$

Problem 16.37 *The specific energy for a 5 m wide rectangular channel is to be 4 Nm/N. If the rate of flow of water through the channel is 20 m^3/s, determine the alternate depths of flow.*

Solution. Given :

Width of channel, $b = 5$ m

Specific energy, $E = 4$ Nm/N $= 4$ m

Discharge, $Q = 20\ m^3/s$

The specific energy (E) is given by equation (16.21) as,

$$E = h + \frac{V^2}{2g}, \quad \text{where } V = \frac{\text{Discharge}}{\text{Area}} = \frac{Q}{b \times h} = \frac{20}{5 \times h} = \frac{4}{h}$$

$\therefore$ Specific energy, $$E = h + \frac{V^2}{2g} = h + \left(\frac{4}{h}\right)^2 \times \frac{1}{2g} = h + \frac{8}{g \times h^2}$$

But $E = 4.0$

Equating the two values of E, $4 = h + \frac{8}{9.81 \times h^2} = h + \frac{0.8155}{h^2}$

$$4h^2 = h^3 + .8155 \quad \text{or} \quad h^3 - 4h^2 + .8155 = 0$$

This is a cubic equation. Solving by trial and error, we get

$h =$ **3.93 m and 0.48 m. Ans.**

▶ 16.8 HYDRAULIC JUMP OR STANDING WAVE

Consider the flow of water over a dam as shown in Fig. 16.27. The height of water at the section 1-1 is small. As we move towards downstream, the height or depth of water increases rapidly over a short length of the channel. This is because at the section 1-1, the flow is a *shooting flow* as the depth of water at section 1-1 is less than critical depth. Shooting flow is an unstable type of flow and does not continue on the downstream side. Then this shooting will convert itself into a streaming or tranquil flow and hence depth of water will increase. This sudden increase of depth of water is called a hydraulic jump or a standing wave. Thus hydraulic jump is defined as :

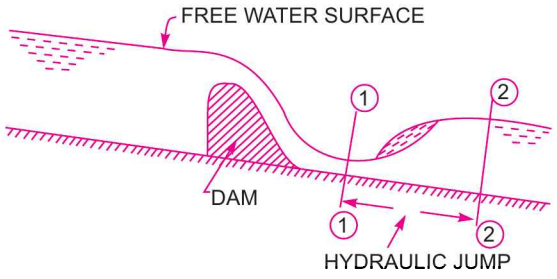


Fig. 16.27 *Hydraulic jump.*

"The rise of water level, which takes place due to the transformation of the unstable shooting flow (Super-critical) to the stable streaming flow (sub-critical flow)."

When hydraulic jump takes place, a loss of energy due to eddy formation and turbulence occurs.

16.8.1 Expression for Depth of Hydraulic Jump. Before deriving an expression for the depth of hydraulic jump, the following assumptions are made :

1. The flow is uniform and pressure distribution is due to hydrostatic before and after the jump.
2. Losses due to friction on the surface of the bed of the channel are small and hence neglected.
3. The slope of the bed of the channel is small, so that the component of the weight of the fluid in the direction of flow is negligibly small.

Consider a hydraulic jump formed in a channel of horizontal bed as shown in Fig. 16.28. Consider two sections 1-1 and 2-2 before and after hydraulic jump.

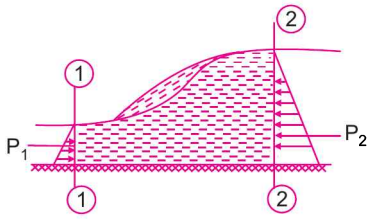


Fig. 16.28 *Hydraulic jump.*

Let d_1 = Depth of flow at section 1-1,
d_2 = Depth of flow at section 2-2,
V_1 = Velocity of flow at section 1-1,
V_2 = Velocity of flow at section 2-2,
$\bar{Z}_1$ = Depth of centroid of area at section 1-1 below free surface,
$\bar{Z}_2$ = Depth of centroid of area at section 2-2 below free surface,
A_1 = Area of cross-section at section 1-1, and
A_2 = Area of cross-section at section 2-2.

Consider unit width of the channel.

The forces acting on the mass of water between sections 1-1 and 2-2 are :

(*i*) Pressure force, P_1 on section 1-1,
(*ii*) Pressure force, P_2 on section 2-2,
(*iii*) Frictional force on the floor of the channel, which is assumed to be negligible.

Let q = discharge per unit width

$$= V_1 d_1 = V_2 d_2 \qquad ...(i)$$

Now pressure force P_1 on section 1-1

$$= \rho g A_1 \bar{Z}_1 = \rho g \times d_1 \times 1 \times \frac{d_1}{2}$$

$$= \frac{\rho g d_1^2}{2} \qquad \left(\because \quad A_1 = d_1 \times 1, \bar{Z}_1 = \frac{d_1}{2}\right)$$

Similarly pressure force on section 2-2,

$$P_2 = \rho g A_2 \bar{Z}_2$$

$$= \rho g \times d_2 \times 1 \times \frac{d_2}{2} = \frac{\rho g d_2^2}{2}$$

Net force acting on the mass of water between sections 1-1 and 2-2

$$= P_2 - P_1 \qquad | \because \quad P_2 \text{ is greater than } P_1 \text{ and } d_2 \text{ is greater than } d_1$$

$$= \frac{\rho g d_2^2}{2} - \frac{\rho g d_1^2}{2} = \frac{\rho g}{2} [d_2^2 - d_1^2] \qquad ...(ii)$$

But from momentum principle, the net force acting on a mass of fluid must be equal to the rate of change of momentum in the same section.

∴ Rate of change of momentum in the direction of force
= mass of water per sec × change of velocity in direction of force

Now mass of water per second = ρ × discharge per unit width × width
= ρ × q × 1 = ρq m^3/s

Change of velocity in the direction of force = $(V_1 - V_2)$

[as net force is acting from right to left, the change of velocity should be taken from right to left and hence is equal to $(V_1 - V_2)$]

∴ Rate of change of momentum in the direction of force = $\rho q(V_1 - V_2)$ ...(*iii*)

Hence according to momentum principle, the expression given by equation (*ii*) is equal to the expression given by equation (*iii*)

or $$\frac{\rho g}{2}(d_2^2 - d_1^2) = \rho q(V_1 - V_2)$$

But from equation (*i*), $V_1 = \frac{q}{d_1}$ and $V_2 = \frac{q}{d_2}$

∴ $$\frac{\rho g}{2}(d_2^2 - d_1^2) = \rho q\left(\frac{q}{d_1} - \frac{q}{d_2}\right)$$

or $$\frac{g}{2}(d_2 + d_1)(d_2 - d_1) = q^2\left(\frac{d_2 - d_1}{d_1 d_2}\right)$$ (Dividing by ρ)

or $$\frac{g}{2}(d_2 + d_1) = \frac{q^2}{d_1 d_2}$$ [Dividing by$(d_2 - d_1)$]

or $$(d_2 + d_1) = \frac{2q^2}{g d_1 d_2}$$ ...(*iv*)

Multiplying both sides by d_2, we get

$$d_2^2 + d_1 d_2 = \frac{2q^2}{g d_1} \quad \text{or} \quad d_2^2 + d_1 d_2 - \frac{2q^2}{g d_1} = 0$$ ...(*v*)

Equation (*v*) is a quadratic equation in d_2 and hence its solution is

$$d_2 = \frac{-d_1 \pm \sqrt{d_1^2 - 4 \times 1 \times \left(\frac{-2q^2}{g d_1}\right)}}{2 \times 1}$$

$$= \frac{-d_1 \pm \sqrt{d_1^2 + \frac{8q^2}{g d_1}}}{2} = -\frac{d_1}{2} \pm \sqrt{\frac{d_1^2}{4} + \frac{2q^2}{g d_1}}$$

The two roots of the equation are $-\frac{d_1}{2}-\sqrt{\frac{d_1^2}{4}+\frac{2q^2}{gd_1}}$ and $-\frac{d_1}{2}+\sqrt{\frac{d_1^2}{4}+\frac{2q^2}{gd_1}}$

First root is not possible as it gives –ve depth. Hence

$$d_2 = -\frac{d_1}{2}+\sqrt{\frac{d_1^2}{4}+\frac{2q^2}{gd_1}} \qquad ...(16.27)$$

$$= -\frac{d_1}{2}+\sqrt{\frac{d_1^2}{4}+\frac{2\times(V_1d_1)^2}{gd_1}} \qquad \{\because \quad q_1 = V_1d_1\}$$

$$= -\frac{d_1}{2}+\sqrt{\frac{d_1^2}{4}+\frac{2V_1^2d_1}{g}} \qquad ...(16.28)$$

$\therefore$ Depth of Hydraulic jump $= (d_2 - d_1)$. ...(16.29)

16.8.2 Expression for Loss of Energy Due to Hydraulic Jump. As mentioned in Art. 16.8 that when hydraulic jump takes place, a loss of energy due to eddies formation and turbulence occurs. This loss of energy is equal to the difference of specific energies at sections 1-1 and 2-2.

Or loss of energy due to hydraulic jump,

$$h_L = E_1 - E_2$$

$$= \left(d_1+\frac{V_1^2}{2g}\right)-\left(d_2+\frac{V_2^2}{2g}\right) \qquad \left(\because \quad E_1 = d_1+\frac{V_1^2}{2g} \text{ and so } E_2\right)$$

$$= \left(\frac{V_1^2}{2g}-\frac{V_2^2}{2g}\right)-(d_2-d_1)$$

$$= \left(\frac{q^2}{2gd_1^2}-\frac{q^2}{2gd_2^2}\right)-(d_2-d_1) \qquad \left(\because \quad V_1 = \frac{q}{d_1} \text{ and } V_2 = \frac{q}{d_2}\right)$$

$$= \frac{q^2}{2g}\left[\frac{1}{d_1^2}-\frac{1}{d_2^2}\right]-[d_2-d_1] = \frac{q^2}{2g}\left[\frac{d_2^2-d_1^2}{d_1^2d_2^2}\right]-[d_2-d_1] \qquad ...(vi)$$

But from equation (*iv*), $q^2 = gd_1d_2\frac{(d_2+d_1)}{2}$

Substituting the value of q^2 in equation (*vi*), we get

Loss of energy, $h_L = gd_1d_2\frac{(d_2+d_1)}{2}\times\frac{d_2^2-d_1^2}{2gd_1^2d_2^2}-(d_2-d_1) = \frac{(d_2+d_1)(d_2^2-d_1^2)}{4d_1d_2}-(d_2-d_1)$

$$= \frac{(d_2+d_1)(d_2+d_1)(d_2-d_1)}{4d_1d_2}-(d_2-d_1) = (d_2-d_1)\left[\frac{(d_2+d_1)^2}{4d_1d_2}-1\right]$$

$$= (d_2 - d_1)\left[\frac{d_2^2 + d_1^2 + 2d_1d_2 - 4d_1d_2}{4d_1d_2}\right] = (d_2 - d_1)\frac{[d_2 - d_1]^2}{4d_1d_2}$$

$$\therefore \qquad h_L = \frac{[d_2 - d_1]^3}{4d_1d_2}. \qquad \text{...(16.30)}$$

16.8.3 Expression for Depth of Hydraulic Jump in Terms of Upstream Froude Number.

Let V_1 = Velocity of flow on the upstream side,

and d = Depth of flow on upstream side,

Then Froude Number $(F_e)_1$ on the upstream side of the jump is given by

$$(F_e)_1 = \frac{V_1}{\sqrt{gd_1}} \qquad \text{...(vii)}$$

Now the depth of flow after the hydraulic jump is d_2 and it is given by equation (16.28) as

$$d_2 = -\frac{d_1}{2} + \sqrt{\frac{d_1^2}{4} + \frac{2V_1^2d_1}{g}} = -\frac{d_1}{2} + \sqrt{\frac{d_1^2}{4}\left(1 + \frac{8V_1^2}{gd_1}\right)}$$

$$= -\frac{d_1}{2} + \frac{d_1}{2}\sqrt{1 + \frac{8V_1^2}{gd_1}} \qquad \text{...(viii)}$$

But from equation (*vii*), $(F_e)_1 = \dfrac{V_1}{\sqrt{gd_1}}$ or $(F_e)_1^2 = \dfrac{V_1^2}{gd_1}$

Substituting this value in equation (*viii*), we get

$$d_2 = -\frac{d_1}{2} + \frac{d_1}{2}\sqrt{1 + 8(F_e)_1^2} = \frac{d_1}{2}\left(\sqrt{1 + 8(F_e)_1^2} - 1\right) \qquad \text{...(16.31)}$$

16.8.4 Length of Hydraulic Jump. This is defined as the length between the two sections where one section is taken before the hydraulic jump and the second section is taken immediately after the jump. For a rectangular channel from experiments, it has been found equal to 5 to 7 times the height of the hydraulic jump.

Problem 16.38 *The depth of flow of water, at a certain section of a rectangular channel of 4 m wide, is 0.5 m. This discharge through the channel is 16 m^3/s. If a hydraulic jump takes place on the downstream side, find the depth of flow after the jump.*

Solution. Given :

Width of channel, $b = 4$ m

Depth of flow before jump, $d_1 = 0.5$ m

Discharge, $Q = 16$ m^3/s

$\therefore$ Discharge per unit width, $q = \dfrac{Q}{b} = \dfrac{16}{4} = 4$ m^2/s

Let the depth of flow after jump = d_2

Depth of flow after the jump is given by equation (16.27), as

$$d_2 = -\frac{d_1}{2} + \sqrt{\frac{d_1^2}{4} + \frac{2q^2}{gd_1}} = -\frac{0.5}{2} + \sqrt{\frac{0.5^2}{4} + \frac{2 \times 4^2}{9.81 \times 0.5}}$$

$$= -0.25 + \sqrt{0.0625 + 6.5239} = -0.25 + 2.566 = \mathbf{2.316\ m.\ Ans.}$$

Problem 16.39 *The depth of flow of water, at a certain section of a rectangular channel of 2 m wide, is 0.3 m. The discharge through the channel is 1.5 m^3/s. Determine whether a hydraulic jump will occur, and if so, find its height and loss of energy per kg of water.*

Solution. Given :

Depth of flow, $d_1 = 0.3$ m

Width of channel, $b = 2$ m

Discharge, $Q = 1.5$ m^3/s

Discharge per unit width, $q = \frac{Q}{b} = \frac{1.5}{2.0} = 0.75$ m^2/s.

Hydraulic jump will occur if the depth of flow on the upstream side is less than the critical depth on upstream side or if the Froude number on the upstream side is more than one.

Critical depth (h_c) is given by equation (16.23) as

$$h_c = \left(\frac{q^2}{g}\right)^{1/3} = \left(\frac{0.75^2}{9.81}\right)^{1/3} = 0.3859$$

Now the depth on the upstream side is 0.3 m. This depth is less than critical depth and hence hydraulic jump will occur.

The depth of flow after hydraulic jump is given by equation (16.27) as

$$d_2 = -\frac{d_1}{2} + \sqrt{\frac{d_1^2}{4} + \frac{2q^2}{gd_1}} = -\frac{0.3}{2} + \sqrt{\frac{0.3^2}{4} + \frac{2 \times 0.75^2}{9.81 \times 0.3}}$$

$$= -1.5 + \sqrt{.0255 + .3822} = -0.15 + .6362 = 0.4862 \text{ m}$$

$\therefore$ Height of hydraulic jump $= d_2 - d_1 = 0.4862 - 0.30 =$ **0.1862 m. Ans.**

Loss of energy per kg of water due to hydraulic jump is given by equation (16.30) as

$$h_L = \frac{(d_2 - d_1)^3}{4d_1d_2} = \frac{[0.4862 - 0.30]^3}{4 \times .4862 \times .30}$$

$$= \frac{0.1862^3}{4 \times .4862 \times .30} = \textbf{0.01106 m-kg/kg. Ans.}$$

Problem 16.40 *A sluice gate discharges water into a horizontal rectangular channel with a velocity of 10 m/s and depth of flow of 1 m. Determine the depth of flow after the jump and consequent loss in total head.*

Solution. Given :

Velocity of flow before hydraulic jump, $V_1 = 10$ m/s

Depth of flow before hydraulic jump, $d_1 = 1$ m

Discharge per unit width, $q = V_1 \times d_1 = 10 \times 1 = 10$ m^2/s

The depth of flow after jump is given by equation (16.27) as

$$d_2 = -\frac{d_1}{2} + \sqrt{\frac{d_1^2}{4} + \frac{2q^2}{gd_1}} = -\frac{1.0}{2} + \sqrt{\frac{1^2}{4} + \frac{2 \times 10^2}{9.81 \times 1}}$$

$$= -0.50 + \sqrt{0.25 + 20.387} = \textbf{4.043 m. Ans.}$$

Loss in total head is given by equation (16.30) as

$$h_L = \frac{(d_2 - d_1)^3}{4d_1d_2} = \frac{(4.043 - 1.0)^3}{4 \times 1.0 \times 4.043} = \textbf{1.742 m. Ans.}$$

Problem 16.41 *A sluice gate discharges water into a horizontal rectangular channel with a velocity of 6 m/s and depth of flow is 0.4 m. The width of the channel is 8 m. Determine whether a hydraulic jump will occur, and if so, find its height and loss of energy per kg of water. Also determine the power lost in the hydraulic jump.*

Solution. (*i*) Given :

Velocity of flow, $V_1 = 6$ m/s

Depth of flow, $d_1 = 0.4$ m

Width of channel, $b = 8$ m

$\therefore$ Discharge per unit width, $q = \dfrac{Q}{b} = \dfrac{V_1 \times \text{area}}{b} = \dfrac{V_1 \times d_1 \times b}{b}$

$= V_1 \times d_1 = 6 \times 0.4 = 2.4$ m^2/s

Froude number on the upstream side,

$$(F_e)_1 = \frac{V_1}{\sqrt{gd_1}} = \frac{6.0}{\sqrt{9.81 \times 0.4}} = 3.0289 \simeq 3.029.$$

As Froude number is more than one, the flow is shooting on the upstream side. Shooting flow is unstable flow and it will convert itself into streaming flow by raising its height and hence hydraulic jump will take place.

(*ii*) Let the depth of hydraulic jump = d_2

Using equation (16.31), we have

$$d_2 = \frac{d_1}{2}\left(\sqrt{1 + 8(F_e)_1^2} - 1\right) = \frac{0.4}{2}\left(\sqrt{1 + 8 \times 3.029^2} - 1\right) = 1.525 \text{ m}$$

$\therefore$ Height of hydraulic jump $= d_2 - d_1 = 1.525 - 0.4 =$ **1.125 m. Ans.**

(*iii*) Loss of energy per kg of water is given by equation (16.30)

$$h_L = \frac{(d_2 - d_1)^3}{4d_1d_2} = \frac{(1.525 - 0.4)^3}{4 \times 0.4 \times 1.525} = \textbf{0.5835 m-kg/kg of water. Ans.}$$

(*iv*) Power lost in kW $= \dfrac{\rho g \times Q \times h_L}{1000}$, where $Q = V \times$ area

$= V_1 \times d_1 \times b = 6 \times 0.4 \times 8 = 19.2$ m^3/s

$\therefore$ Power, $P = \dfrac{1000 \times 9.81 \times 19.2 \times 0.5835}{1000} =$ **109.9 kW. Ans.**

Problem 16.42 *A hydraulic jump forms at the downstream end of spillway carrying 17.93 m^3/s discharge. If the depth before jump is 0.80 m, determine the depth after the jump and energy loss.*

Solution. Given :

Discharge $Q = 17.93$ m^3/s

Depth before jump, $d_1 = 0.8$ m

Taking width $b = 1$ m, we get

Discharge per unit width, $q = \dfrac{17.93}{1} = 17.93$

Let d_2 = Depth after jump and h_L = Loss of energy.

Using equation (16.27), we get $d_2 = -\dfrac{d_1}{2} + \sqrt{\dfrac{d_1^2}{4} + \dfrac{2q^2}{gd_1}} = -\dfrac{0.8}{2} + \sqrt{\dfrac{0.8^2}{4} + \dfrac{2 \times 17.93^2}{9.81 \times 0.8}}$

$$= -0.4 + \sqrt{0.16 + 81.927} = -0.4 + 9.06 = \textbf{8.66 m. Ans.}$$

Using equation (16.30) for loss of energy,

$$h_L = \frac{(d_2 - d_1)^3}{4 d_1 d_2} = \frac{(8.66 - 0.8)^3}{4 \times 0.8 \times 8.66} = \textbf{17.52 m. Ans.}$$

▶ 16.9 GRADUALLY VARIED FLOW (G.V.F.)

If the depth of flow in a channel changes gradually over a long length of the channel, the flow is said to be gradually varied flow and is denoted by G.V.F.

16.9.1 Equation of Gradually Varied Flow. Before deriving an equation for gradually varied flow, the following assumptions are made :

1. The bed slope of the channel is small,
2. The flow is steady and hence discharge Q is constant,
3. Accelerative effect is negligible and hence hydrostatic pressure distribution prevails over channel cross-section.
4. The energy correction factor, α is unity.
5. The roughness co-efficient is constant for the length of the channel and it does not depend on the depth of flow.
6. The formulae, such as Chezy's formula, Manning's formula, which are applicable, to the uniform flow are also applicable to the gradually varied flow for determining the slope of energy line.
7. The channel is prismatic.

Consider a rectangular channel having gradually varied flow as shown in Fig. 16.29. The depth of flow is gradually decreasing in the direction of flow.

Let Z = height of bottom of channel above datum
h = depth of flow,
V = mean velocity of flow,
i_b = slope of the channel bed,
i_e = slope of the energy line,
b = width of channel, and
Q = discharge through the channel

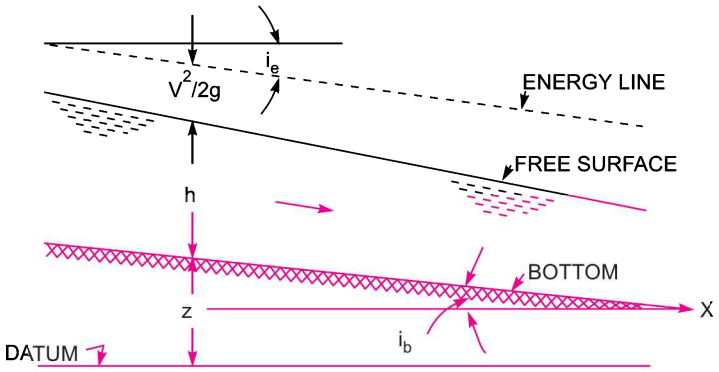


Fig. 16.29 *Equation for gradually varied flow.*

The energy equation at any section is given by Bernoulli's equation,

$$E = Z + h + \frac{V^2}{2g} \qquad ...(i)$$

Differentiating this equation with respect to x, where x is measured along the bottom of the channel in the direction of flow, we get

$$\frac{dE}{dx} = \frac{dZ}{dx} + \frac{dh}{dx} + \frac{d}{dx}\left(\frac{V^2}{2g}\right) \qquad ...(ii)$$

Now $$\frac{d}{dx}\left(\frac{V^2}{2g}\right) = \frac{d}{dx}\left(\frac{Q^2}{A^2 \times 2g}\right) \qquad \left(\because\ V = \frac{Q}{A} = \frac{Q}{b \times h}\right) \text{ (as } A = b \times h\text{)}$$

$$= \frac{d}{dx}\left(\frac{Q^2}{b^2h^2 \times 2g}\right) = \frac{Q^2}{b^2 \times 2g}\frac{d}{dx}\left(\frac{1}{h^2}\right) \text{ (} \because\ Q, b \text{ and } g \text{ are constant)}$$

$$= \frac{Q^2}{b^2 \times 2g}\frac{d}{dh}\left[\frac{1}{h^2}\right]\frac{dh}{dx} = \frac{Q^2}{b^2 \times 2g}\left[\frac{-2}{h^3}\right]\frac{dh}{dx} = \frac{-2Q^2}{b^2 \times 2gh^3}\frac{dh}{dx}$$

$$= -\frac{Q^2}{b^2h^2 \times gh}\frac{dh}{dx} = -\frac{V^2}{gh}\frac{dh}{dx} \qquad \left[\because\ \frac{Q}{bh} = V\right]$$

Substituting the value of $\frac{d}{dx}\left(\frac{V^2}{2g}\right)$ in equation (ii), we get

$$\frac{dE}{dx} = \frac{dZ}{dx} + \frac{dh}{dx} - \frac{V^2}{gh}\frac{dh}{dx} = \frac{dZ}{dx} + \frac{dh}{dx}\left[1 - \frac{V^2}{gh}\right] \qquad ...(iii)$$

Now $$\frac{dE}{dx} = \text{slope of the energy line} = -i_e$$

and $$\frac{dZ}{dx} = \text{slope of the bed of the channel} = -i_b$$

–ve sign with i_e and i_b is taken as with the increase of x, the value of E and Z decreases.

Substituting the value of $\frac{dE}{dx}$ and $\frac{dZ}{dx}$ in equation (iii), we get

$$-i_e = -i_b + \frac{dh}{dx}\left[1 - \frac{V^2}{gh}\right] \quad \text{or} \quad i_b - i_e = \frac{dh}{dx}\left[1 - \frac{V^2}{gh}\right]$$

or $$\frac{dh}{dx} = \frac{(i_b - i_e)}{\left[1 - \frac{V^2}{gh}\right]} \qquad ...(16.32)$$

$$= \frac{(i_b - i_e)}{[1 - (F_e)^2]} \qquad \left[\because\ \frac{V}{\sqrt{gh}} = F_e\right] \qquad ...(16.33)$$

As h is the depth of flow and x is the distance measured along the bottom of the channel hence $\frac{dh}{dx}$ represents the variation of the water depth along the bottom of the channel.

This is also called the slope of the free water surface. Thus :

(*i*) When $\frac{dh}{dx} = 0$, h is constant or depth of the water above the bottom of channel is constant. It means that free surface of water is parallel to the bed of the channel.

(*ii*) When $\frac{dh}{dx} > 0$ or $\frac{dh}{dx}$ is +ve, it means the depth of water increases in the direction of flow. The profile of the water so obtained is called *back water curve.*

(*iii*) When $\frac{dh}{dx} < 0$ or $\frac{dh}{dx}$ is –ve, it means that the depth of water decreases in the direction of flow. The profile of the water so obtained is called *drop down curve.*

Problem 16.43 *Find the rate of change of depth of water in a rectangular channel of 10 m wide and 1.5 m deep, when the water is flowing with a velocity of 1 m/s. The flow of water through the channel of bed slope 1 in 4000, is regulated in such a way that energy line is having a slope of .00004.*

Solution. Given :

Width of channel, $b = 10$ m

Depth of channel, $h = 3$ m

Velocity of flow, $V = 1$ m/s

Bed slope, $i_b = \frac{1}{4000} = .00025$

Slope of energy line, $i_e = .00004$

Let rate of change of depth of water = $\frac{dh}{dx}$

Using equation (16.32) as $\frac{dh}{dx} = \frac{(i_b - i_e)}{\left(1 - \frac{V^2}{gh}\right)} = \frac{.00025 - .00004}{\left(1 - \frac{1 \times 1}{9.81 \times 3}\right)} = \frac{.00021}{.966} =$ **.000217. Ans.**

Problem 16.44 *Find the slope of the free water surface in a rectangular channel of width 20 m, having depth of flow 5 m. The discharge through the channel is 50 m^3/s. The bed of the channel is having a slope of 1 in 4000. Take the value of Chezy's constant C = 60.*

Solution. Given :

Width of channel, $b = 20$ m

Depth of flow, $h = 5$ m

Discharge, $Q = 50$ m^3/s

Bed slope, $i_b = \frac{1}{4000} = .00025$

Chezy's constant, $C = 60$

The discharge, Q is given by $Q = V \times$ Area $= C\sqrt{mi} \times A = AC\sqrt{mi}$

where A = Area of flow $= b \times h = 20 \times 5 = 100$ m^2,

$$m = \text{hydraulic mean depth} = \frac{A}{P} = \frac{100}{b+2h} = \frac{100}{20+2\times5} = \frac{100}{30} = \frac{10}{3} \text{ m},$$

$i = i_e$ = slope of energy line.

The slope of the energy line* is determined from Chezy's formula

$$50 = 100 \times 60 \times \sqrt{\frac{10}{3} \times i_e} = 10954.45 \sqrt{i_e}$$

or
$$i_e = \left(\frac{50}{10954.45}\right)^2 = 0.0000208$$

The slope of free water surface $= \frac{dh}{dx}$

Using equation (16.32) as
$$\frac{dh}{dx} = \frac{i_b - i_e}{1 - \frac{V^2}{gh}} = \frac{0.00025 - 0.0000208}{1 - \frac{V^2}{9.81 \times 5.0}}$$

Now
$$V = \frac{Q}{\text{Area}} = \frac{50}{b \times h} = \frac{50}{20 \times 5} = 0.5$$

$$\frac{dh}{dx} = \frac{0.00025 - 0.0000208}{1 - \frac{0.5 \times 0.5}{9.81 \times 5.0}} = \frac{0.0002292}{0.9949} = \mathbf{0.00023. \text{ Ans.}}$$

16.9.2 Back Water Curve and Afflux. Consider the flow over a dam as shown in Fig. 16.30. On the upstream side of the dam, the depth of water will be rising. If there had not been any obstruction (such as dam) in the path of flow of water in the channel, the depth of water would have been constant as shown by dotted line parallel to the bed of the channel in Fig. 16.30. Due to obstruction, the water level rises and it has maximum depth from the bed at some section.

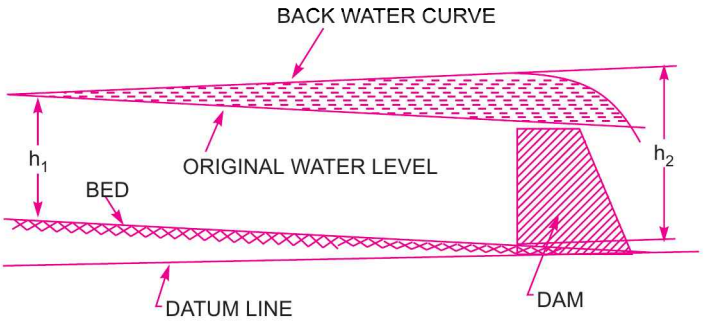


Fig. 16.30 *Back water curve and afflux.*

Let h_1 = depth of water at the point, where the water starts rising up, and
h_2 = maximum height of rising water from bed.

Then $(h_2 - h_1)$ = afflux. Thus *afflux* is defined as the maximum increase in water level due to obstruction in the path of flow of water. The profile of the rising water on the upstream side of the

* Please refer to Art. 16.9.1 point number 6.

dam is called *back water curve*. The distance along the bed of the channel between the section where water starts rising to the section where water is having maximum height is known as *length of back water curve*.

16.9.3 Expression for the Length of Back Water Curve. Consider the flow of water through a channel in which depth of water is rising as shown in Fig. 16.31. Let the two sections 1-1 and 2-2 are at such a distance that the distance between them represents the length of back water curve.

Let h_1 = depth of flow at section 1-1,

V_1 = velocity of flow at section 1-1,

h_2 = depth of flow at section 2-2,

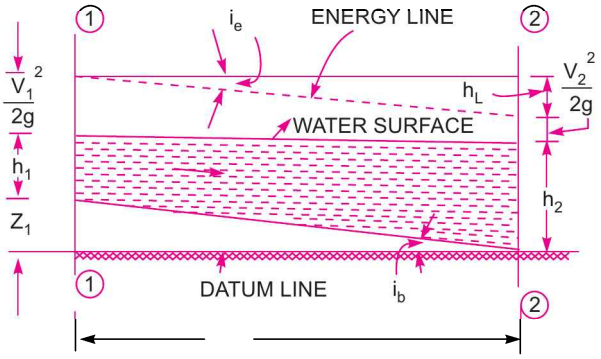


Fig. 16.31 *Length of back water curve.*

V_2 = velocity of flow at section 2-2,

i_b = bed slope,

i_e = energy line slope, and

L = length of back water curve.

Applying Bernoulli's equation at sections 1-1 and 2-2,

$$Z_1 + h_1 + \frac{V_1^2}{2g} = Z_2 + h_2 + \frac{V_2^2}{2g} + h_L \qquad ...(i)$$

where h_L = Loss of energy due to friction = $i_e \times L$

Also taking datum line passing through the bed of the channel at section 2-2. Then $Z_2 = 0$

$\therefore$ Equation (i) becomes as $Z_1 + h_1 + \frac{V_1^2}{2g} = h_2 + \frac{V_2^2}{2g} + i_e \times L$

From Fig. 16.31, $Z_1 = i_b \times L$

$\therefore$ $$i_b \times L + h_1 + \frac{V_1^2}{2g} = h_2 + \frac{V_2^2}{2g} + i_e \times L$$

or $$i_b \times L - i_e \times L = \left(h_2 + \frac{V_2^2}{2g}\right) - \left(h_1 + \frac{V_1^2}{2g}\right)$$

or $$L\,(i_b - i_e) = E_2 - E_1\,, \quad \text{where } E_2 = h_2 + \frac{V_2^2}{2g},\ E_1 = h_1 + \frac{V_1^2}{2g}$$

$$\therefore \qquad L = \frac{E_2 - E_1}{i_b - i_e}. \qquad \ldots(16.34)$$

Equation (16.34) is used to calculate the length of back water curve. The value of i_e (slope of energy line) is calculated either by Manning's formula or by Chezy's formula. The mean values of velocity, depth of flow, hydraulic mean depth etc., are used between sections 1-1 and 2-2 for calculating the value of i_e.

Problem 16.45 *Determine the length of the back water curve caused by an afflux of 2.0 m in a rectangular channel of width 40 m and depth 2.5 m. The slope of the bed is given as 1 in 11000. Take Manning's N = 0.03.*

Solution. Given :

Width of channel, $b = 40$ m

Afflux, $(h_2 - h_1) = 2.0$ m

Depth of channel, $h_1 = 2.5$ m

$\therefore \quad h_2 = 2.0 + 2.5 = 4.5$ m

Bed slope, $i_b = \dfrac{1}{11000} = 0.0000909$

Manning's, $N = 0.03$

Area of flow at section 1, $A_1 = b \times h_1 = 40 \times 2.5 = 100\ \text{m}^2$

Wetted perimeter, $P_1 = b + 2h_1 = 40 + 2 \times 2.5 = 45$ m

$\therefore$ Hydraulic mean depth, $m_1 = \dfrac{A_1}{P_1} = \dfrac{100}{45} = 2.22$ m

Using Manning's formula, $V = \dfrac{1}{N} . m^{2/3}\, i_b^{1/2}$

$\therefore$ Velocity at section 1, $V_1 = \dfrac{1}{N} m_1^{2/3}\, i_b^{1/2} = \dfrac{1}{0.03} \times 2.22^{2/3} \times 0.0000909^{1/2}$

$$= \frac{1}{0.03} \times 1.7 \times 0.009534 = 0.54 \text{ m/s}$$

Specific energy at section 1, $E_1 = \dfrac{V_1^2}{2g} + h_1 = \dfrac{0.54^2}{2 \times 9.81} + 2.5 = 2.5148$ m

From continuity, velocity at section 2 is given as

$$V_1 A_1 = V_2 \times A_2$$

$$\therefore \qquad V_2 = \frac{V_1 \times A_1}{A_2} = \frac{0.54 \times 100}{b \times h_2} = \frac{0.54 \times 100}{40 \times 4.5} = 0.3 \text{ m/s}$$

where area $A_2 = b \times h_2 = 40 \times 4.5 = 180\ \text{m}^2$

Wetted perimeter at section 2, $P_2 = b + 2h_2 = 40 + 2 \times 4.5 = 49$ m

$$\therefore \quad m_2 = \frac{A_2}{P_2} = \frac{180}{49} = 3.673 \text{ m}$$

Specific energy at section 2, $E_2 = h_2 + \frac{V_2^2}{2g} = 4.5 + \frac{0.3^2}{2 \times 9.81} = 4.504$ m

To find average velocity (V_{av}), first find average depth (h_{av}) as

$$h_{av} = \frac{h_1 + h_2}{2} = \frac{2.5 + 4.5}{2} = 3.5 \text{ m}$$

$$\therefore \quad V_{av} = \frac{V_1 A_1}{A_{av}} = \frac{V_1 \times b \times h_1}{b \times h_{av}} = \frac{V_1 \times h_1}{h_{av}} = \frac{0.54 \times 2.5}{3.5} = 0.3857 \text{ m/s}$$

Also $$m_{av} = \frac{m_1 + m_2}{2} = \frac{2.22 + 3.673}{2} = 2.9465$$

To find the value of i_e, use Manning's formula as

$$V_{av} = \frac{1}{N} m_{av}^{2/3} \times i_e^{1/2}$$

or $$0.3857 = \frac{1}{0.03} \times 2.9465^{2/3} \times i_e^{1/2} = 68.534\, i_e^{1/2}$$

or $$i_e = \left(\frac{0.3857}{68.534}\right)^2 = 0.00003167$$

The length of back water curve (L) is obtained from equation (16.34)

$$L = \frac{E_2 - E_1}{i_b - i_e} = \frac{4.504 - 2.5148}{0.0000909 - 0.00003167}$$

$$= \frac{1.9892}{0.00005923} = \textbf{33584.3 m. Ans.}$$

HIGHLIGHTS

1. If the depth of flow, velocity of flow, slope of the bed of channel and cross-section remain constant, the flow is called uniform, otherwise it is called non-uniform flow.
2. Non-uniform flow is also called varied flow. If the depth of flow changes abruptly over a small length of channel, the flow is called rapidly varied flow. If the depth of flow changes gradually over a long length of the channel, the flow is said to be gradually varied flow.
3. If Reynold number for open channel flow is less than 500, the flow is said to be laminar and if R_e is more than 2000, the flow is said to be turbulent.
4. If Froude number is less than 1.0, the flow is sub-critical or streaming. If $F_e = 1.0$, the flow is critical. If $F_e > 1.0$, the flow is super-critical or shooting.
5. Velocity of Chezy's formula is given as $V = C\sqrt{mi}$

where C = Chezy's constant, m = Hydraulic mean depth = $\frac{\text{Area}}{\text{Wetted perimeter}}$,

i = Slope of the bed.

6. The value of Chezy's constant, C is given by empirical formulae as :

(*i*) $C = \dfrac{157.6}{1.81 + \dfrac{K}{\sqrt{m}}}$ ... Bazin Formula

where K = Bazin's constant, m = Hydraulic mean depth

(*ii*) $C = \dfrac{23 + \dfrac{0.00155}{i} + \dfrac{1}{N}}{1 + \left(23 + \dfrac{0.00155}{i}\right)\dfrac{N}{\sqrt{m}}}$ ... Kutter's Formula

where N = Kutter's constant.

(*iii*) $C = \dfrac{1}{N} m^{1/6}$ ... Manning's Formula

where N = Manning's constant = Kutter's constant.

7. Most economical section is one that gives maximum discharge for a given values of cross-section area, slope of the bed and co-efficient of resistance.

8. Conditions for maximum discharge through :

(*a*) Rectangular section,

(*i*) $b = 2d$ (*ii*) $m = \dfrac{d}{2}$

(*b*) Trapezoidal section,

(*i*) Half of top width = Sloping side or $\dfrac{b + 2nd}{2} = d\sqrt{n^2 + 1}$

(*ii*) $m = \dfrac{d}{2}$

(*iii*) A semi-circle drawn from the mid-point of the top width with radius equal to depth of flow will touch the three sides of the channel.

9. Best side slope for most economical trapezoidal section is,

$$\theta = 60° \text{ or } n = \frac{1}{\sqrt{3}} = \frac{1}{\tan\theta}.$$

10. For circular sections, area cannot be maintained constant and hence there are two different conditions, one is for maximum velocity and other for maximum discharge.

11. Condition for maximum velocity through a circular channel is,

Depth of flow, d = 0.81 diameter of circular channel

Hydraulic mean depth, m = 0.305 diameter of circular channel

12. Condition for maximum discharge through a circular channel is,

Depth, d = 0.95 diameter of circular channel.

13. For a circular channel,

Wetted perimeter, $P = 2R\theta$

Area of flow, $A = R^2\left(\theta - \frac{\sin 2\theta}{2}\right)$

where R = Radius of circular channel,
θ = Half the angle subtended by the water surface at the centre.

14. Total energy of a flowing liquid per unit weight, Total energy = $z + h + V^2 / 2g$.

15. Specific energy of a flowing liquid per unit weight,

$$E = h + \frac{V^2}{2g}, \qquad \text{where } h = \text{Depth of flow, } V = \text{Velocity of flow.}$$

16. The depth of flow at which specific energy is minimum is called critical depth, which is given by

$$h_c = \left(\frac{q^2}{g}\right)^{1/3}, \quad \text{where } q = \text{discharge per unit width} = \frac{\text{Total discharge}}{b}.$$

17. The velocity of flow at critical depth is known as critical velocity, which is given as $V_c = \sqrt{g \times h_c}$.

18. Minimum specific energy is related with critical depth by the relation, $E_{\min} = \frac{3}{2} h_c$.

19. The flow corresponding to critical depth (or when Froude number is equal to 1.0) is known as critical flow.

20. If the depth of flow in a channel is greater than the critical depth (or Froude number is less than 1.0), the flow is said sub-critical or streaming flow.

21. If the depth of flow in a channel is less than the critical depth (or Froude number is more than 1.0), the flow is known as super-critical or shooting flow.

22. The condition for maximum discharge for a given value of specific energy is that the depth of flow should be critical.

23. The rise of water-level which takes place due to the transformation of the shooting to the streaming flow is known as hydraulic jump. The depth of flow after the jump is given by

$$d_2 = -\frac{d_1}{2} + \sqrt{\frac{d_1^2}{4} + \frac{2q^2}{gd_1}} \qquad \text{... (In terms of } q\text{)}$$

$$= -\frac{d_1}{2} + \sqrt{\frac{d_1^2}{4} + \frac{2V_1^2\, d_1}{g}} \qquad \text{... (In terms of } V_1\text{)}$$

$$= \frac{d_1}{2}\left(\sqrt{1 + 8(F_e)_1^2} - 1\right) \qquad \text{... (In terms of } F_{e_1}\text{)}$$

Depth of hydraulic jump, $= d_2 - d_1$

where d_1 = depth of flow before hydraulic jump,
V_1 = velocity of flow before hydraulic jump.

24. Energy lost due to hydraulic jump per kg of liquid

$$h_L = (E_1 - E_2) = \left(d_1 + \frac{V_1^2}{2g}\right) - \left(d_2 + \frac{V_2^2}{2g}\right) = \frac{(d_2 - d_1)^3}{4d_1 d_2}.$$

25. Equation of gradually varied flow, $\frac{dh}{dx} = \frac{i_b - i_e}{1 - \frac{V^2}{gh}}$ ... (In terms of V)

$$= \frac{i_b - i_e}{\left(1 - F_e^2\right)} \quad \text{... (In term of } F_e\text{)}$$

where $\frac{dh}{dx}$ = slope of free water surface, i_b = bed slope,

i_e = slope of the energy line, h = depth of flow, and V = velocity of flow.

26. Afflux is the increase in water level due to some obstruction across the flowing liquid, while back water curve is the profile of the rising water on the upstream side of the obstruction.

27. Length of back water curve is given by, $L = \frac{E_2 - E_1}{i_b - i_e}$

where E_1 = Specific energy at the section, where water starts rising = $h_1 + \frac{V_1^2}{2g}$

and E_2 = Specific energy at the end of the water curve = $h_2 + \frac{V_2^2}{2g}$.

EXERCISE

(A) THEORETICAL PROBLEMS

1. What do you understand by 'Flow in open channel' ?

2. Differentiate between : (*i*) Uniform flow and non-uniform flow, (*ii*) Steady and unsteady flow, (*iii*) Laminar and turbulent flow and (*iv*) Critical, sub-critical and super-critical flow in a open channel.

3. Explain the terms : (*i*) Rapidly varied flow and (*ii*) Gradually varied flow.

4. Derive an expression for the discharge through a channel by Chezy's formula.

5. Explain the terms : (*i*) Slope of the bed, (*ii*) Hydraulic mean depth and (*iii*) Wetted perimeter.

6. (*a*) What are the empirical formulae for determining the value of Chezy's constant ?

(*b*) What is the relation between Manning's constant and Chezy's constant.

7. State the following formulae for the values of C :

(*i*) Bazin's formulae, (*ii*) Kutter's formula, and (*iii*) Manning's formula.

8. (*a*) Define the term most economical section of a channel. What are the conditions for the rectangular channel of the best section ?

(*b*) What is meant by an economical section of a channel ?

9. Prove that for the trapezoidal channel of most economical section :

(*i*) Half of top width = Length of one of the sloping sides

(*ii*) Hydraulic mean depth = $\frac{1}{2}$ depth of flow.

10. (*a*) Derive the condition for the best side slope of the most economical trapezoidal channel.

(*b*) Find the side slope in a trapezoidal section of maximum efficiency which will carry the same flow as a half square section of the same area.

11. Prove that for a channel of circular section, the depth of flow,

$d = 0.81\ D$ for maximum velocity, and

$= 0.95\ D$ for maximum discharge,

where D = Diameter of circular channel, d = depth of flow.

12. Explain the terms : Specific energy of a flowing liquid, minimum specific energy, critical depth, critical velocity and alternate depths as applied to non-uniform flow.

13. What is specific energy curve ? Draw specific energy curve, and then derive expressions for critical depth and critical velocity.

14. (*a*) Derive an expression for critical depth and critical velocity.

(*b*) Define critical depth in an open channel in as many ways as you can.

15. Derive the condition for maximum discharge for a given value of specific energy.

16. Explain the term hydraulic jump. Derive an expression for the depth of hydraulic jump in terms of the upstream Froude number.

17. Derive an expression for the variation of depth along the length of the bed of the channel for gradually varied flow in an open channel. State clearly all the assumptions made.

18. Find an expression for loss of energy head for a hydraulic jump.

19. Define the terms : (*i*) Afflux and (*ii*) Back water curve. Prove that the length of the back water curve is given by,

$$L = \frac{(E_2 - E_1)}{i_b - i_e}$$

where L = Length of back water curve,

E_2 = Specific energy at the end of back water curve,

E_1 = Specific energy at the section where water starts rising,

i_b = Slope of bed, and i_e = Slope of the energy line.

20. Find, in terms of specific energy E, an expression for the critical depth in a trapezoidal channel with bottom width B and side slopes of 1 vertical to n horizontal.

21. Show that in a rectangular channel :

(*i*) Critical depth is two-third of specific energy, and

(*ii*) Froude number at critical depth is unity.

22. Obtain the condition for a trapezoidal channel with side slopes 2 H : 1 V to be most efficient for a given area A let B be its bed width.

23. By applying the momentum equation to open channel flow, show that the consequent depths and flow rate are related by $2q^2/g = y_1y_2\ (y_1 + y_2)$.

State the assumptions made in the derivation.

24. Derive the differential equation for steady gradually varied flow in open channels and list all assumptions.

$$\frac{dh}{dx} = \frac{(i_b - i_e)}{\left(1 - F_e^2\right)}.$$

25. What is the essential difference between gradually varied flow and rapidly varied flow ? Illustrate with neatly drawn sketches.

26. (*a*) Prove that the loss of energy head in a hydraulic jump is equal to $(d_2 - d_1)^3/4d_1d_2$, where d_1 and d_2 are the conjugate depths.

(*b*) Obtain the relationship between the Froude Numbers of flow before and after the hydraulic jump in a horizontal rectangular channel.

(B) NUMERICAL PROBLEMS

1. Find the velocity of flow and rate of flow of water through a rectangular channel of 5 m wide and 2 m deep, when it is running full. The channel is having bed slope of 1 in 3000. Take Chezy's constant $C = 50$. [**Ans.** 0.962 m/s, 9.62 m^3/s]
2. A flow of water of 150 litres per second flows down in a rectangular flume of width 70 cm and having adjustable bottom slope. If Chezy's constant C is 60, find the bottom slope necessary for uniform flow with a depth of flow of 40 cm. Also find the conveyance K of the flume. [**Ans.** 1 in 2341.5, 7.258]
3. Find the discharge through a trapezoidal channel of width 6 m and side slope of 1 horizontal to 3 vertical. The depth of flow of water is 3 m and Chezy's constant, $C = 60$. The slope of the bed of the channel is given 1 in 5000. [**Ans.** 23.26 m^3/s]
4. Find the rate of flow of water through a V-shaped channel having total angle between the sides as 60°. Take the value of $C = 50$ and slope of the bed 1 in 1500. The depth of flow is 6 m. [**Ans.** 32.864 m^3/s]
5. Find the discharge through a rectangular channel 3 m wide, having depth of water 2 m and bed slope as 1 in 1500. Take the value of $K = 2.36$ in Bazin's formula. [**Ans.** 5.184 m^3/s]
6. Find the discharge through the rectangular channel given in the above question, taking the value of $N = 0.012$ in Manning's formula. [**Ans.** 11.64 m^3/s]
7. Find the bed slope of trapezoidal channel of bed width 3 m, depth of water 2.5 m and side slope of 2 horizontal to 3 vertical, when the discharge through the channel is 10 m^3/s. Taking the value of $N = 0.03$ in Manning's formula

$$C = \frac{1}{N} m^{1/6}.$$

[**Ans.** 1 in 1803]

8. Find the diameter of a circular sewer pipe which is laid at a slope of 1 in 10000 and carries a discharge of 1000 litres/s when flowing half full. Take the value of Manning's $N = 0.02$. [**Ans.** 2.6 m]
9. A rectangular channel carries water at the rate of 500 litres/s when bed slope is 1 in 3000. Find the most economical dimensions of the channel if $C = 60$. [**Ans.** $b = 1.272$ m, $d = 0.636$ m]
10. A trapezoidal channel has side slopes of 1 horizontal to 2 vertical and the slope of the bed is 1 in 2000. The area of the section is 42 m^2. Find the dimensions of the section if it is most economical. Determine the discharge of the most economical section if $C = 60$. [**Ans.** $d = 4.918$ m, $b = 6.08$ m, $Q = 88.36$ m^3/s]
11. A trapezoidal channel with side slopes of 1 to 1 has to be designed to convey 9 m^3/s at a velocity of 1.5 m/s so that the amount of concrete lining for the bed and sides is the minimum. Calculate the area of lining required for one metre length of canal. [**Ans.** 6.62 m^2]
12. A power canal of trapezoidal section has to be excavated through hard clay at the least cost. Determine the dimensions of the channel given, discharge equal to 15 m^3/s, bed slope 1 : 2000 and Manning's, $N = 0.020$. [**Ans.** $b = 2.956$ m, $d = 2.56$ m]
13. Find the discharge through a circular pipe of diameter 4.0 m, if the depth of water in the pipe is 1.33 m and pipe is laid at a slope of 1 in 1500. Take the value of Chezy's constant = 60. [**Ans.** 4.89 m^3/s]
14. Water is flowing through a circular channel at the rate of 500 litres/s. The depth of water in the channel is 0.7 times the diameter and the slope of the bed of the channel is 1 in 8000. Find the diameters of the circular channel if the value of Manning's, $N = 0.015$. [**Ans.** 1.425 m]
15. The rate of flow of water through a circular channel of diameter 0.8 m is 200 litres/s. Find the slope of the bed of the channel for maximum velocity. Take $C = 50$. [**Ans.** 1/2787]
16. Determine the maximum discharge of water through a circular channel of diameter 2.0 m when the bed slope of the channel is 1 in 1500. Take $C = 50$. [**Ans.** 3.02 m^3/s]
17. The discharge of water through a rectangular channel of width 6 m, is 18 m^3/s when depth of flow of water is 2 m. Calculate : (*i*) specific energy of the flowing water, (*ii*) critical depth and critical velocity and (*iii*) value of minimum specific energy. [**Ans.** (*i*) 2.115 m, (*ii*) 0.971 m, 3.087 m/s, (*iii*) 1.457 m]

18. The specific energy for a 6 m wide rectangular channel is to be 5 kg-m/kg. If the rate of flow of water through the channel is 24 m^2/s, determine the alternate depths of flow. [**Ans.** 4.831 m, 0.169 m]

19. The depth of flow of water, at a certain section of a rectangular channel of 5 m wide is 0.6 m. The discharge through the channel is 15 m^3/s. If a hydraulic jump takes place on the downstream side, find the depth of flow after the jump. [**Ans.** 1.474 m]

20. For the Question 19, find the loss of energy per kg of water due to hydraulic jump. [**Ans.** 0.188 m]

21. A sluice gate discharges water into a horizontal rectangular channel with a velocity of 8 m/s and depth of flow is 0.5 m. The width of the channel is 6 m. Determine whether a hydraulic jump will occur, and if so, find its height and loss of energy per kg of water. Also determine the horse power lost in the hydraulic jump. [**Ans.** Yes, 1.816 m, 1.293 m, 413.76 h.p.]

22. Find the rate of change of depth of water in a rectangular channel of 12 m wide and 2 m deep, when the water is flowing with a velocity of 1.5 m/s. The flow of water through the channel of bed slope 1 in 300, is regulated in such a way that energy line is having a slope of 1 in 8000. [**Ans.** 0.000235]

23. Find the slope of the free water surface in a rectangular channel of width 15 m, having depth of flow 4 m. The discharge through the channel is 40 m^3/s. The bed of the channel is having a slope of 1 in 4000. Take the value of Chezy's constant, $C = 50$. [**Ans.** 0.000184]

24. Determine the length of the back water curve caused by an afflux of 1.5 m in a rectangular channel of width 50 m and depth 2.0 m. The slope of the bed is given as 1 in 2000. Take Manning's, $N = 0.03$. [**Ans.** 4566 m]

25. A trapezoidal channel with bottom slope 0.000169, bottom width 10 m and side slopes 1 : 1 carries 20 m^3/s when Manning's constant = 0.015. Determine the normal depth.

[**Hint.** $i = 0.000169$, $b = 10$ m, $n = 1$, $N = 0.015$, $Q = 20\ m^3/s$

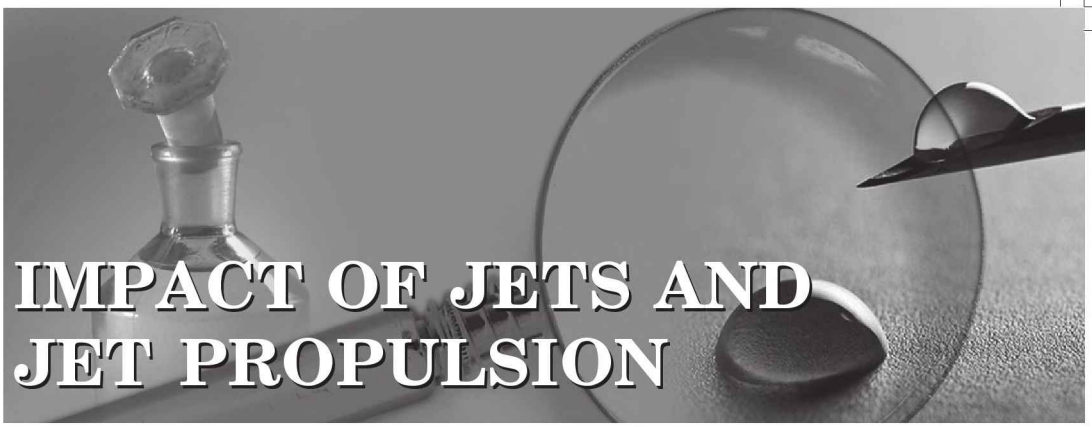
Use $$Q = \frac{1}{N} m^{2/3} \times i^{1/2} \times A, \text{ where } A = (b + nd) \times d = (10 + d)\, d$$

$$P = b + 2d\sqrt{1+n^2} = 10 + 2 \times \sqrt{2} \times d, m = \frac{(10+d)\,d}{\left(10 + 2\sqrt{2} \times d\right)}$$

$\therefore$ $$20 = \frac{1}{0.015} \times \left[\frac{(10+d)\,d}{\left(10 + 2\sqrt{2}\, d\right)}\right]^{2/3} \times 0.000169^{1/2} \times (10+d)\,d$$

or $$\frac{\left[(10+d)\,d\right]^{5/3}}{\left[10 + 2\sqrt{2}\, d\right]^{2/3}} = 23.$$ Find 'd' by hit and trial. [$d = 1.65$ m]]

17 CHAPTER



IMPACT OF JETS AND JET PROPULSION

▶ 17.1 INTRODUCTION

The liquid comes out in the form of a jet from the outlet of a nozzle, which is fitted to a pipe through which the liquid is flowing under pressure. If some plate, which may be fixed or moving, is placed in the path of the jet, a force is exerted by the jet on the plate. This force is obtained from Newton's second law of motion or from impulse-momentum equation. Thus impact of jet means the force exerted by the jet on a plate which may be stationary or moving. In this chapter, the following cases of the impact of jet *i.e.,* the force exerted by the jet on a plate, will be considered :

1. Force exerted by the jet on a stationary plate when
   (*a*) Plate is vertical to the jet, (*b*) Plate is inclined to the jet, and (*c*) Plate is curved.
2. Force exerted by the jet on a moving plate, when
   (*a*) Plate is vertical to the jet, (*b*) Plate is inclined to the jet, and (*c*) Plate is curved.

▶ 17.2 FORCE EXERTED BY THE JET ON A STATIONARY VERTICAL PLATE

Consider a jet of water coming out from the nozzle, strikes a flat vertical plate as shown in Fig. 17.1

Let V = velocity of the jet, d = diameter of the jet,

a = area of cross-section of the jet = $\frac{\pi}{4} d^2$.

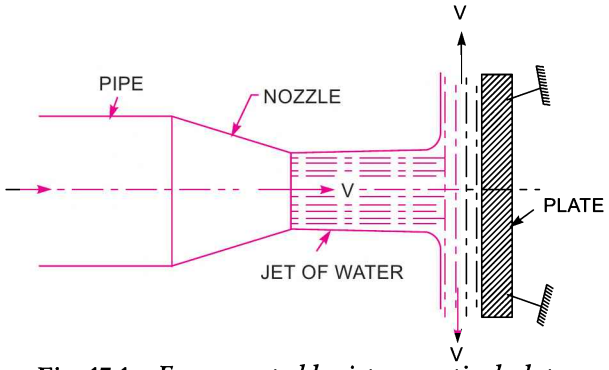


Fig. 17.1 *Force exerted by jet on vertical plate.*

The jet after striking the plate, will move along the plate. But the plate is at right angles to the jet. Hence the jet after striking, will get deflected through 90°. Hence the component of the velocity of jet, in the direction of jet, after striking will be zero.

The force exerted by the jet on the plate in the direction of jet,

$$F_x = \text{Rate of change of momentum in the direction of force}$$

$$= \frac{\text{Initial momentum} - \text{Final momentum}}{\text{Time}}$$

$$= \frac{(\text{Mass} \times \text{Initial velocity} - \text{Mass} \times \text{Final velocity})}{\text{Time}}$$

$$= \frac{\text{Mass}}{\text{Time}} [\text{Initial velocity} - \text{Final velocity}]$$

$$= (\text{Mass/sec}) \times (\text{velocity of jet before striking} - \text{velocity of jet after striking})$$

$$= \rho a V[V - 0] \qquad (\because \text{mass/sec} = \rho \times a\ V)$$

$$= \rho a V^2 \qquad \ldots(17.1)$$

For deriving above equation, we have taken initial velocity minus final velocity and not final velocity minus initial velocity. If the force exerted on the jet is to be calculated then final minus initial velocity is taken. But if the force exerted by the jet on the plate is to be calculated, then initial velocity minus final velocity is taken.

Note. In equation (17.1), if the value of density (ρ) is taken in S.I. units (*i.e.*, kg/m^3), the force (F_x) will be in Newton (N). The value of ρ for water in S.I. units is equal to 1000 kg/m^3.

17.2.1 Force Exerted by a Jet on Stationary Inclined Flat Plate. Let a jet of water, coming out from the nozzle, strikes an inclined flat plate as shown in Fig. 17.2.

Let V = Velocity of jet in the direction of x,

θ = Angle between the jet and plate,

a = Area of cross-section of the jet.

Then mass of water per sec striking the plate = $\rho \times aV$.

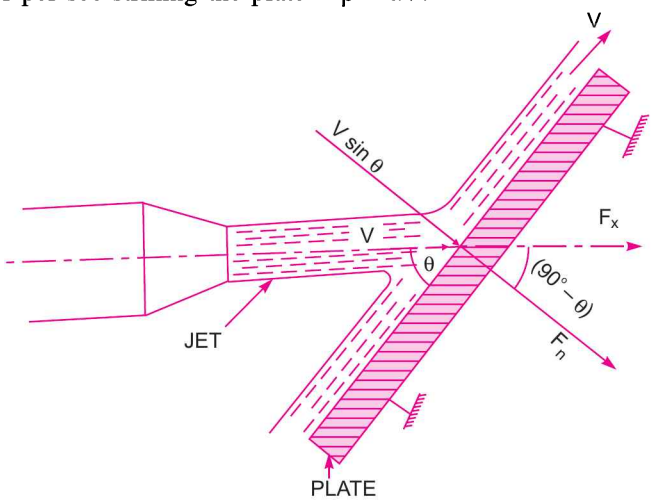


Fig. 17.2 *Jet striking stationary inclined plate.*

If the plate is smooth and if it is assumed that there is no loss of energy due to impact of the jet, then jet will move over the plate after striking with a velocity equal to initial velocity *i.e.*, with a velocity V. Let us find the force exerted by the jet on the plate in the direction normal to the plate. Let this force is represented by F_n

Then F_n = mass of jet striking per second
× [Initial velocity of jet before striking in the direction of n
– Final velocity of jet after striking in the direction of n]
$= \rho aV\ [V \sin \theta - 0\ ] = \rho aV^2 \sin \theta$ ...(17.2)

This force can be resolved into two components, one in the direction of the jet and other perpendicular to the direction of flow. Then we have,

F_x = component of F_n in the direction of flow
$= F_n \cos (90° - \theta) = F_n \sin \theta = \rho A\ V^2 \sin \theta \times \sin \theta$ $(\therefore\ F_n = \rho aV^2 \sin \theta)$
$= \rho AV^2 \sin^2 \theta$ ...(17.3)

And, F_y = component of F_n, perpendicular to flow
$= F_n \sin (90° - \theta) = F_n \cos \theta = \rho AV^2 \sin \theta \cos \theta.$ ...(17.4)

17.2.2 Force Exerted by a Jet on Stationary Curved Plate

(A) Jet strikes the curved plate at the centre. Let a jet of water strikes a fixed curved plate at the centre as shown in Fig. 17.3. The jet after striking the plate, comes out with the same velocity if the plate is smooth and there is no loss of energy due to impact of the jet, in the tangential direction of the curved plate. The velocity at outlet of the plate can be resolved into two components, one in the direction of jet and other perpendicular to the direction of the jet.

Component of velocity in the direction of jet = $- V \cos \theta$.

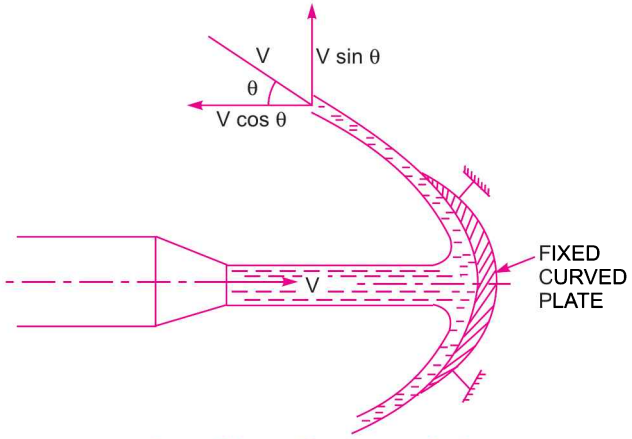


Fig. 17.3 *Jet striking a fixed curved plate at centre.*

(–ve sign is taken as the velocity at outlet is in the opposite direction of the jet of water coming out from nozzle).

Component of velocity perpendicular to the jet = $V \sin \theta$

Force exerted by the jet in the direction of jet,

$$F_x = \text{Mass per sec} \times [V_{1x} - V_{2x}]$$

where V_{1x} = Initial velocity in the direction of jet = V

V_{2x} = Final velocity in the direction of jet = $- V \cos \theta$

$\therefore$ $F_x = \rho aV[V - (-V\cos\theta)] = \rho aV[V + V\cos\theta]$

$= \rho aV^2[1 + \cos\theta]$ ...(17.5)

Similarly, F_y = Mass per sec $\times [V_{1y} - V_{2y}]$

where V_{1y} = Initial velocity in the direction of $y = 0$

V_{2y} = Final velocity in the direction of $y = V\sin\theta$

$\therefore$ $F_y = \rho aV[0 - V\sin\theta] = -\rho aV^2\sin\theta$ ...(17.6)

–ve sign means that force is acting in the downward direction. In this case the angle of deflection of the jet $= (180° - \theta)$ ...[17.6 (*A*)]

(B) Jet strikes the curved plate at one end tangentially when the plate is symmetrical. Let the jet strikes the curved fixed plate at one end tangentially as shown in Fig. 17.4. Let the curved plate is symmetrical about x-axis. Then the angle made by the tangents at the two ends of the plate will be same.

Let V = Velocity of jet of water,

θ = Angle made by jet with x-axis at inlet tip of the curved plate.

If the plate is smooth and loss of energy due to impact is zero, then the velocity of water at the outlet tip of the curved plate will be equal to V. The forces exerted by the jet of water in the directions of x and y are

F_x = (mass/sec) $\times [V_{1x} - V_{2x}]$

$= \rho aV[V\cos\theta - (-V\cos\theta)]$

$= \rho aV[V\cos\theta + V\cos\theta]$

$= 2\rho aV^2\cos\theta$ ...(17.7)

$F_y = \rho aV[V_{1y} - V_{2y}]$

$= \rho aV[V\sin\theta - V\sin\theta] = 0$

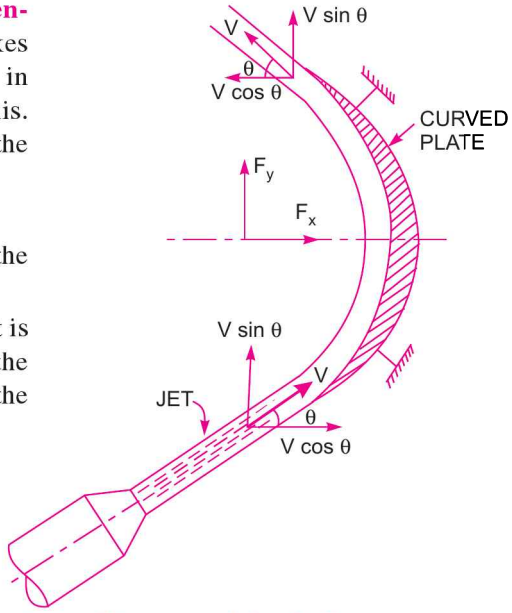


Fig. 17.4 *Jet striking curved fixed plate at one end.*

(C) Jet strikes the curved plate at one end tangentially when the plate is unsymmetrical. When the curved plate is unsymmetrical about x-axis, then angle made by the tangents drawn at the inlet and outlet tips of the plate with x-axis will be different.

Let θ = angle made by tangent at inlet tip with x-axis,

ϕ = angle made by tangent at outlet tip with x-axis.

The two components of the velocity at inlet are

$V_{1x} = V\cos\theta$ and $V_{1y} = V\sin\theta$

The two components of the velocity at outlet are

$V_{2x} = -V\cos\phi$ and $V_{2y} = V\sin\phi$

$\therefore$ The forces exerted by the jet of water in the directions of x and y are

$F_x = \rho aV[V_{1x} - V_{2x}] = \rho aV[V\cos\theta - (-V\cos\phi)]$

$= \rho aV[V\cos\theta + V\cos\phi] = \rho aV^2[\cos\theta + \cos\phi]$ ...(17.8)

$F_y = \rho aV[V_{1y} - V_{2y}] = \rho aV[V\sin\theta - V\sin\phi]$

$= \rho aV^2[\sin\theta - \sin\phi]$. ...(17.9)

Problem 17.1 *Find the force exerted by a jet of water of diameter 75 mm on a stationary flat plate, when the jet strikes the plate normally with velocity of 20 m/s.*

Solution. Given :

Diameter of jet, $d = 75 \text{ mm} = 0.075 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} (.075)^2 = .004417 \text{ m}^2$

Velocity of jet, $V = 20 \text{ m/s}$.

The force exerted by the jet of water on a stationary vertical plate is given by equation (17.1) as

$$F = \rho a V^2 \quad \text{where } \rho = 1000 \text{ kg/m}^3$$

$\therefore$ $F = 1000 \times .004417 \times 20^2 \text{ N} =$ **1766.8 N. Ans.**

Problem 17.2 *Water is flowing through a pipe at the end of which a nozzle is fitted. The diameter of the nozzle is 100 mm and the head of water at the centre nozzle is 100 m. Find the force exerted by the jet of water on a fixed vertical plate. The co-efficient of velocity is given as 0.95.*

Solution. Given :

Diameter of nozzle, $d = 100 \text{ mm} = 0.1 \text{ m}$

Head of water, $H = 100 \text{ m}$

Co-efficient of velocity, $C_v = 0.95$

Area of nozzle, $a = \frac{\pi}{4} (.1)^2 = .007854 \text{ m}^2$

Theoretical velocity of jet of water is given as

$$V_{th} = \sqrt{2gH} = \sqrt{2 \times 9.81 \times 100} = 44.294 \text{ m/s}$$

But $$C_v = \frac{\text{Actual velocity}}{\text{Theoretical velocity}}$$

$\therefore$ Actual velocity of jet of water, $V = C_v \times V_{th} = 0.95 \times 44.294 = 42.08$ m/s.

Force on a fixed vertical plate is given by equation (17.1) as

$$F = \rho a V^2 = 1000 \times .007854 \times 42.08^2 \quad (\because \text{ In S.I. units } \rho \text{ for water} = 1000 \text{ kg/m}^3)$$

$$= 13907.2 \text{ N} = \textbf{13.9 kN. Ans.}$$

Problem 17.3 *A jet of water of diameter 75 mm moving with a velocity of 25 m/s strikes a fixed plate in such a way that the angle between the jet and plate is 60°. Find the force exerted by the jet on the plate (i) in the direction normal to the plate and (ii) in the direction of the jet.*

Solution. Given :

Diameter of jet, $d = 75 \text{ mm} = 0.075 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} (.075)^2 = 0.004417 \text{ m}^2$

Velocity of jet, $V = 25 \text{ m/s}$.

Angle between jet and plate $\theta = 60°$

(*i*) The force exerted by the jet of water in the direction normal to the plate is given by equation (17.2) as

$$F_n = \rho a V^2 \sin \theta$$

$$= 1000 \times .004417 \times 25^2 \times \sin 60° = \textbf{2390.7 N. Ans.}$$

(*ii*) The force in the direction of the jet is given by equation (17.3),

$$F_x = \rho a V^2 \sin^2 \theta$$

$$= 1000 \times .004417 \times 25^2 \times \sin^2 60° = \textbf{2070.4 N. Ans.}$$

Problem 17.4 *A jet of water of diameter 50 mm strikes a fixed plate in such a way that the angle between the plate and the jet is 30°. The force exerted in the direction of the jet is 1471.5 N. Determine the rate of flow of water.*

Solution. Given :

Diameter of jet, $d = 50 \text{ mm} = 0.05 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4} (.05)^2 = .001963 \text{ m}^2$

Angle, $\theta = 30°$

Force in the direction of jet, $F_x = 1471.5 \text{ N}$

Force in the direction of jet is given by equation (17.3) as $F_x = \rho a V^2 \sin^2 \theta$

As the force is given in Newton, the value of ρ should be taken equal to 1000 kg/m^3

$\therefore$ $1471.5 = 1000 \times .001963 \times V^2 \times \sin^2 30° = .05\ V^2$

$\therefore$ $V^2 = \frac{150}{.05} = 3000.0$

$V = 54.77 \text{ m/s}$

$\therefore$ Discharge, $Q = \text{Area} \times \text{Velocity}$

$= .001963 \times 54.77 = 0.1075 \text{ m}^3/\text{s} = \textbf{107.5 liters/s. Ans.}$

Problem 17.5 *A jet of water of diameter 50 mm moving with a velocity of 40 m/s, strikes a curved fixed symmetrical plate at the centre. Find the force exerted by the jet of water in the direction of the jet, if the jet is deflected through an angle of 120° at the outlet of the curved plate.*

Solution. Given :

Diameter of the jet, $d = 50 \text{ mm} = 0.05 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4} (.05)^2 = 0.001963 \text{ m}^2$

Velocity of jet, $V = 40 \text{ m/s}$

Angle of deflection $= 120°$

From equation [17.6 (*A*)], the angle of deflection $= 180° - \theta$

$\therefore$ $180° - \theta = 120°$ or $\theta = 180° - 120° = 60°$

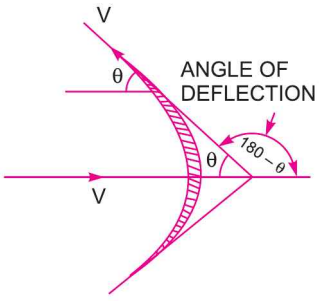


Fig. 17.5

Force exerted by the jet on the curved plate in the direction of the jet is given by equation (17.5) as

$$F_x = \rho a V^2 [1 + \cos \theta]$$

$$= 1000 \times .001963 \times 40^2 \times [1 + \cos 60°] = \textbf{4711.15 N. Ans.}$$

Problem 17.6 *A jet of water of diameter 75 mm moving with a velocity of 30 m/s, strikes a curved fixed plate tangentially at one end at an angle of 30° to the horizontal. The jet leaves the plate at an angle of 20° to the horizontal. Find the force exerted by the jet on the plate in the horizontal and vertical direction.*

Solution. Given :

Diameter of the jet, $d = 75 \text{ mm} = 0.075 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4} (.075)^2 = .004417 \text{ m}^2$

Velocity of jet, $V = 30 \text{ m/s}$

Angle made by the jet at inlet tip with horizontal, $\theta = 30°$

Angle made by the jet at outlet tip with horizontal, $\phi = 20°$

The force exerted by the jet of water in the direction of x is given by equation (17.8) and in the direction of y by equation (17.9),

$\therefore$ $F_x = \rho a V^2 [\cos \theta + \cos \phi]$

$= 1000 \times .004417 [\cos 30° + \cos 20°] \times 30^2 =$ **7178.2 N. Ans.**

and $F_y = \rho a V^2 [\sin \theta - \sin \phi]$

$= 1000 \times .004417 [\sin 30° - \sin 20°] \times 30^2 =$ **628.13 N. Ans.**

▶ 17.3 FORCE EXERTED BY A JET ON A HINGED PLATE

Consider a jet of water striking a vertical plate at the centre which is hinged at O. Due to the force exerted by the jet on the plate, the plate will swing through some angle about the hinge as shown in Fig. 17.6

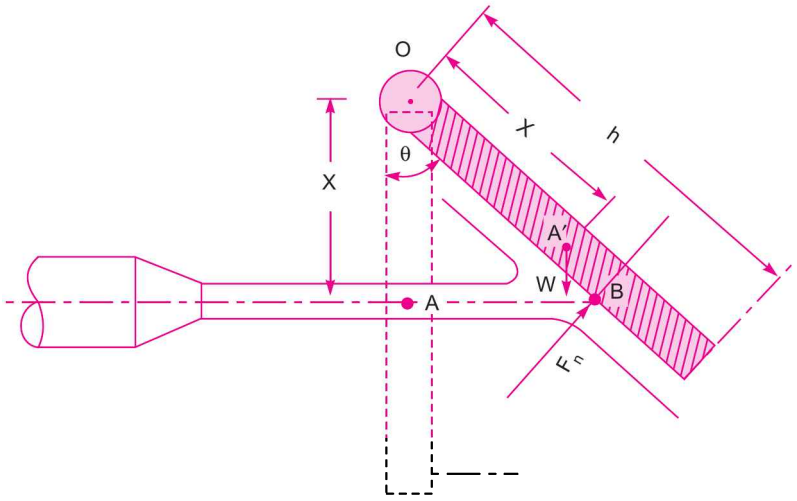


Fig. 17.6 *Force on a hinged plate.*

Let x = distance of the centre of jet from hinge O,

θ = angle of swing about hinge,

W = weight of plate acting at C.G. of the plate.

The dotted lines show the position of the plate, before the jet strikes the plate. The point A on the plate will be at A' after the jet strikes the plate. The distance $OA = OA' = x$. Let the weight of the plate is acting at A'. When the plate is in equilibrium after the jet strikes the plate, the moment of all the forces about the hinge must be zero. Two forces are acting on the plate. They are :

1. Force due to jet of water, normal to the plate,

$$F_n = \rho a V^2 \sin \theta'$$

where θ' = Angle between jet and plate = $(90° - \theta)$

2. Weight of the plate, W

Moment of force F_n about hinge = $F_n \times OB = \rho a V^2 \sin (90° - \theta) \times OB = \rho a V^2 \cos \theta \times OB$

$$= \rho a V^2 \cos \theta \times \frac{OA}{\cos \theta} = \rho a V^2 \times OA = \rho a V^2 \times x$$

Moment of weight W about hinge $= W \times OA' \sin \theta = W \times x \times \sin \theta$

For equilibrium of the plate, $\rho a V^2 \times x = W \times x \times \sin \theta$

$$\therefore \qquad \sin \theta = \frac{\rho a V^2}{W} \qquad \ldots(17.10)$$

From equation (17.10), the angle of swing of the plate about hinge can be calculated.

Problem 17.7 *A jet of water of 2.5 cm diameter, moving with a velocity of 10 m/s, strikes a hinged square plate of weight 98.1 N at the centre of the plate. The plate is of uniform thickness. Find the angle through which the plate will swing.*

Solution. Given :

Diameter of jet, $d = 2.5$ cm $= 0.025$ m

Velocity of jet, $V = 10$ m/s

Weight of plate, $W = 98.1$ N

Area of jet, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} \times (.025)^2 = .00049 \text{ m}^2$

The angle through which the plate will swing is given by equation (17.10) as

$$\sin \theta = \frac{\rho a V^2}{W} = 1000 \times \frac{.00049 \times 10^2}{98.1} \qquad (\because \quad \rho = 1000)$$

$$= .499$$

$$\therefore \qquad \theta = \mathbf{29.96°. \text{ Ans.}}$$

Problem 17.8 *A jet of water of 30 mm diameter strikes a hinged square plate at its centre with a velocity of 20 m/s. The plate is deflected through an angle of 20°. Find the weight of the plate.*

If the plate is not allowed, to swing, what will be the force required at the lower edge of the plate to keep the plate in vertical position.

Solution. Given :

Diameter of the jet, $d = 30$ mm $= 3$ cm $= 0.03$ m

$\therefore$ Area, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} (.03)^2 = .0007068 \text{ m}^2$

Velocity of jet, $V = 20$ m/s

Angle of swing, $\theta = 20°$

Using equation (17.10) for angle of swing,

$$\sin\theta = \frac{\rho a V^2}{W}$$

or $$\sin 20^\circ = 1000 \times \frac{.0007068 \times 20^2}{W} = \frac{282.72}{W}$$

$$\therefore \quad W = \frac{282.72}{\sin 20^\circ} = 826.6 \text{ N}$$

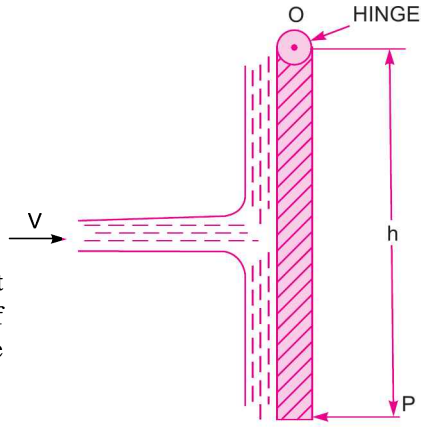


Fig. 17.7

If the plate is not allowed to swing, a force P will be applied at the lower edge of the plate as shown in Fig. 17.7. The weight of the plate is acting vertically downward through the C.G. of the plate.

Let F = Force exerted by jet of water

h = Height of plate

= Distance of P from the hinge.

The jet strikes at the centre of the plate and hence distance of the centre of the jet from hinge = $\frac{h}{2}$.

Taking moments* about the hinge, O, $P \times h = F \times \frac{h}{2}$.

$$\therefore \quad P = \frac{F \times h}{2 \times h} = \frac{F}{2} = \frac{\rho a V^2}{2} \qquad \left(\because \quad F = \rho a V^2\right)$$

$$= 1000 \times \frac{.0007068 \times 20^2}{2} = \textbf{141.36 N. Ans.}$$

Problem 17.9 *A rectangular plate, weighing 58.86 N is suspended vertically by a hinge on the top of horizontal edge. The centre of gravity of the plate is 10 cm from the hinge. A horizontal jet of water 2 cm diameter, whose axis is 15 cm below the hinge impinges normally on the plate with a velocity of 5 m/s. Find the horizontal force applied at the centre of the gravity to maintain the plate in its vertical position. Find the corresponding velocity of the jet, if the plate is deflected through 30° and the same force continues to act at the centre of gravity of the plate.*

Solution. Given :

Weight of plate, $W = 58.86$ N

Distance of W from hinge, $x = 10$ cm $= 0.1$ m

Diameter of jet, $d = 2$ cm $= 0.02$ m

$\therefore$ Area, $$a = \frac{\pi}{4} d^2 = \frac{\pi}{4} \times .02^2 = .000314 \text{ m}^2$$

Distance of the axis of the jet of water from hinge = 15 cm = 0.15 m

Velocity of jet, $V = 5$ m/s

(*i*) Let the force applied at the centre of gravity of the plate to keep the plate in vertical position = P as shown in Fig. 17.8 (*a*).

* The weight of the plate is passing through the hinge O. Hence moment of W about hinge is zero.

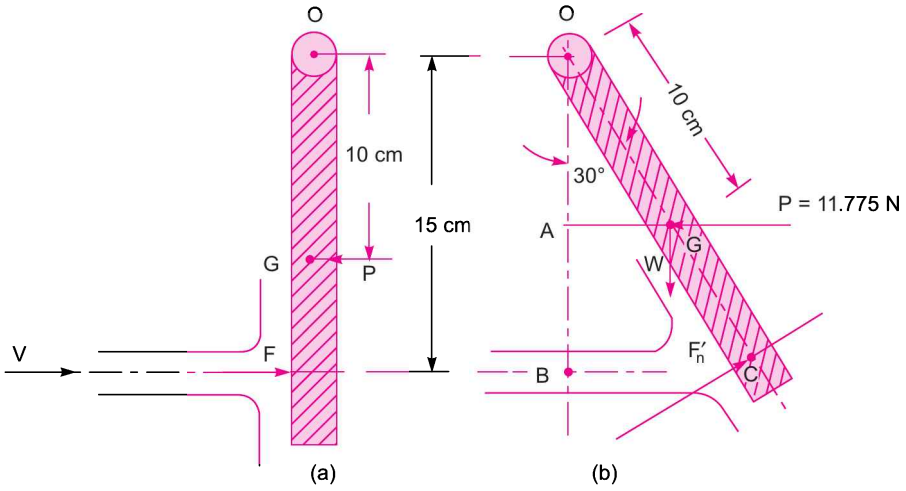


Fig. 17.8

The force exerted by a jet of water on the vertical plate,

$$F = \rho a V^2 = 1000 \times .000314 \times 5^2 = 7.85 \text{ N}$$

This force F is acting at a distance of 15 cm or 0.15 m from the hinge. Taking moments about hinge, we get

$$F \times 0.15 = P \times 0.10$$

$$\therefore \quad P = \frac{F \times 0.15}{0.10} = \frac{7.85 \times .15}{.10} = \textbf{11.775 N. Ans.}$$

(*ii*) The plate is deflected through an angle of 30° as shown in Fig. 17.8(*b*).

$\therefore$ Angle of swing = 30°

The force at the C.G. = P = 11.775 N

Let the velocity of the jet in this position = V m/s

For the deflected position of the plate as shown in Fig. 17.8 (*b*), the plate is in equilibrium under the action of three forces, which are :

(*i*) Weight of the plate, W acting at G at a distance 10 cm from O.

(*ii*) Horizontal force, P acting at G.

(*iii*) Normal force F_n' on the plate due to jet of water.

The angle between the jet and the plate, $\theta = 90° - 30° = 60°$

Hence, F_n' is given by equation (17.2) as

$$F_n' = \rho a V^2 \sin \theta = \rho a V^2 \sin 60°$$
$$= 1000 \times .000314 \times V^2 \times \sin 60° = 0.2717\ V^2$$

Taking moments of all forces about hinge O, we get

$$F_n' \times OC = P \times OA + W \times AG \qquad \ldots(i)$$

where $OB = OC \cos 30°$

$$\therefore \quad OC = \frac{OB}{\cos 30^\circ} = \frac{15}{\cos 30^\circ} = 17.32 \text{ cm} = 0.1732 \text{ m}$$

$$OA = OG \cos 30° = 10 \times .866 = 8.66 \text{ cm} = 0.0866 \text{ m}$$

$$AG = OG \sin 30° = 10 \times \frac{1}{2} = 5 \text{ cm} = .05 \text{ m}$$

Substituting these values in equation (*i*), we get

$$0.2717\ V^2 \times .1732 = 11.775 \times .0866 + 58.86 \times 0.05 = 3.962$$

$$\therefore \quad V = \sqrt{\frac{3.962}{0.2717 \times .1732}} = 9.175 \text{ m/s.}$$

Problem 17.10 *A jet of water of diameter 25 mm strikes a 20 cm ×20 cm square plate of uniform thickness with a velocity of 10 m/s at the centre of the plate which is suspended vertically by a hinge on its top horizontal edge. The weight of the plate is 98.1 N. The jet strikes normal to the plate. What force must be applied at the lower edge of the plate so that plate is kept vertical ? If the plate is allowed to deflect freely, what will be the inclination of the plate with vertical due to the force exerted by jet of water?*

Solution. Given :

Diameter of the jet, $d = 25 \text{ mm} = 25 \times 10^{-3} \text{ m} = .025 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4} (.025)^2 = .00049 \text{ m}^2$

Size of the plate, $= 20 \text{ cm} \times 20 \text{ cm}$

Weight of the plate, $W = 98.1 \text{ N}$

Velocity of jet, $V = 10 \text{ m/s}$

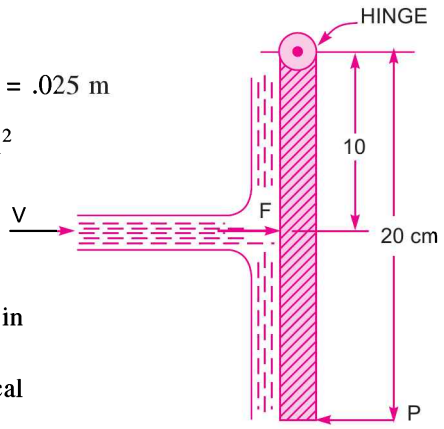


Fig. 17.9 (*a*)

(*i*) Let the force applied at the lower edge to keep the plate in vertical position is = P . See Fig.17.9 (*a*).

Force exerted by the jet of water at the centre of the vertical plate,

$$F = \rho a V^2$$
$$= 1000 \times .00049 \times 10^2 = 49 \text{ N.}$$

This force is acting at a distance of $\frac{20}{2} = 10$ cm from the hinge. The force P is acting at a distance of 20 cm from the hinge.

Taking moments about hinge,

$$F \times 10 = P \times 20$$

$$\therefore \quad 49 \times 10 = P \times 20$$

$$\therefore \quad P = \frac{49 \times 10}{20} = \mathbf{24.5 \text{ N. Ans.}}$$

(*ii*) When the plate is allowed to deflect freely about hinge.

Let the inclination of the plate with vertical = θ

In this position, the angle between the plate and jet will be

$$= (90° - \theta) .$$

$\therefore$ Force exerted by water normal to the plate is given by equation (17.2) as

$$F_n = \rho a V^2 \sin (90° - \theta) = \rho a V^2 \cos \theta$$

The distance $OB = \frac{OA}{\cos\theta} = \frac{10}{\cos\theta}$

The weight W of the plate is acting at a distance 10 cm from hinge. Distance

$$DG = OG \sin\theta = 10 \times \sin\theta$$

Taking moments about hinge, we get

$$F_n \times OB = W \times GD$$

or $$\rho a V^2 \cos\theta \times \frac{10}{\cos\theta} = W \times 10 \times \sin\theta$$

$\therefore$ $$\rho a V^2 = W \times \sin\theta$$

$\therefore$ $$\sin\theta = \frac{\rho a V^2}{W} = 1000 \times \frac{.00049 \times 10^2}{98.1} = 0.5$$

$\therefore$ $\theta = \mathbf{30°}$. **Ans.**

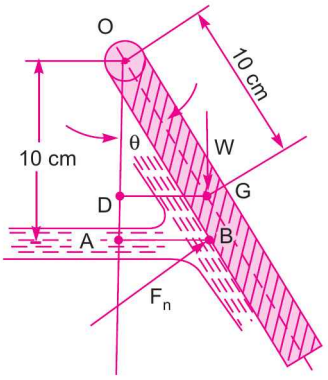


Fig. 17.9. (*b*)

Problem 17.10 (A) *A square plate of uniform thickness and length of side 300 mm hangs vertically from hinge at its top edge. When a horizontal water jet strikes the plate at its centre, the plate is deflected and comes to rest at angle of 30° to the vertical. The jet is 25 mm in diameter and has a velocity of 6 m/s. Determine the weight of the plate.*

Solution. Given :

Length of plate, $L = 300$ mm $= 0.3$ m

Angle of swing, or angle made by deflected plate with the vertical, $\theta = 30°$

Dia. of the jet, $d = 25$ mm $= 0.025$ m

$\therefore$ Area of jet, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} (0.025^2)$ m^2

Velocity of jet, $V = 6$ m/s

Let W = Weight of plate

Using equation (17.10), we get $\sin\theta = \frac{\rho \times a \times V^2}{W}$

$\therefore$ $$W = \frac{\rho^* \times a \times V^2}{\sin\theta} = \frac{1000 \times \left(\frac{\pi}{4} \times 0.025^2\right) \times 6^2}{\sin 30°} = \mathbf{35.33\ N.\ Ans.}$$

▶ 17.4 FORCE EXERTED BY A JET ON MOVING PLATES

The following cases of the moving plates will be considered :

1. Flat vertical plate moving in the direction of the jet and away from the jet,
2. Inclined plate moving in the direction of the jet, and
3. Curved plate moving in the direction of the jet or in the horizontal direction.

* If $\rho = 1000$ kg/m^3, then weight W will be in Newton.

17.4.1 Force on Flat Vertical Plate Moving in the Direction of Jet. Fig. 17.10 shows a jet of water striking a flat vertical plate moving with a uniform velocity away from the jet.

Let V = Velocity of the jet (absolute),
a = Area of cross-section of the jet,
u = Velocity of the flat plate.

In this case, the jet does not strike the plate with a velocity V, but it strikes with a relative velocity, which is equal to the absolute velocity of jet of water minus the velocity of the plate.

Hence relative velocity of the jet with respect to plate

$= (V - u)$

Mass of water striking the plate per sec

$= \rho \times$ Area of jet $\times$ Velocity with which jet strikes the plate

$= \rho a \times [V - u]$

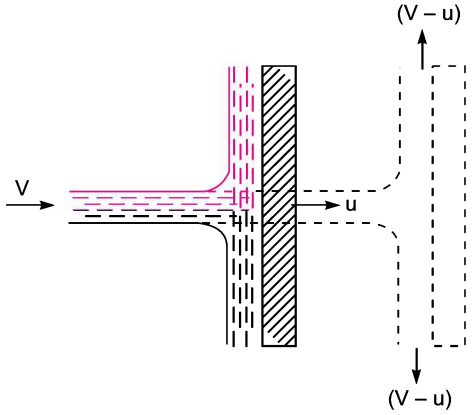


Fig. 17.10 *Jet striking a flat vertical moving plate.*

$\therefore$ Force exerted by the jet on the moving plate in the direction of the jet,

F_x = Mass of water striking per sec $\times$ [Initial velocity with which water strikes – Final velocity]

$= \rho a(V - u)\,[(V - u) - 0]$ ($\because$ Final velocity in the direction of jet is zero)

$= \rho a(V - u)^2$ ...(17.11)

In this case, the work will be done by the jet on the plate, as plate is moving. For the stationary plates, the work done is zero.

$\therefore$ Work done per second by the jet on the plate

$$= \text{Force} \times \frac{\text{Distance in the direction of force.}}{\text{Time}}$$

$= F_x \times u = \rho a(V - u)^2 \times u$ ...(17.12)

In equation (17.12), if the value of ρ for water is taken in S.I. units (*i.e.,* 1000 kg/m^3), the work done will be in N m/s. The term $\frac{\text{'Nm'}}{\text{s}}$ is equal to W (watt).

17.4.2 Force on the Inclined Plate Moving in the Direction of the Jet. Let a jet of water strikes an inclined plate, which is moving with a uniform velocity in the direction of the jet as shown in Fig. 17.11.

Let V = Absolute velocity of jet of water,
u = Velocity of the plate in the direction of jet,
a = Cross-sectional area of jet, and
θ = Angle between jet and plate.

Relative velocity of jet of water = $(V - u)$

$\therefore$ The velocity with which jet strikes = $(V - u)$

Mass of water striking per second

$= \rho \times a \times (V - u)$

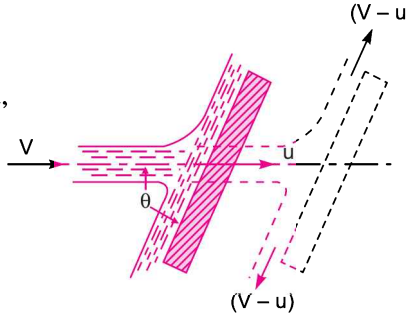


Fig. 17.11 *Jet striking an inclined moving plate.*

If the plate is smooth and loss of energy due to impact of the jet is assumed zero, the jet of water will leave the inclined plate with a velocity equal to $(V - u)$.

The force exerted by the jet of water on the plate in the direction normal to the plate is given as

F_n = Mass striking per second × [Initial velocity in the normal direction with which jet strikes – Final velocity]

$$= \rho a\,(V - u)\,[(V - u)\sin\theta - 0] = \rho a\,(V - u)^2 \sin\theta \qquad ...(17.13)$$

This normal force F_n is resolved into two components namely F_x and F_y in the direction of the jet and perpendicular to the direction of the jet respectively.

$$\therefore \qquad F_x = F_n \sin\theta = \rho a\,(V - u)^2 \sin^2\theta \qquad ...(17.14)$$

$$F_y = F_n \cos\theta = \rho a\,(V - u)^2 \sin\theta \cos\theta \qquad ...(17.15)$$

$\therefore$ Work done per second by the jet on the plate

$= F_x \times$ Distance per second in the direction of x

$$= F_x \times u = \rho a(V - u)^2 \sin^2\theta \times u = \rho a\,(V - u)^2\, u \sin^2\theta \text{ N m/s.} \qquad ...(17.16)$$

Problem 17.11 *A jet of water of diameter 10 cm strikes a flat plate normally with a velocity of 15 m/s. The plate is moving with a velocity of 6 m/s in the direction of the jet and away from the jet. Find:*

(i) the force exerted by the jet on the plate

(ii) work done by the jet on the plate per second.

Solution. Given :

Diameter of the jet, $d = 10 \text{ cm} = 0.1 \text{ m}$

$\therefore$ Area, $a = \dfrac{\pi}{4} d^2 = \dfrac{\pi}{4} (.1)^2 = .007854 \text{ m}^2$

Velocity of jet, $V = 15$ m/s

Velocity of the plate, $u = 6$ m/s.

(*i*) The force exerted by the jet on a moving flat vertical plate is given by equation (17.11),

$$F_x = \rho a\,(V - u)^2$$

$$= 1000 \times .007854 \times (15 - 6)^2 \text{ N} = \mathbf{636.17\ N.\ Ans.}$$

(*ii*) Work done per second by the jet

$$= F_x \times u = 636.17 \times 6 = \mathbf{3817.02\ Nm/s.\ Ans.}$$

Problem 17.12 *For Problem 17.11, find the power and efficiency of the jet.*

Solution. The given data from Problem 17.11 is

$$a = .007854 \text{ m}^2,\ V = 15 \text{ m/s},\ u = 6 \text{ m/s}$$

Also work done per second by the jet = 3817.02 Nm/s

(*i*) Power of the jet in kW $= \dfrac{\text{Work done per second}}{1000} = \dfrac{3817.02}{1000} = \mathbf{3.817\ kW.\ Ans.}$

(*ii*) Efficiency of the jet (η) $= \dfrac{\text{Output of the jet per second}}{\text{Input of the jet per second}}$ ...(*i*)

where output of jet/sec = Work done by jet per second = 3817.02 Nm/s

And input per second = Kinetic energy of the jet/sec

$$= \frac{1}{2}\left(\frac{\text{mass}}{\text{sec}}\right) V^2 = \frac{1}{2}(\rho a V) \times V^2 = \frac{1}{2}\rho a\, V^3$$

$$= \frac{1}{2} \times 1000 \times .007854 \times 15^3 \text{ Nm/s} = 13253.6 \text{ Nm/s}$$

$\therefore$ η of the jet $= \dfrac{3817.02}{13253.6} = 0.288 =$ **28.8%. Ans.**

Problem 17.12 (A) *A nozzle of 50 mm diameter delivers a stream of water at 20 m/s perpendicular to a plate that moves away from the jet at 5 m/s. Find :*

(i) the force on the plate,

(ii) the work done, and

(iii) the efficiency of jet. (J.N.T.U., Hyderabad S 2002)

Solution. Given :

Dia. of jet = 50 mm = 0.05 m

$\therefore$ Area, $a = \dfrac{\pi}{4}(0.05^2) = 0.0019635 \text{ m}^2$

Velocity of jet, $V = 20$ m/s, Velocity of plate, $u = 5$ m/s

(*i*) The force on the plate is given by equation (17.11) as,

$$F_x = \rho a\,(V - u)^2$$

$$= 1000 \times 0.0019635 \times (20 - 5)^2 = \textbf{441.78 N. Ans.}$$

(*ii*) The work done by the jet

$$= F_x \times u = 441.78 \times 5 = \textbf{2208.9 Nm/s. Ans.}$$

(*iii*) The efficiency of the jet, $\eta = \dfrac{\text{Output of jet}}{\text{Input of jet}}$

$$= \frac{\text{Work done/s}}{\text{K.E. of jet/s}} = \frac{F_x \times u}{\frac{1}{2} m V^2}$$

$$= \frac{F_x \times u}{\frac{1}{2}(\rho A V) \times V^2}$$

$$= \frac{2208.9}{\frac{1}{2}(1000 \times 0.0019635 \times 20) \times 20^2} = \frac{2208.9}{6540}$$

$$= 0.3377 = \textbf{33.77\%. Ans.}$$

Problem 17.13 *A 7.5 cm diameter jet having a velocity of 30 m/s strikes a flat plate, the normal of which is inclined at 45° to the axis of the jet. Find the normal pressure on the plate : (i) when the plate is stationary , and (ii) when the plate is moving with a velocity of 15 m/s and away from the jet. Also determine the power and efficiency of the jet when the plate is moving.*

Solution. Given :

Diameter of the jet, $d = 7.5 \text{ cm} = 0.075 \text{ m}$

$\therefore$ Area, $a = \frac{\pi}{4} (.075)^2 = .004417 \text{ m}^2$

Angle between the jet and plate $\theta = 90° - 45° = 45°$

Velocity of jet, $V = 30$ m/s.

(*i*) When the plate is stationary, the normal force on the plate is given by equation (17.2) as

$$F_n = \rho a V^2 \sin \theta = 1000 \times .004417 \times 30^2 \times \sin 45° = \textbf{2810.96 N. Ans.}$$

(*ii*) When the plate is moving with a velocity 15 m/s and away from the jet, the normal force on the plate is given by equation (17.13) as

$$F_n = \rho a (V - u)^2 \sin \theta \qquad \text{where } u = 15 \text{ m/s.}$$
$$= 1000 \times .004417 \times (30 - 15)^2 \times \sin 45° = \textbf{702.74 N. Ans.}$$

(*iii*) The power and efficiency of the jet when plate is moving is obtained as

Work done per second by the jet

= Force in the direction of jet × Distance moved by the plate in the direction of jet/sec

= $F_x \times u$, where $F_x = F_n \sin \theta = 702.74 \times \sin 45° = 496.9$ N

Work done/sec = $496.9 \times 15 = 7453.5$ Nm/s

$\therefore$ $$\text{Power in kW} = \frac{\text{Work done per second}}{1000} = \frac{7453.5}{1000} = \textbf{7.453 kW. Ans.}$$

$$\text{Efficiency of the jet} = \frac{\text{Output}}{\text{Input}} = \frac{\text{Work done per second}}{\text{Kinetic energy of the jet}}$$

$$= \frac{7453.5}{\frac{1}{2}(\rho a V) \times V^2} = \frac{7453.5}{\frac{1}{2}\rho a V^3} = \frac{7453.5}{\frac{1}{2} \times 1000 \times .004417 \times 30^3}$$

$$= 0.1249 \simeq 0.125 = \textbf{12.5\%. Ans.}$$

17.4.3 Force on the Curved Plate when the Plate is Moving in the Direction of Jet. Let a jet of water strikes a curved plate at the centre of the plate which is moving with a uniform velocity in the direction of the jet as shown in Fig. 17.12.

Let V = Absolute velocity of jet,

a = Area of jet,

u = Velocity of the plate in the direction of the jet.

Relative velocity of the jet of water or the velocity with which jet strikes the curved plate = $(V - u)$.

If plate is smooth and the loss of energy due to impact of jet is zero, then the velocity with which the jet will be leaving the curved vane = $(V - u)$.

This velocity can be resolved into two components, one in the direction of the jet and other perpendicular to the direction of the jet.

Component of the velocity in the direction of jet

$$= -(V - u) \cos \theta$$

(–ve sign is taken as at the outlet, the component is in the opposite direction of the jet).

Component of the velocity in the direction perpendicular to the direction of the jet = $(V - u) \sin \theta$.

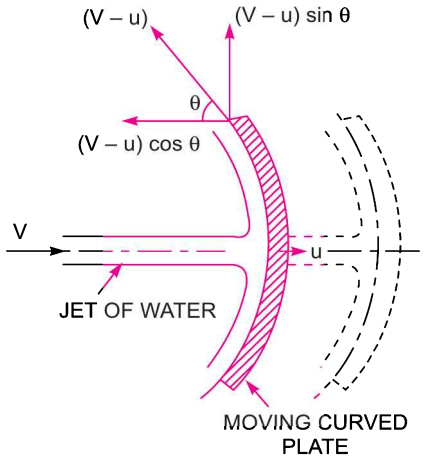


Fig. 17.12 *Jet striking a curved moving plate.*

Mass of the water striking the plate = $\rho \times a \times$ Velocity with which jet strikes the plate

$= \rho a(V - u)$

$\therefore$ Force exerted by the jet of water on the curved plate in the direction of the jet,

F_x = Mass striking per sec × [Initial velocity with which jet strikes the plate in the direction of jet – Final velocity]

$= \rho a(V - u)\ [(V - u) - (-(V - u) \cos \theta)]$

$= \rho a(V - u)\ [(V - u) + (V - u) \cos \theta]$

$= \rho a(V - u)^2\ [1 + \cos \theta]$ ...(17.17)

Work done by the jet on the plate per second

$= F_x \times$ Distance travelled per second in the direction of x

$= F_x \times u = \rho a(V - u)^2\ [1 + \cos \theta] \times u$

$= \rho a(V - u)^2 \times u\ [1 + \cos \theta]$ ...(17.18)

Problem 17.14 *A jet of water of diameter 7.5 cm strikes a curved plate at its centre with a velocity of 20 m/s. The curved plate is moving with a velocity of 8 m/s in the direction of the jet. The jet is deflected through an angle of 165°. Assuming the plate smooth find :*

(i) Force exerted on the plate in the direction of jet, (ii) Power of the jet, and (iii) Efficiency of the jet.

Solution. Given :

Diameter of the jet, $d = 7.5$ cm $= 0.075$ m

$\therefore$ Area, $a = \frac{\pi}{4}(.075)^2 = 0.004417$

Velocity of the jet, $V = 20$ m/s

Velocity of the plate, $u = 8$ m/s

Angle of deflection of the jet, $= 165°$

$\therefore$ Angle made by the relative velocity at the outlet of the plate,

$\theta = 180° - 165° = 15°$.

(*i*) Force exerted by the jet on the plate in the direction of the jet is given by equation (17.17) as

$= F_x = \rho a(V - u)^2\ (1 + \cos \theta)$

$= 1000 \times .004417 \times (20 - 8)^2\ [1 + \cos 15°] =$ **1250.38 N. Ans.**

(*ii*) Work done by the jet on the plate per second

$= F_x \times u = 1250.38 \times 8 = 10003.04$ N m/s

$\therefore$ Power of the jet $= \frac{10003.04}{1000} =$ **10 kW. Ans.**

(*iii*) Efficiency of the jet $= \frac{\text{Output}}{\text{Input}} = \frac{\text{Work done by jet/sec}}{\text{Kinetic energy of jet/sec}}$

$= \frac{1250.38 \times 8}{\frac{1}{2}(\rho a V) \times V^2} = \frac{1250.38 \times 8}{\frac{1}{2} \times 1000 \times .004417 \times V^3}$

$= \frac{1250.38 \times 8}{\frac{1}{2} \times 1000 \times .004417 \times 20^3} = 0.564 =$ **56.4%. Ans**

Problem 17.15 *A jet of water from a nozzle is deflected through 60° from its original direction by a curved plate which it enters tangentially without shock with a velocity of 30 m/s and leaves with a mean velocity of 25 m/s. If the discharge from the nozzle is 0.8 kg/s, calculate the magnitude and direction of the resultant force on the vane, if the vane is stationary.*

Solution. Given :

Velocity at inlet, $V_1 = 30$ m/s

Velocity at outlet, $V_2 = 25$ m/s

Mass per second $= 0.8$ kg/s

Force in the direction of jet,

$$F_x = \text{Mass per second} \times (V_{1x} - V_{2x})$$

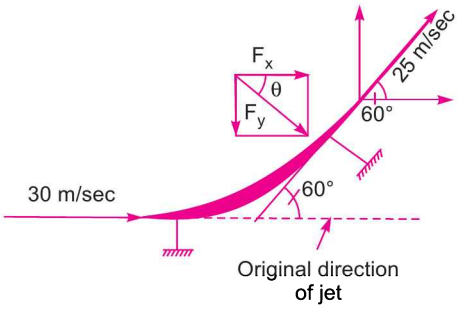


Fig. 17.13 (*a*)

where V_{1x} = Initial velocity in the direction of x

$= 30$ m/s

V_{2x} = Final velocity in the direction of x

$$= 25 \cos 60^\circ = 25 \times \frac{1}{2} = 12.5 \text{ m/s}$$

$$\therefore \quad F_x = 0.8[30 - 12.5] = 0.8 \times 17.5 = 14.0 \text{ N}$$

Similarly, force normal to the jet,

$$F_y = \text{Mass per second} \times (V_{1y} - V_{2y})$$

$$= 0.8\ [0 - 25 \sin 60^\circ] = -17.32 \text{ N}$$

–ve sign means the force, F_y, is acting in the vertically downward direction.

$\therefore$ Resultant force on the vane $= \sqrt{F_x^2 + F_y^2} = \sqrt{14^2 + (-17.32)^2} =$ **22.27 N. Ans.**

The angle made by the resultant with x-axis

$$\tan \theta = \frac{F_y}{F_x} = \frac{-17.32}{14.0} = -1.237$$

–ve sign means the angle θ is in the clockwise direction with x- axis as shown in Fig. 17.13 (*a*)

$$\therefore \quad \theta = \tan^{-1} 1.237 = \mathbf{51^\circ\ 2.86'.\ Ans.}$$

Problem 17.16 (*a*) *A stationary vane having an inlet angle of zero degree and an outlet angle of 25° as shown in Fig. 17.13(b), receives water at a velocity of 50 m/s. Determine the components of force acting on it in the direction of the jet velocity and normal to it. Also find the resultant force in magnitude and direction per unit weight of the flow.*

(b) If the vane stated above is moving with a velocity of 20 m/s in the direction of the jet, calculate the force components in the direction of the vane velocity and across it, also the resultant force in magnitude and direction. Calculate the work done and power developed per unit weight of the flow.

Solution. Given :

(*a*) Velocity of jet, $V = 50$ m/s

Angle at outlet, $= 25^\circ$

For the stationary vane, the force in the direction of jet is given as

$$F_x = \text{Mass per sec} \times [V_{1x} - V_{2x}]$$

where $V_{1x} = 50$ m/s

$V_{2x} = -50 \cos 25^\circ = -45.315$

∴ Force in the direction of jet per unit weight of water

$$= \frac{\text{Mass/sec}\,[50 - (-45.315)]}{\text{Weight of water/sec}}$$

or
$$F_x = \frac{(\text{Mass/sec})\,[50 + 45.315]}{(\text{Mass/sec}) \times g}$$

$$= \frac{1}{g}\,[50 + 45.315]\ \text{N/N} = \frac{95.315}{9.81} = 9.716\ \text{N/N}$$

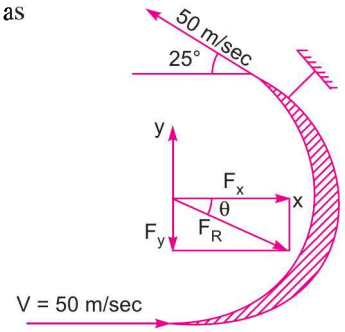


Fig. 17.13 (*b*)

Force exerted by jet in the direction perpendicular to the direction of the jet per unit weight of the flow,

$$F_y = \frac{(\text{Mass per sec})\,[V_{1y} - V_{2y}]}{g \times \text{Mass per sec}}$$

$$= \frac{1}{g}\,[V_{1y} - V_{2y}] = \frac{1}{g}\,[0 - 50 \sin 25^\circ] \quad (\because V_{1y} = 0,\ V_{2y} = 50 \sin 25^\circ)$$

$$= \frac{-50 \sin 25^\circ}{9.81} = \mathbf{-\ 2.154.\ Ans.}$$

–ve sign means the force F_y is acting in the downward direction.

∴ Resultant force per unit weight of water = $\sqrt{F_x^2 + F_y^2}$

or
$$\mathbf{F_R} = \sqrt{(9.716)^2 + (2.154)^2} = \mathbf{9.952\ N.\ Ans.}$$

The angle made by the resultant with the x-axis,

$$\tan\theta = \frac{F_y}{F_x} = \frac{2.154}{9.716} = 0.2217$$

∴ $\theta = \tan^{-1} .2217 = \mathbf{12.50^\circ.\ Ans.}$

(*b*) Velocity of the vane = 20 m/s.

When the vane is moving in the direction of the jet, the force exerted by the jet on the plate in the direction of jet,

F_x' = [Mass of water striking/sec] × $[V_{1x} - V_{2x}]$

where V_{1x} = Initial velocity of the striking water

$= (V - u) = 50 - 20 = 30$ m/s

V_{2x} = Final velocity in the direction of x

$= -(V - u) \cos 25^\circ = 30 \times \cos 25^\circ = -27.189$ m/s.

∴ F_x = Mass per sec [30 + 27.189]

Force in the direction of jet per unit weight,

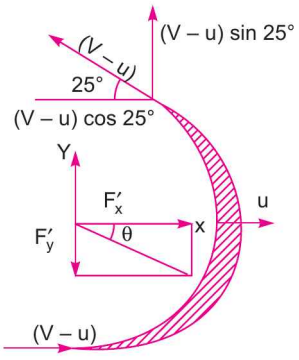


Fig. 17.14

$$F_x' = \frac{\text{Mass per sec } [30 + 27.189]}{\text{Mass per sec} \times g}$$

$$= \frac{(30 + 27.189)}{9.81} = 5.829 \text{ N.}$$

Force exerted by the jet in the direction perpendicular to direction of jet, per unit weight,

$$F_y' = \frac{1}{g} [V_{1y} - V_{2y}]$$

where $V_{1y} = 0$; $V_{2y} = (V - u) \sin 25^\circ = (50 - 20) \sin 25^\circ = 30 \sin 25^\circ$

$$F_y' = \frac{1}{9.81} [0 - 30 \sin 25^\circ] = -1.292 \text{ N}$$

$\therefore$ Resultant force $= \sqrt{(5.829)^2 + (1.292)^2} = 5.917$ N

The angle made by the resultant with x-axis, $\tan \theta = \frac{1.292}{5.829} = 0.2217$

$\therefore$ $\theta = \tan^{-1} .2217 = 12.30^\circ$

$\therefore$ Work done per second per unit weight of flow

$= F_x' \times u = 5.829 \times 20 = 116.58$ N m/s

$\therefore$ Power developed $= \frac{\text{Work done per second}}{1000} = \frac{116.58}{1000} =$ **0.116 kW. Ans.**

Problem 17.17 *A jet of water of diameter 50 mm moving with a velocity of 25 m/s impinges on a fixed curved plate tangentially at one end at an angle of 30° to the horizontal. Calculate the resultant force of the jet on the plate if the jet is deflected through an angle of 50°. Take g = 10 m/s²*

Solution. Given :

Dia. of jet, $d = 50$ mm $= 0.05$ m

$\therefore$ Area of jet, $a = \frac{\pi}{4} (0.05)^2 \text{m}^2$

Velocity of jet, $V = 25$ m/s

The angle made by the jet at inlet with horizontal, $\theta = 30^\circ$

Angle of deflection $= 50^\circ$

$\therefore$ Angle made by the jet at outlet with horizontal is given by,

$\phi = \theta +$ Angle of deflection

$= 30^\circ + 50^\circ = 80^\circ$

Value of g $= 10$ m/s²

The force exerted by the jet of water in the direction of x is given by,

$$F_x = \rho a V [V_{1x} - V_{2x}] \qquad ...(i)$$

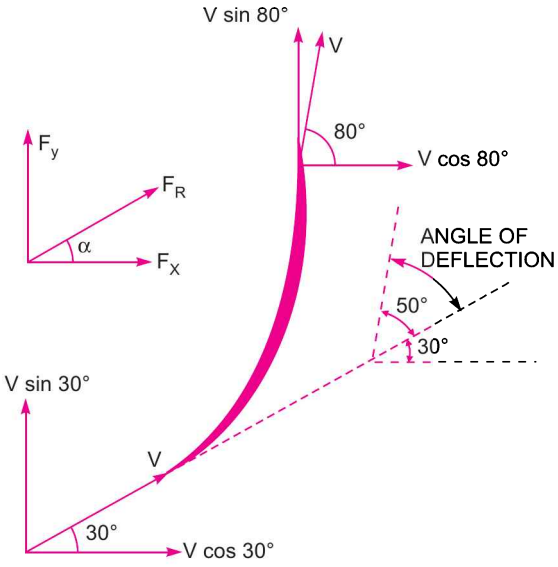


Fig. 17.14 (*a*)

where $\rho = 1000$ ($\because$ g is given as 10 m/s^2)

$$a = \frac{\pi}{4}(0.05)^2;\ V = 25 \text{ m/s};$$

$$V_{1x} = V \cos 30° = 25 \cos 30°,$$

$$V_{2x} = V \cos 80° = 25 \cos 80°.$$

Substituting these values in equation (i), we get

$$F_x = 1000 \times \frac{\pi}{4}(0.05)^2 \times 25\,[25 \cos 30° - 25 \cos 80°] = 849.7 \text{ N}$$

The force in the direction of y is given by,

$$F_y = \rho a V[V_{1y} - V_{2y}]$$

$$= 1000 \times \frac{\pi}{4}(0.05)^2 \times 25[25 \sin 30° - 25 \sin 80°] = -594.9 \text{ N}$$

The –ve sign shows that force F_y is acting in the downward direction.

The resultant force is given by,

$$F_R = \sqrt{F_x^2 + F_y^2}$$

$$= \sqrt{849.7^2 + 594.9^2} = \mathbf{1037\ N.\ Ans.}$$

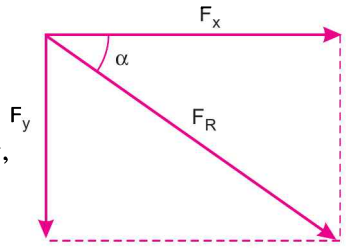


Fig. 17.14 (b)

And the angle made by the resultant with the horizontal is given by,

$$\tan \alpha = \frac{F_y}{F_x} = \frac{594.9}{849.7} = 0.7$$

$$\therefore \quad \alpha = \tan^{-1} 0.7 = \mathbf{35°.}\ \mathbf{Ans.}$$

17.4.4 Force Exerted by a Jet of Water on an Unsymmetrical Moving Curved Plate when Jet Strikes Tangentially at one of the Tips. Fig. 17.15 shows a jet of water striking a moving curved plate (also called vane) tangentially, at one of its tips. As the jet strikes tangentially, the loss of energy due to impact of the jet will be zero. In this case as plate is moving, the velocity with which jet of water strikes is equal to the relative velocity of the jet with respect to the plate. Also as the plate is moving in different direction of the jet, the relative velocity at inlet will be equal to the vector difference of the velocity of jet and velocity of the plate at inlet.

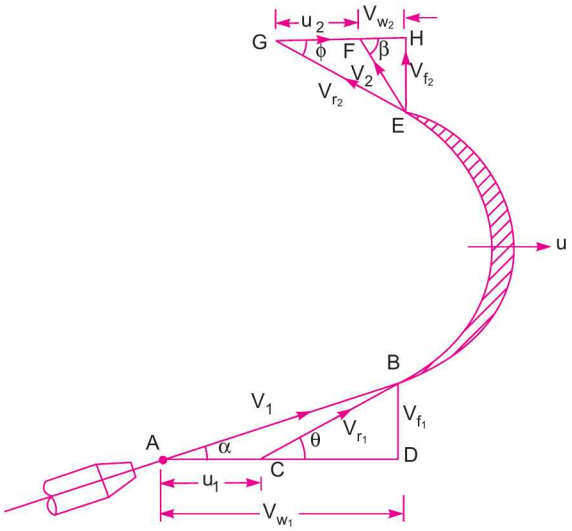


Fig. 17.15 *Jet striking a moving curved vane at one of the tips.*

Let V_1 = Velocity of the jet at inlet.

u_1 = Velocity of the plate (vane) at inlet.

V_{r_1} = Relative velocity of jet and plate at inlet.

α = Angle between the direction of the jet and direction of motion of the plate, also called guide blade angle.

θ = Angle made by the relative velocity (V_{r_2}) with the direction of motion at inlet also called vane angle at inlet.

V_{w_1} and V_{f_1} = The components of the velocity of the jet V_1, in the direction of motion and perpendicular to the direction of motion of the vane respectively.

V_{w_1} = It is also known as velocity of whirl at inlet.

V_{f_1} = It is also known as velocity of flow at inlet.

V_2 = Velocity of the jet, leaving the vane or velocity of jet at outlet of the vane.

u_2 = Velocity of the vane at outlet.

V_{r_2} = Relative velocity of the jet with respect to the vane at outlet.

β = Angle made by the velocity V_2 with the direction of motion of the vane at outlet.

ϕ = Angle made by the relative velocity V_{r_2} with the direction of motion of the vane at outlet and also called vane angle at outlet.

V_{w_1} and V_{f_1} = Components of the velocity V_2, in the direction of motion of vane and perpendicular to the direction of motion of vane at outlet.

V_{w_2} = It is also called the velocity of whirl at outlet.

V_{f_2} = Velocity of flow at outlet.

The triangles *ABD* and *EGH* are called the velocity triangles at inlet and outlet. These velocity triangles are drawn as given below :

1. Velocity Triangle at Inlet. Take any point *A* and draw a line $AB = V_1$ in magnitude and direction which means line *AB* makes an angle α with the horizontal line *AD*. Next draw a line $AC = u_1$ in magnitude. Join *C* to *B*. Then *CB* represents the relative velocity of the jet at inlet. If the loss of energy at inlet due to impact is zero, then *CB* must be in the tangential direction to the vane at inlet. From *B* draw a vertical line *BD* in the downward direction to meet the horizontal line *AC* produced at *D*.

Then BD = Represents the velocity of flow at inlet = V_{f_1}

AD = Represents the velocity of whirl at inlet = V_{w_1}

$\angle BCD$ = Vane angle at inlet = θ.

2. Velocity Triangle at Outlet. If the vane surface is assumed to be very smooth, the loss of energy due to friction will be zero. The water will be gliding over the surface of the vane with a relative velocity equal to V_{r_1} and will come out of the vane with a relative relocity V_{r_2}. This means that the relative velocity at outlet $V_{r_2} = V_{r_1}$. And also the relative velocity at outlet should be in tangential direction to the vane at outlet.

Draw *EG* in the tangential direction of the vane at outlet and cut $EG = V_{r_2}$. From *G*, draw a line *GF* in the direction of vane at outlet and equal to u_2, the velocity of the vane at outlet. Join *EF*. Then *EF* represents the absolute velocity of the jet at outlet in magnitude and direction. From *E* draw a vertical line *EH* to meet the line *GF* produced at *H*. Then

EH = Velocity of flow at outlet = V_{f_2}

FH = Velocity of whirl at outlet = V_{w_2}

ϕ = Angle of vane at outlet.

β = Angle made by V_2 with the direction of motion of vane at outlet.

If the vane is smooth and is having velocity in the direction of motion at inlet and outlet equal then we have

$u_1 = u_2 = u$ = Velocity of vane in the direction of motion and

$V_{r_1} = V_{r_2}$.

Now mass of water striking vane per sec = $\rho a V_{r_1}$ ...(*i*)

where a = Area of jet of water, V_{r_1} = Relative velocity at inlet.

∴ Force exerted by the jet in the direction of motion

F_x = Mass of water striking per sec × [Initial velocity with which jet strikes in the direction of motion – Final velocity of jet in the direction of motion] ...(*ii*)

But initial velocity with which jet strikes the vane = V_{r_1}

The component of this velocity in the direction of motion

$= V_{r_1} \cos\theta = (V_{w_1} - u_1)$ (See Fig. 17.15)

Similarly, the component of the relative velocity at outlet in the direction of motion = $- V_{r_2} \cos\phi$

$= - [u_2 + V_{w_2}]$

–ve sign is taken as the component of V_{r_2} in the direction of motion is in the opposite direction.

Substituting the equation (*i*) and all above values of the velocities in equation (*ii*), we get

$$F_x = \rho a V_{r_1} [(V_{w_1} - u_1) - \{ - (u_2 + V_{w_2})\}] = \rho a V_{r_1} [V_{w_1} - u_1 + u_2 + V_{w_2}]$$

$$= \rho a V_{r_1} [V_{w_1} + V_{w_2}] \quad (\because \ u_1 = u_2) \quad ...(iii)$$

Equation (*iii*) is true only when angle β shown in Fig. 17.15 is an acute angle. If $\beta = 90°$, the $V_{w_2} = 0$, then equation (*iii*) becomes as,

$$F_x = \rho a V_{r_1} [V_{w_1}]$$

If β is an obtuse angle, the expression for F_x will become

$$F_x = \rho a V_{r_1} [V_{w_1} - V_{w_2}]$$

Thus in general, F_x is written as $F_x = \rho a V_{r_1} [V_{w_1} \pm V_{w_2}]$ ...(17.19)

Work done per second on the vane by the jet

= Force × Distance per second in the direction of force

$= F_x \times u = \rho a V_{r_1} [V_{w_1} \pm V_{w_2}] \times u$ ...(17.20)

∴ Work done per second per unit weight of fluid striking per second

$$= \frac{\rho a V_{r_1} \left[V_{w_1} \pm V_{w_2}\right] \times u}{\text{Weight of fluid striking/s}} \frac{\text{Nm/s}}{\text{N/s}} = \frac{\rho a V_{r_1} \left[V_{w_1} \pm V_{w_2}\right] \times u}{g \times \rho a V_{r_1}} = \text{Nm/N}$$

$$= \frac{1}{g} [V_{w_1} \pm V_{w_2}] \times u \text{ Nm/N} \quad ...(17.21)$$

Work done/sec per unit mass of fluid striking per second

$$= \frac{\rho a V_{r_1}\left[V_{w_1} \pm V_{w_2}\right] \times u}{\text{Mass of fluid striking / s}} \frac{\text{Nm / s}}{\text{kg / s}} = \frac{\rho a V_{r_1}\left[V_{w_1} \pm V_{w_2}\right] \times u}{\rho a V_{r_1}} \text{ Nm/kg}$$

$$= (V_{w_1} \pm V_{w_2}) \times u \text{ Nm/kg} \qquad ...[17.21(A)]$$

Note. Equation (17.21) gives the work done per unit weight whereas equation [17.21(*A*)] gives the work done per unit mass.

3. Efficiency of Jet. The work done by the jet on the vane given by equation (17.20), is the output of the jet whereas the initial kinetic energy of the jet is the input. Hence, the efficiency (η) of the jet is expressed as

$$\eta = \frac{\text{Output}}{\text{Input}} = \frac{\text{Work done per second on the vane}}{\text{Initial K. E. per second of the jet}} = \frac{\rho a V_{r_1}(V_{w_1} \pm V_{w_2}) \times u}{\frac{1}{2} m V_1^2}$$

where m = mass of the fluid per second in the jet = $\rho a V_1$

V_1 = initial velocity of jet

$$\therefore \qquad \eta = \frac{\rho a V_{r_1}\left[V_{w_1} \pm V_{w_2}\right] \times u}{\frac{1}{2}(\rho a V_1) \times V_1^2}. \qquad ...[17.21(B)]$$

Problem 17.18 *A jet of water having a velocity of 20 m/s strikes a curved vane, which is moving with a velocity of 10 m/s. The jet makes an angle of 20° with the direction of motion of vane at inlet and leaves at an angle of 130° to the direction of motion of vane an outlet. Calculate :*

(i) Vane angles, so that the water enters and leaves the vane without shock.

(ii) Work done per second per unit weight of water striking (or work done per unit weight of water striking) the vane per second.

Solution. Given :

Velocity of jet, $V_1 = 20$ m/s

Velocity of vane, $u_1 = 10$ m/s

Angle made by jet at inlet, with direction of motion of vane, $\alpha = 20°$

Angle made by the leaving jet, with the direction of motion $= 130°$

$\therefore \quad \beta = 180° - 130° = 50°$

In this problem, $u_1 = u_2 = 10$ m/s

$V_{r_1} = V_{r_2}$

(*i*) **Vane Angle** means angle made by the relative velocities at inlet and outlet, *i.e.*, θ and ϕ.

From Fig. 17.16, in ΔABD, we have $\tan \theta = \frac{BD}{CD}$

$$= \frac{V_{f_1}}{AD - AC} = \frac{V_{f_1}}{V_{w_1} - u_1} \qquad ...(i)$$

where $V_{f_1} = V_1 \sin \alpha = 20 \times \sin 20^o = 6.84$ m/s

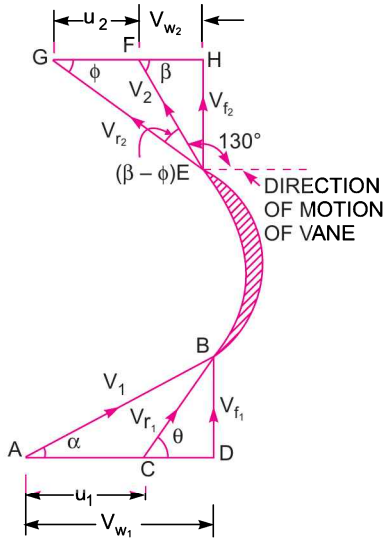


Fig. 17.16

$V_{w_1} = V_1 \cos \alpha = 20 \times \cos 20^\circ = 18.794$ m/s.

$u_1 = 10$ m/s

$$\therefore \quad \tan \theta = \frac{6.84}{18.794 - 10} = .7778 \text{ or } \theta = 37.875^\circ$$

$$\therefore \quad \theta = \mathbf{37^\circ\ 52.5'.\ Ans.}$$

From, ΔABC, $\quad \sin \theta = \dfrac{V_{f_1}}{V_{r_1}}$ or $V_{r_1} = \dfrac{V_{f_1}}{\sin \theta} = \dfrac{6.84}{\sin 37.875^\circ} = 11.14$

$$\therefore \quad V_{r_2} = V_{r_1} = 11.14 \text{ m/s.}$$

From, ΔEFG, applying sine rule, we have

$$\frac{V_{r_2}}{\sin (180^\circ - \beta)} = \frac{u_2}{\sin (\beta - \phi)}$$

or $$\frac{11.14}{\sin \beta} = \frac{10}{\sin [\beta - \phi]} \quad \text{or} \quad \frac{11.14}{\sin 50^\circ} = \frac{10}{\sin [50^\circ - \phi]} \qquad (\because \ \beta = 50^\circ)$$

$$\therefore \quad \sin (50^\circ - \phi) = \frac{10 \times \sin 50^\circ}{11.14} = 0.6876 = \sin 43.44^\circ$$

$$\therefore \quad 50^\circ - \phi = 43.44^\circ \quad \text{or} \quad \phi = 50^\circ - 43.44^\circ = 6.56^\circ$$

$$\therefore \quad \phi = \mathbf{6^\circ\ 33.6'.\ Ans.}$$

(*ii*) Work done per second per unit weight of the water striking the vane per second is given by equation (17.21) as

$$= \frac{1}{g} [V_{w_1} + V_{w_2}] \times u \text{ Nm/N (+ ve sign is taken as } \beta \text{ is an acute angle)}$$

where $V_{w_1} = 18.794$ m/s, $V_{w_2} = GH - GF = V_{r_2} \cos \phi - u_2 = 11.14 \times \cos 6.56^\circ - 10 = 1.067$ m/s

$u = u_1 = u_2 = 10$ m/s

$\therefore$ Work done per unit weight of water

$$= \frac{1}{9.81} [18.794 + 1.067] \times 10 \text{ Nm/N} = \mathbf{20.24\ Nm/N.\ Ans.}$$

Problem 17.19 *A jet of water having a velocity of 40 m/s strikes a curved vane, which is moving with a velocity of 20 m/s. The jet makes an angle of 30° with the direction of motion of vane at inlet and leaves at an angle of 90° to the direction of motion of vane at outlet. Draw the velocity triangles at inlet and outlet and determine the vane angles at inlet and outlet so that the water enters and leaves the vane without shock.*

Solution. Given :

Velocity of jet, $V_1 = 40$ m/s

Velocity of vane, $u_1 = 20$ m/s

Angle made by jet at inlet, $\alpha = 30^\circ$

Angle made by leaving jet $= 90^\circ$

$\therefore$ $\beta = 180^\circ - 90^\circ = 90^\circ$

For this problem, we have

$$u_1 = u_2 = u = 20 \text{ m/s}$$

Vane angles at inlet and outlet are θ and ϕ respectively.

From ΔBCD, we have

$$\tan\theta = \frac{BD}{CD} = \frac{BD}{AD - AC} = \frac{V_{f_1}}{V_{w_1} - u_1}$$

where $V_{f_1} = V_1 \sin\alpha = 40 \times \sin 30^\circ = 20$ m/s

$V_{w_1} = V_1 \cos\alpha = 40 \times \cos 30^\circ = 34.64$ m/s

$u_1 = 20$ m/s

$$\therefore \quad \tan\theta = \frac{20}{34.64 - 20} = \frac{20}{14.64} = 1.366 = \tan 53.79^\circ$$

$$\therefore \quad \theta = 53.79^\circ \quad \text{or} \quad \mathbf{53^\circ\ 47.4'.\ Ans.}$$

Also from ΔBCD,

$$\sin\theta = \frac{V_{f_1}}{V_{r_1}} \quad \text{or} \quad V_{r_1} = \frac{V_{f_1}}{\sin\theta} = \frac{20}{\sin 53.79^\circ}$$

$$\therefore \quad V_{r_1} = 24.78$$

But $V_{r_2} = V_{r_1} = 24.78$

Hence, from ΔEFG, $\cos\phi = \frac{u_2}{V_{r_2}} = \frac{20}{24.78} = 0.8071 = \cos 36.18^\circ$

$$\therefore \quad \phi = 36.18^\circ \quad \text{or} \quad \mathbf{36^\circ\ 10.8'.\ Ans.}$$

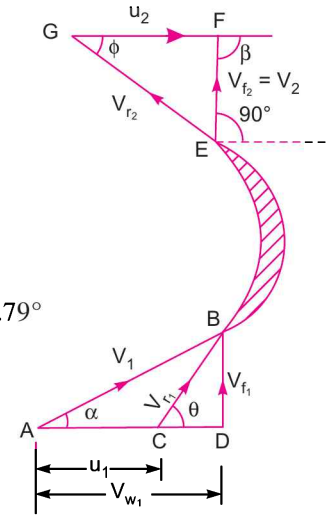


Fig. 17.17

Problem 17.20 *A jet of water of diameter 50 mm, having a velocity of 20 m/s strikes a curved vane which is moving with a velocity of 10 m/s in the direction of the jet. The jet leaves the vane at an angle of 60° to the direction of motion of vane at outlet. Determine :*

(i) The force exerted by the jet on the vane in the direction of motion.

(ii) Work done per second by the jet.

Solution. Given :

Diameter of the jet, $d = 50$ mm $= 0.05$ m

$$\therefore \quad \text{Area,} \quad a = \frac{\pi}{4}(.05)^2 = .001963 \text{ m}^2$$

Velocity of jet, $V_1 = 20$ m/s

Velocity of vane, $u_1 = 10$ m/s

As jet and vane are moving in the same direction,

$$\therefore \quad \alpha = 0$$

Angle made by the leaving jet, with the direction of motion $= 60^\circ$

$$\therefore \quad \beta = 180^\circ - 60^\circ = 120^\circ$$

For this problem, we have

$$u_1 = u_2 = u = 10 \text{ m/s}$$

$$V_{r_1} = V_{r_2}$$

From Fig. 17.18, we have

$$V_{r_1} = AB - AC = V_1 - u_1$$

$$= 20 - 10 = 10 \text{ m/s}$$

$$V_{w_1} = V_1 = 20 \text{ m/s}$$

$$\therefore \quad V_{r_2} = V_{r_1} = 10 \text{ m/s}$$

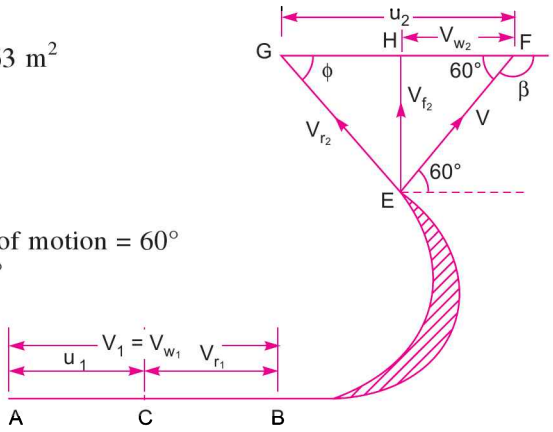


Fig. 17.18

Now in ΔEFG, $EG = V_{r_2} = 10$ m/s,

$GF = u_2 = 10$ m/s

$\angle GEF = 180^\circ - (60^\circ + \phi) = (120^\circ - \phi)$

From sine rule, we have

$$\frac{EG}{\sin 60^\circ} = \frac{GF}{\sin (120^\circ - \phi)} \quad \text{or} \quad \frac{10}{\sin 60^\circ} = \frac{10}{\sin (120^\circ - \phi)}$$

or $\sin 60^\circ = \sin (120^\circ - \phi)$

$\therefore$ $60^\circ = 120^\circ - \phi$ or $\phi = 120^\circ - 60^\circ = 60^\circ$

Now $V_{w_2} = HF = GF - GH$

$= u_2 - V_{r_2} \cos \phi = 10 - 10 \times \cos 60^\circ = 10 - 5 = 5$ m/s.

(*i*) The force exerted by the jet on the vane in the direction of motion is given by equation (17.19) as

$F_x = \rho a V_{r_1} [V_{w_1} - V_{w_2}]$ (–ve sign is taken as β is an obtuse angle)

$= 1000 \times .001963 \times 10 [20 - 5]$ N = **294.45 N. Ans.**

(*ii*) Work done per second by the jet

$= F_x \times u = 294.45 \times 10 = 2944.5$ N m/s

= **2944.5 W. Ans.** $[\because$ Nm / s = W (watt)$]$

Problem 17.21 *A jet of water having a velocity of 15 m/s strikes a curved vane which is moving with a velocity of 5 m/s. The vane is symmetrical and is so shaped that the jet is deflected through 120°. Find the angle of the jet at inlet of the vane so that there is no shock. What is the absolute velocity of the jet at outlet in magnitude and direction and the work done per unit weight of water. Assume the vane to be smooth.*

Solution. Given :

Velocity of jet, $V_1 = 15$ m/s

Velocity of vane, $u_1 = 5$ m/s

As vane is symmetrical. Hence angle $\theta = \phi$

Angle of deflection of the jet $= 120^\circ = 180^\circ - (\theta + \phi)$

$\therefore$ $\theta + \phi = 60^\circ$ or each angle, *i.e.*, $\theta = \phi = 30^\circ$

Let the angle of jet at inlet $= \alpha$

Absolute velocity of jet at outlet $= V_2$

Angle made by V_2 at outlet with direction of motion of vane $= \beta^*$.

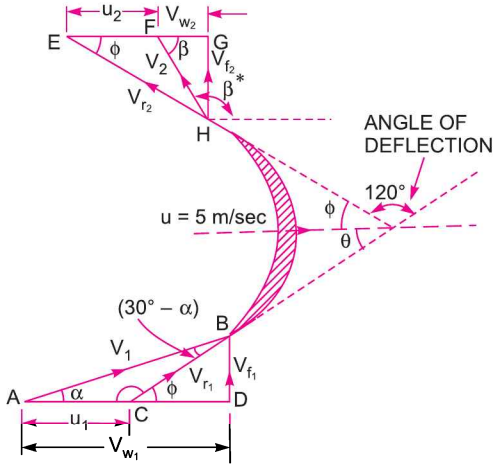


Fig. 17.19

For this problem, $u_1 = u_2 = u = 5$ m/s

$V_{r_1} = V_{r_2}$ (as vane is smooth)

Applying the sine rule to ΔACB,

$$\frac{AB}{\sin(180^\circ - \theta)} = \frac{AC}{\sin(30^\circ - \alpha)} \quad \text{or} \quad \frac{V_1}{\sin(180^\circ - 30^\circ)} = \frac{u_1}{\sin(30^\circ - \alpha)}$$

or $$\frac{15}{\sin 30^\circ} = \frac{5}{\sin(30^\circ - \alpha)} \quad \text{or} \quad \sin(30^\circ - \alpha) = \frac{5 \sin 30^\circ}{15}$$

$$= \frac{1}{3} \times 0.5 = .1667 = \sin 9.596^\circ$$

$\therefore$ $30^\circ - \alpha = 9.596^\circ$ or $\alpha = 30^\circ - 9.596^\circ = 20.404^\circ$ or **20° 24′. Ans.**

Also from sine rule to ΔACB, we have

$$\frac{AB}{\sin(180^\circ - 30^\circ)} = \frac{CB}{\sin \alpha} \quad \text{or} \quad \frac{V_1}{\sin 30^\circ} = \frac{V_{r_1}}{\sin 20.404^\circ}$$

$\therefore$ $$V_{r_1} = \frac{V_1 \sin 20.404^\circ}{\sin 30^\circ} = 10.46 \text{ m/s}$$

$\therefore$ $V_{r_2} = V_{r_1} = 10.46$ m/s

From velocity ΔHEG at outlet,

$$V_{r_2} \cos \phi = u_2 + V_{w_2} \quad \text{or} \quad 10.46 \cos 30^\circ = 5.0 + V_{w_2}$$

$\therefore$ $V_{w_2} = 10.46 \cos 30^\circ - 5.0 = 4.06$ m/s

Also, we have $V_{r_2} \sin \phi = V_{f_2}$ or $V_{f_2} = 10.46 \sin 30^\circ = 5.23$ m/s

In ΔHFG, $$V_2 = \sqrt{V_{f_2}^2 + V_{w_2}^2} = \sqrt{5.23^2 + 4.06^2}$$

$$= \sqrt{27.353 + 16.483} = \mathbf{6.62 \text{ m/s. Ans.}}$$

$$\tan \beta = \frac{V_{f_2}}{V_{w_2}} = \frac{5.23}{4.06} = 1.288 = \tan 52.17^\circ$$

$\therefore$ $\beta = 52.17^\circ$ or $52^\circ\ 10.2'$

$\therefore$ Angle made by absolute velocity at outlet with the direction of motion β^*

$$= 180^\circ - \beta = 180^\circ - (52^\circ\ 10.2') = 127^\circ\ 49.8'$$

$\therefore$ $\beta^* = \mathbf{127^\circ\ 49.8. \text{ Ans.}}$

Work done* per unit weight of the water striking

$$= \frac{1}{g} [V_{w_1} + V_{w_2}] \times u \text{ Nm } (\because \text{ + ve sign taken as } \beta \text{ is an acute angle})$$

$$= \frac{1}{9.81} [V_1 \cos \alpha + 4.06] \times 5 \qquad (\because \quad V_{w_1} = V_1 \cos \alpha)$$

$$= \frac{5}{9.81} [15 \cos 20.404^\circ + 4.06] = \mathbf{9.225 \text{ Nm/N. Ans.}}$$

---

* Work done per unit weight of water striking is the same as work done per second per unit weight of water striking per second refer to equation (17.21).

Problem 17.22 *A jet of water moving at 12 m/s impinges on vane shaped to deflect the jet through 120° when stationary. If the vane is moving at 5 m/s, find the angle of the jet so that there is no shock at inlet. What is the absolute velocity of the jet at exit in magnitude and direction and the work done per second per unit weight of water striking per second? Assume that the vane is smooth.*

Solution. Given :

Velocity of jet, $V_1 = 12$ m/s

Velocity of vane, $u = u_1 = u_2 = 5$ m/s

Angle of deflection of jet $= 120°$

$\therefore \quad \theta + \phi = 180° - 120° = 60°.$

It is not given that the vane is symmetrical and without this condition problem cannot be solved. Assuming vane to be symmetrical, we have $\theta = \phi$

Then $\theta = \phi = 30°$

(*i*) Angle of jet at inlet with the direction of motion of vane $= \alpha$

In ΔABC, applying sine rule, we have

$$\frac{AB}{\sin(180^\circ - \theta)} = \frac{AC}{\sin(30^\circ - \alpha)} \text{ or } \frac{V_1}{\sin\theta} = \frac{u_1}{\sin(30^\circ - \alpha)} \text{ or } \frac{12}{\sin 30^\circ} = \frac{5}{\sin(30^\circ - \alpha)}$$

$$\therefore \quad \sin(30° - \alpha) = \frac{5 \sin 30^\circ}{12} = 0.2083 = \sin 12.02°$$

$\therefore \quad 30° - \alpha = 12.02°$ or $\alpha = 30° - 12.02° = 17.98°$ or **17° 59′. Ans.**

Again applying sine rule to ΔABC, we have

$$\frac{V_1}{\sin(180^\circ - \theta)} = \frac{V_{r_1}}{\sin\alpha} \text{ or } \frac{12}{\sin\theta} = \frac{V_{r_1}}{\sin 17.98^\circ}$$

$$\therefore \quad V_{r_1} = \frac{12 \sin 17.98^\circ}{\sin\theta} = \frac{12 \times \sin 17.98^\circ}{\sin 30^\circ} = 7.41 \text{ m/s}$$

In ΔABD, $V_{w_1} = V_1 \times \cos\alpha = 12 \cos 17.98° = 11.41$ m/s

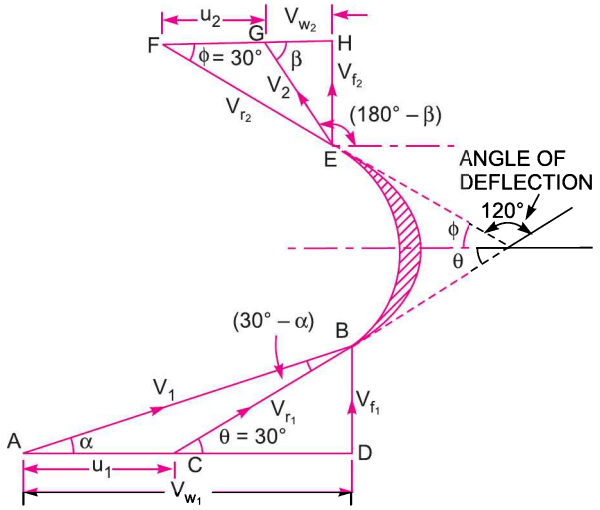


Fig. 17.20

(*ii*) The absolute velocity of jet at outlet = V_2

The angle made by V_2 at outlet with the direction of the motion of vane = $180° - \beta$

Now as vane is given smooth, $V_{r_2} = V_{r_1} = 7.41$ m/s

At outlet, from ΔEFH, we have $V_{r_2} \cos \phi = u_2 + V_{w_2}$ or $7.41 \cos 30° = 5 + V_{w_2}$

$\therefore \quad V_{w_2} = 7.41 \cos 30° - 5.0 = 1.417$ m/s

Also $V_{f_2} = V_{r_2} \sin 30° = 7.41 \sin 30° = 3.705$ m/s

And $$\tan \beta = \frac{V_{f_2}}{V_{w_2}} = \frac{3.705}{1.417} = 2.614 = \tan 69.07°$$

$\therefore \quad \beta = 69.07°$ or $69° 4.2'$

$\therefore$ Angle made by V_2 at outlet with the direction of motion of vane

$= 180° - \beta = 180° - (69° 4.2') =$ **110° 55.8′. Ans.**

Also $$V_2 = \sqrt{V_{f_2}^2 + V_{w_2}^2} = \sqrt{(3.705)^2 + (1.417)^2} = \sqrt{13.727 + 2.007}$$

=3.96 m/s. Ans.

(*iii*) Work done per second per unit weight of water striking per second

$$= \frac{1}{g} [V_{w_1} + V_{w_2}] u = \frac{1}{9.81} [11.41 + 1.417] \times 5 = \textbf{6.537 Nm/N. Ans.}$$

Problem 17.23 *A jet of water having a velocity of 15 m/s, strikes a curved vane which is moving with a velocity of 5 m/s in the same direction as that of the jet at inlet. The vane is so shaped that the jet is deflected through 135°. The diameter of jet is 100 mm. Assuming the vane to be smooth , find :*

(i) Force exerted by the jet on the vane in the direction of motion,

(ii) Power exerted on the vane, and

(iii) Efficiency of the vane.

Solution. Given :

Velocity of jet, $V_1 = 15$ m/s

Velocity of vane, $u = u_1 = u_2 = 5$ m/s

At inlet jet and vane are in the same direction, hence $\alpha = 0$

Diameter of jet, $d = 100$ mm $= 0.1$ m

$\therefore$ Area, $$a = \frac{\pi}{4} (.1)^2 = .007854 \text{ m}^2$$

Angle of deflection of the jet $= 135° = 180° - \phi$ ($\because \theta = 0°$)

$\therefore \quad \phi = 180° - 135° = 45°$

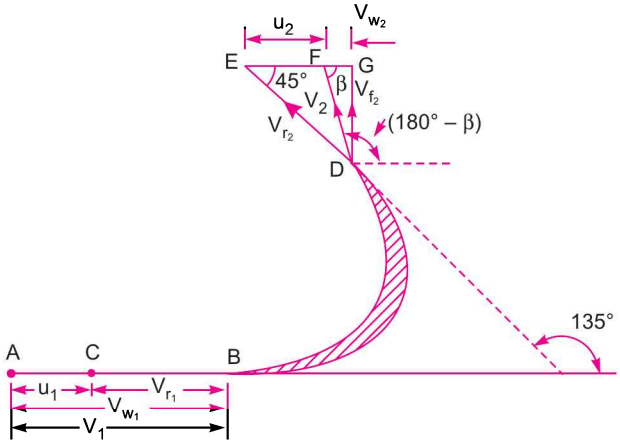


Fig. 17.21

As vane is given smooth hence $V_{r_1} = V_{r_2}$

From the inlet velocity triangle, which is a straight line in this case, we have

$$V_{r_1} = V_1 - u_1 = 15 - 5 = 10 \text{ m/s}$$

$$V_{w_1} = V_1 = 15 \text{ m/s}$$

From the outlet velocity triangle DEG, we have

$$V_{r_2} = V_{r_1} = 10 \text{ m/s}$$

$$u_2 = u_1 = u = 5 \text{ m/s}$$

$$V_{r_2} \cos \phi = u_2 + V_{w_2} \quad \text{or} \quad 10 \cos 45° = 5 + V_{w_2}$$

$$\therefore \quad V_{w_2} = 10 \cos 45° - 5 = 7.07 - 5 = 2.07 \text{ m/s}.$$

(*i*) Force exerted by the jet on the vane in the direction of motion is given by equation (17.19) as

$$F_x = \rho a V_{r_1} [V_{w_1} + V_{w_2}] \text{ (+ve sign is taken as } \beta \text{ is an acute angle)}$$

$$= 1000 \times .007854 \times 10[15 + 2.07] = \mathbf{1340.6 \text{ N. Ans.}}$$

(*ii*) Power of the vane is given as

$$= F_x \times u \text{ N m/s} = 1340.6 \times 5 = 6703 \text{ W} = \mathbf{6.703 \text{ kW. Ans.}}$$

(*iii*) Efficiency of the vane $= \dfrac{\text{Work done per second on vane}}{\text{Kinetic energy supplied by jet per second}}$

$$= \frac{F_x \times u}{\frac{1}{2} \times (\text{mass per second}) \times V^2} = \frac{F_x \times u}{\frac{1}{2} (\rho a V_1) \times V_1^2}$$

$$= \frac{1340.6 \times 5.0}{\frac{1}{2} \times (1000 \times .007854 \times 15) \times 15^2} = \mathbf{0.505 = 50.5\%. \text{ Ans.}}$$

17.4.5 Force Exerted by a Jet of Water on a Series of Vanes. The force exerted by a jet of water on a *single* moving plate (which may be flat or curved) is not practically feasible. This case is only a theoretical one. In actual practice, a large number of plates are mounted on the circumference of a wheel at a fixed distance apart as shown in Fig. 17.22. The jet strikes a plate and due to the force exerted by the jet on the plate, the wheel starts moving and the 2nd plate mounted on the wheel appears before the jet, which again exerts the force on the 2nd plate. Thus each plate appears successively before the jet and the jet exerts force on each plate. The wheel starts moving at a constant speed.

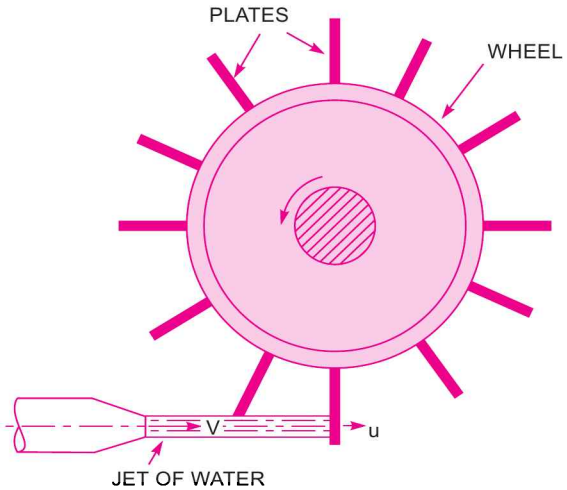


Fig. 17.22 *Jet striking a series of vanes.*

Let V = Velocity of jet,

d = Diameter of jet,

a = Cross-sectional area of jet,

$= \frac{\pi}{4} d^2$

u = Velocity of vane.

In this case the mass of water coming out from the nozzle per second is always in contact with the plates, when all the plates are considered. Hence mass of water per second striking the series of plates = ρaV.

Also the jet strikes the plate with a velocity = $(V - u)$.

After striking, the jet moves tangential to the plate and hence the velocity component in the direction of motion of plate is equal to zero.

$\therefore$ The force exerted by the jet in the direction of motion of plate,

F_x = Mass per second [Initial velocity – Final velocity]

$= \rho aV[(V - u) - 0] = \rho aV[V - u]$ ...(17.22)

Work done by the jet on the series of plates per second

= Force × Distance per second in the direction of force

$= F_x \times u = \rho aV[V - u] \times u$

Kinetic energy of the jet per second

$$= \frac{1}{2} mV^2 = \frac{1}{2} (\rho aV) \times V^2 = \frac{1}{2} \rho aV^3$$

$\therefore$ Efficiency, $$\eta = \frac{\text{Work done per second}}{\text{Kinetic energy per second}} = \frac{\rho aV[V - u] \times u}{\frac{1}{2}\rho aV^3} = \frac{2u[V - u]}{V^2} \quad ...(17.23)$$

Condition for Maximum Efficiency. Equation (17.23) gives the value of the efficiency of the wheel. For a given jet velocity V, the efficiency will be maximum when

$$\frac{d\eta}{du} = 0 \quad \text{or} \quad \frac{d}{du}\left[\frac{2u(V - u)}{V^2}\right] = 0 \quad \text{or} \quad \frac{d}{du}\left[\frac{2uV - 2u^2}{V^2}\right] = 0$$

or $$\frac{2V - 2 \times 2u}{V^2} = 0 \quad \text{or} \quad 2V - 4u = 0 \quad \text{or} \quad V = \frac{4u}{2} = 2u \text{ or } u = \frac{V}{2}. \quad ...(17.24)$$

Maximum Efficiency. Substituting the value of $V = 2u$ in equation (17.23), we get the maximum efficiency as

$$\eta_{max} = \frac{2u\,[2u - u]}{(2u)^2} = \frac{2u \times u}{2u \times 2u} = \frac{1}{2} = 0.5 \text{ or } 50\%. \quad ...(17.25)$$

17.4.6 Force Exerted on a Series of Radial Curved Vanes. For a radial curved vane, the radius of the vane at inlet and outlet is different and hence the tangential velocities of the radial vane at inlet and outlet will not be equal. Consider a series of radial curved vanes mounted on a wheel as shown in Fig. 17.23. The jet of water strikes the vanes and the wheel starts rotating at a constant angular speed.

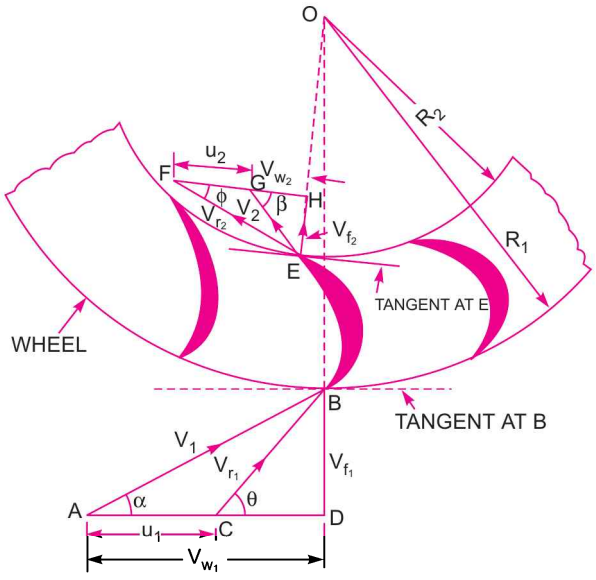


Fig. 17.23 *Series of radial curved vanes mounted on a wheel.*

Let R_1 = Radius of wheel at inlet of the vane,

R_2 = Radius of the wheel at the outlet of the vane,

ω = Angular speed of the wheel.

Then $u_1 = \omega R_1$ and $u_2 = \omega R_2$

The velocity triangles at inlet and outlet are drawn as shown in Fig. 17.23.

The mass of water striking per second for a series of vanes

= Mass of water coming out from nozzle per second

= $\rho a V_1$, where a = Area of jet and V_1 = Velocity of jet.

Momentum of water striking the vanes in the tangential direction per sec at inlet

= Mass of water per second × Component of V_1 in the tangential direction

= $\rho a V_1 \times V_{w_1}$ ($\because$ Component of V_1 in tangential direction = $V_1 \cos \alpha = V_{w_1}$)

Similarly, momentum of water at outlet per sec

= $\rho a V_1$ × Component of V_2 in the tangential direction

= $\rho a V_1 \times (-V_2 \cos \beta) = -\rho a V_1 \times V_{w_2}$ ($\because$ $V_2 \cos \beta = V_{w_2}$)

–ve sign is taken as the velocity V_2 at outlet is in opposite direction.

Now, angular momentum per second at inlet

= Momentum at inlet × Radius at inlet

= $\rho a V_1 \times V_{w_1} \times R_1$

Angular momentum per second at outlet

= Momentum of outlet × Radius at outlet

= $-\rho a V_1 \times V_{w_2} \times R_2$

Torque exerted by the water on the wheel,

T = Rate of change of angular momentum

= [Initial angular momentum per second – Final angular momentum per second]

$= \rho a V_1 \times V_{w_1} \times R_1 - (-\rho a V_1 \times V_{w_2} \times R_2) = \rho a V_1 [V_{w_1} \times R_1 + V_{w_2} R_2]$

Work done per second on the wheel

= Torque × Angular velocity = $T \times \omega$

$= \rho a V_1 [V_{w_1} \times R_1 + V_{w_2} R_2] \times \omega = \rho a V_1 [V_{w_1} \times R_1 \times \omega + V_{w_2} R_2 \times \omega]$

$= \rho a V_1 [V_{w_1} u_1 + V_{w_2} \times u_2]$ $(\because \quad u_1 = \omega R_1$ and $u_2 = \omega R_2)$

If the angle β in Fig. 17.23 is an obtuse angle then work done per second will be given as

$= \rho a V_1 [V_{w_1} u_1 - V_{w_2} u_2]$

$\therefore$ The general expression for the work done per second on the wheel

$= \rho a V_1 [V_{w_1} u_1 \pm V_{w_2} u_2]$ ...(17.26)

If the discharge is radial at outlet, then $\beta = 90°$ and work done becomes as

$= \rho a V_1 [V_{w_1} u_1]$ $(\because \quad V_{w_2} = 0)$ ...(17.27)

Efficiency of the Radial Curved Vane

The work done per second on the wheel is the output of the system whereas the initial kinetic energy per second of the jet is the input. Hence, efficiency of the system is expressed as

$$\text{Efficiency, } \eta = \frac{\text{Work done per second}}{\text{Kinetic energy per second}} = \frac{\rho a V_1 \left[V_{w_1} u_1 \pm V_{w_2} u_2\right]}{\frac{1}{2}(\text{mass/sec}) \times V_1^2}$$

$$= \frac{\rho a V_1 \left[V_{w_1} u_1 \pm V_{w_2} u_2\right]}{\frac{1}{2}(\rho a V_1) \times V_1^2} = \frac{2\left[V_{w_1} u_1 \pm V_{w_2} u_2\right]}{V_1^2}. \qquad \text{...(17.28)}$$

If there is no loss of energy when water is flowing over the vanes, the work done on the wheel per second is also equal to the change in kinetic energy of the jet per second. Hence, the work done per second on the wheel is also given as

Work done per second on the wheel

= Change of K.E. per second of the jet

= (Initial K.E. per second – Final K.E. per second) of the jet

$$= \left(\frac{1}{2} m V_1^2 - \frac{1}{2} m V_2^2\right)$$

$$= \frac{1}{2} m \left(V_1^2 - V_2^2\right) = \frac{1}{2}\left(\rho a V_1^2\right)\left(V_1^2 - V_2^2\right) \qquad (\because \text{mass/second} = \rho a V_1)$$

$$\text{Hence efficiency, } \eta = \frac{\text{Work done per second on the wheel}}{\text{Initial K.E. per second of the jet}}$$

$$= \frac{\frac{1}{2} \rho a V_1^2 \left(V_1^2 - V_2^2\right)}{\frac{1}{2}\left(\rho a V_1^2\right) \cdot V_1^2}$$

$$= \frac{V_1^2 - V_2^2}{V_1^2} = \left(1 - \frac{V_2^2}{V_1^2}\right) \qquad ...(17.28A)$$

From the above equation, it is clear that for a given initial velocity of the jet (*i.e.*, V_1), the efficiency will be maximum, when V_2 is minimum. But V_2 cannot be zero as in that case the incoming jet will not move out of the vane. Equation (17.28) also gives the efficiency of the system. From this equation, it is clear that efficiency will be maximum when V_{w_2} is added to V_{w_1}. This is only posible if β is an acute* angle. Also for maximum efficiency V_{w_2} should also be maximum. This is only possible if $\beta = 0$. In that case $V_{w_2} = V_2$ and angle ϕ will be zero. But in actual practice ϕ cannot be zero. Hence for maximum efficiency, the angle ϕ should be minimum.

Problem 17.24 *If in Problem 17.23, the jet of water instead of striking a single plate, strikes a series of curved vanes, find for the data given Problem 17.23,*

(i) Force exerted by the jet on the vane in the direction of motion,
(ii) Power exerted on the vane, and
(iii) Efficiency of the vane.

Solution. Given :

From Problem 17.23, $V_1 = 15$ m/s, $u = u_1 = u_2 = 5$m/s,
$\alpha = 0$, $a = .007854$ m^2
$\phi = 45°$, $V_{w_1} = 15$ m/s and $V_{w_2} = 2.07$ m/s.

For the series of vanes, mass of water striking per second
= Mass of water coming out from nozzle
$= \rho a V_1 = 1000 \times .007854 \times 15 = 117.72$

(*i*) Force exerted by the jet on the vane in the direction of motion

$$F_x = \rho a V_1 [V_{w_1} + V_{w_2}] = 117.72 [15 + 2.07] = \textbf{2009.5 N. Ans.}$$

(*ii*) Power of the vane in kW

$$= \frac{\text{Work done per second}}{1000} = \frac{F_x \times u}{1000}\text{kW} = \frac{2009.5 \times 5}{1000}$$
$$= \textbf{10.05 kW. Ans.}$$

(*iii*) Efficiency,
$$\eta = \frac{\text{Work done per second}}{\frac{1}{2}(\text{mass of water per sec}) \times V_1^2}$$
$$= \frac{2009.5 \times 5.0}{\frac{1}{2} \times 117.72 \times 15^2} = 0.7586 \text{ or } \textbf{75.86\%}. \quad \textbf{Ans.}$$

Problem 17.25 *A jet of water having a velocity of 35 m/s impinges on a series of vanes moving with a velocity of 20 m/s. The jet makes an angle of 30° to the direction of motion of vanes when entering and leaves at an angle of 120°. Draw the triangles of velocities at inlet and outlet and find :*

(a) the angles of vanes tips so that water enters and leaves without shock,
(b) the work done per unit weight of water entering the vanes, and
(c) the efficiency.

Solution. Given :

Velocity of jet, $V_1 = 35$ m/s
Velocity of vane, $u_1 = u_2 = 20$ m/s
Angle of jet at inlet, $\alpha = 30°$

---

* The work done is equal to torque multiplied by ω(angular velocity). Torque is the rate of change of angular momentum. Due to change of angular momentum (*i.e.*, initial angular momentum – final angular momentum), V_{w_2} should be in opposite direction so that it can be added to V_{w_1}. This is possible if $\beta \angle 90°$.

Angle made by the jet at outlet with the direction of motion of vanes = 120°

$\therefore$ Angle $\beta = 180° - 120° = 60°$

(*a*) *Angle of vanes tips.*

From inlet velocity triangle

$$V_{w_1} = V_1 \cos \alpha = 35 \cos 30° = 30.31 \text{ m/s}$$

$$V_{f_1} = V_1 \sin \alpha = 35 \sin 30° = 17.50 \text{ m/s}$$

$$\tan \theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{17.50}{30.31 - 20} = 1.697$$

$\therefore$ $\theta = \tan^{-1} 1.697 = \mathbf{60°}$. **Ans.**

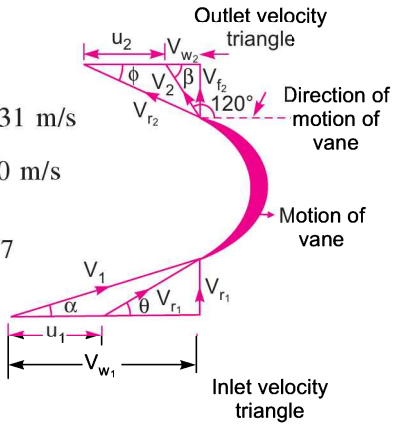


Fig. 17.23(*a*)

By sine rule, $\dfrac{V_{r_1}}{\sin 90°} = \dfrac{V_{f_1}}{\sin \theta}$ or $\dfrac{V_{r_1}}{1} = \dfrac{17.50}{\sin 60°}$

$\therefore$ $V_{r_1} = \dfrac{17.50}{.866} = 20.25$ m/s.

Now, $V_{r_2} = V_{r_1} = 20.25$ m/s

From outlet velocity triangle, by sine rule

$$\frac{V_{r_2}}{\sin 120°} = \frac{u_2}{\sin (60° - \phi)} \quad \text{or} \quad \frac{20.25}{0.886} = \frac{20}{\sin (60° - \phi)}$$

$\therefore$ $\sin (60° - \phi) = \dfrac{20 \times 0.866}{20.25} = 0.855 = \sin (58.75°)$

$60° - \phi = 58.75°$

$\therefore$ $\phi = 60° - 58.75° = \mathbf{1.25°}$. **Ans.**

(*b*) *Work done per unit weight of water entering* $= \dfrac{1}{g}(V_{w_1} + V_{w_2}) \times u_1$ ...(*i*)

$V_{w_1} = 30.31$ m/s and $u_1 = 30$ m/s

The value of V_{w_2} is obtained from outlet velocity triangle

$$V_{w_2} = V_{r_2} \cos \phi - u_2 = 20.25 \cos 1.25° - 20.0 = 0.24 \text{ m/s}$$

$\therefore$ Work done/unit weight $= \dfrac{1}{9.81} [30.31 + 0.24] \times 20 = \mathbf{62.28 \text{ Nm/N}}$. **Ans.**

(*c*) *Efficiency* $= \dfrac{\text{Work done per kg}}{\text{Energy supplied per kg}}$

$$= \frac{62.28}{\frac{V_1^2}{2g}} = \frac{62.28 \times 2 \times 9.81}{35 \times 35} = \mathbf{99.74\%} \text{ Ans.}$$

Problem 17.26 *A jet of water having a velocity of 30 m/s strikes a series of radial curved vanes mounted on a wheel which is rotating at 200 r.p.m. The jet makes an angle of 20° with the tangent to the wheel at inlet and leaves the wheel with a velocity of 5 m/s at an angle of 130° to the tangent to the wheel at outlet. Water is flowing from outward in a radial direction. The outer and inner radii of the wheel are 0.5 m and 0.25 m respectively. Determine :*

(i) Vane angles at inlet and outlet, (ii) Work done per unit weight of water, and

(iii) Efficiency of the wheel.

Solution. Given :

Velocity of jet, $V_1 = 30$ m/s

Speed of wheel, $N = 200$ r.p.m.

$\therefore$ Angular speed, $\omega = \frac{2\pi N}{60} = \frac{2\pi \times 200}{60} = 20.94$ rad/s

Angle of jet at inlet, $\alpha = 20°$

Velocity of jet at outlet, $V_2 = 5$ m/s

Angle made by the jet at outlet with the tangent to wheel = 130°

$\therefore$ Angle, $\beta = 180° - 130° = 50°$

Outer radius, $R_1 = 0.5$ m

Inner radius, $R_2 = 0.25$ m

$\therefore$ Velocity $u_1 = \omega \times R_1 = 20.94 \times 0.5 = 10.47$ m/s

And $u_2 = \omega \times R_2 = 20.94 \times 0.25 = 5.235$ m/s.

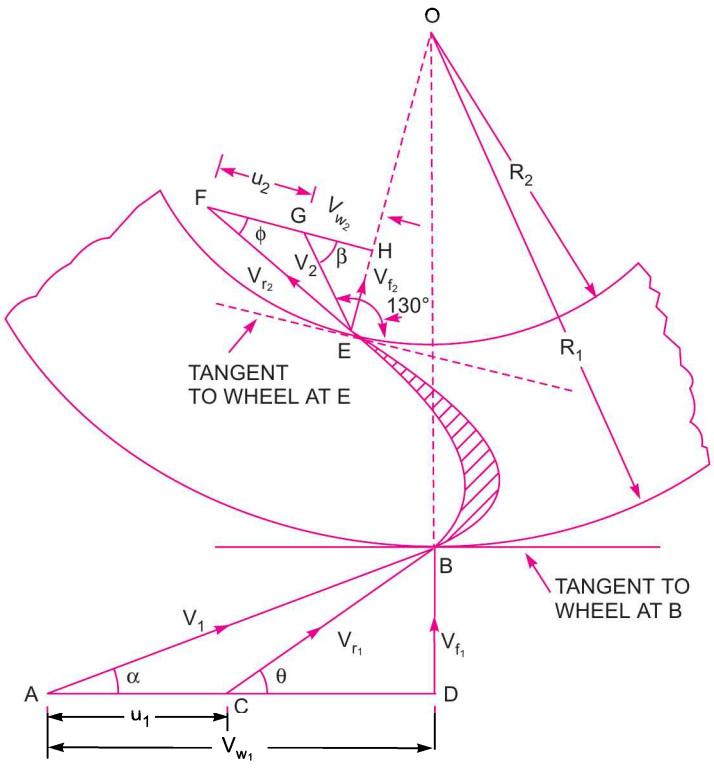


Fig. 17.24

(*i*) **Vane angles** at inlet and outlet means the angle made by the relative velocities V_{r_1} and V_{r_2}, *i.e.*, angle θ and ϕ.

From ΔABD, $V_{w_1} = V_1 \cos \alpha = 30 \times \cos 20° = 28.19$ m/s

$$V_{f_1} = V_1 \sin \alpha = 30 \times \sin 20° = 10.26 \text{ m/s}$$

In ΔCBD, $$\tan \theta = \frac{BD}{CD} = \frac{V_{f_1}}{AD - AC} = \frac{10.26}{V_{w_1} - u_1} = \frac{10.26}{28.19 - 10.47} = 0.579 = \tan 30.07$$

$\therefore$ $\theta = 30.07°$ or **30° 4.2′. Ans.**

From outlet velocity Δ, $V_{w_2} = V_2 \cos \beta = 5 \times \cos 50° = 3.214$ m/s

$$V_{f_2} = V_2 \times \sin \beta = 5 \sin 50° = 3.83 \text{ m/s}$$

In ΔEFH, $$\tan \phi = \frac{V_{f_2}}{u_2 + V_{w_2}} = \frac{3.83}{5.235 + 3.214} = 0.453 = \tan 24.385°$$

$\therefore$ $\phi = 24.385°$ or **24° 23.1′. Ans.**

(*ii*) Work done per second by water is given by equation (17.26)

$$= \rho a V_1 \left[V_{w_1} u_1 + V_{w_2} u_2\right]$$

(+ ve sign is taken as β is acute angle in Fig.17.24)

$\therefore$ Work done* per second per unit weight of water striking per second

$$= \frac{\rho a V_1 \left[V_{w_1} u_1 + V_{w_2} u_2\right]}{\text{Weight of water/s}} = \frac{\rho a V_1 \left[V_{w_1} u_1 + V_{w_2} u_2\right]}{\rho a V_1 \times g}$$

$$= \frac{1}{g}\left[V_{w_1} u_1 + V_{w_2} u_2\right] \text{Nm/N} = \frac{1}{9.81} [28.19 \times 10.47 + 3.214 \times 5.235]$$

$$= \frac{1}{9.81} [295.15 + 16.82] = \mathbf{31.8\ Nm/N.\ Ans.}$$

(*iii*) Efficiency, η is given by equation (17.28) as

$$\eta = \frac{2\left[V_{w_1} u_1 + V_{w_2} u_2\right]}{V_1^2} = \frac{2[28.19 \times 10.47 + 3.214 \times 5.235]}{30^2}$$

$$= \frac{2[295.15 + 16.82]}{30 \times 30} = 0.6932 \text{ or } \mathbf{69.32\%.\ Ans.}$$

▶ 17.5 JET PROPULSION

Jet propulsion means the propulsion or movement of the bodies such as ships, aircrafts, rocket etc., with the help of jet. The reaction of the jet coming out from the orifice provided in the bodies is used to move the bodies. This is explained as given below.

* Work done per second per unit weight striking per second is same as work done per unit weight of water.

A jet of fluid coming out from an orifice or nozzle, when strikes a plate, exerts a force on the plate. The magnitude of the force exerted on the plate can be determined depending upon whether plate is flat, inclined, curved, stationary or moving. This force exerted by the jet on the plate is called as '*action of the jet*'. But according to Newtons third law of motion, every action is accompanied by an equal and opposite reaction. Hence the jet while coming out of the orifice or nozzle, exerts a force on the orifice or nozzle in the opposite direction in which jet is coming out. The magnitude of the force exerted is equal to the 'action of the jet'. This force which is acting on the orifice or nozzle in the opposite direction is called the 'reaction of the jet'. If the body in which orifice or nozzle is fitted, is free to move, the body will start moving in the direction opposite to the jet. The following cases are important where this principle is used :

(*a*) Jet propulsion of a tank to which orifice is fitted, and

(*b*) Jet propulsion of ships.

17.5.1 Jet Propulsion of a Tank with an Orifice. Consider a large tank fitted with an orifice in one of its sides as shown in Fig. 17.25.

Let H = Constant head of water in tank from the centre of orifice,

a = Area of orifice,

V = Velocity of the jet of water,

C_v = Co-efficient of the velocity of orifice.

Then $V = C_v \sqrt{2gH}$

And mass of water coming out from the orifice per second

= ρ × Volume per second = ρ × (Area × Velocity)

= ρ × a × V

Force acting on the water is equal to the rate of change of momentum.

or F = Mass per second × [Change of velocity]

= Mass per second × [Final velocity – Initial velocity].

Note. Here change of velocity is to be taken as final minus initial as we are finding force on water and not force exerted by water.

Initial velocity of water in the tank is zero and final velocity of water when it comes out in the form of jet is equal to V.

$\therefore$ $F = \rho a V\,[V - 0] = \rho a V^2$ ...(17.29)

Thus, F is the force exerted on the jet of water. This jet of water will exert a force on the tank which is equal to F but opposite in direction as shown in Fig. 17.25. The force will be acting at A, the point on the tank in the horizontal line of the centre of the orifice. If the tank is free to move or the tank is fitted with frictionless wheel, it will start moving with some velocity say, 'u' in the direction opposite to the direction of the jet. When the tank starts moving, the velocity of the jet with which it comes out of the orifice will not be equal to V but it will be equal to the relative velocity of the jet with respect to tank.

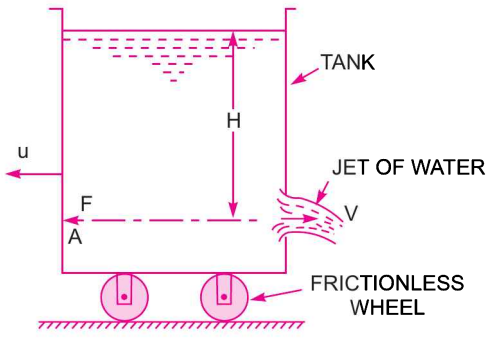


Fig. 17.25 *Jet propulsion of a tank with an orifice.*

Hence if V = Absolute velocity of jet,

u = Velocity of tank,

V_r = Velocity of jet with respect to tank

Then V_r = Vectorial difference of absolute velocity (V) and velocity of tank (u)
$= V - (-u)$(as u is in opposite direction to V hence velocity of tank is taken as $-u$)
$= V + u$

Hence when the tank is moving, the velocity with which jet comes out from the orifice is $(V + u)$.

Mass of water coming out from the orifice per sec
$= \rho \times a \times$ Velocity with which water comes out
$= \rho \times a \times (V_r) = \rho a\,(V + u)$

$\therefore$ Force exerted on the tank is given as
F_x = Mass of water coming out from orifice per second $\times$ [Change of velocity]*
$= \rho a(V + u) \times [(V + u) - u] = \rho a\,[V + u]\,[V]$
$= \rho a[V + u] \times V$ ...(17.30)

Thus, the force given by equation (17.30) is used for propelling the tank.

$\therefore$ Work done on the moving tank by jet per second
$= F_x \times u = \rho a(V + u) \times V \times u$

$\therefore$ Efficiency of propulsion is given as,

$$\eta = \frac{\text{Work done per second}}{\text{Kinetic energy of the issuing jet per second}}$$

$$= \frac{\rho a(V+u)\times V \times u}{\frac{1}{2}(\text{Mass of water issuing per second})\times(\text{Velocity of issuing jet})}$$

$$= \frac{\rho a(V+u)\times V \times u}{\frac{1}{2}\left[\rho a(V+u)\right]\times(V+u)^2} = \frac{2Vu}{(V+u)^2} \quad \text{...(17.31)}$$

Condition for Maximum Efficiency and Expression for Maximum η. For a given value of V, the efficiency will be maximum when $\frac{d\eta}{du} = 0$

or $$\frac{d}{du}\left[\frac{2Vu}{(V+u)^2}\right] = 0 \quad \text{or} \quad \frac{d}{du}\left[2Vu\times(V+u)^{-2}\right] = 0$$

or $$2Vu \times (-2)\,(V+u)^{-3} + (V+u)^{-2} \times 2V = 0$$

or $$\frac{-4Vu}{(V+u)^3} + \frac{2V}{(V+u)^2} = 0 \quad \text{or} \quad -4Vu + 2V(V+u) = 0$$

Dividing by $2V$, $-2u + (V + u) = 0$ or $-u + V = 0$ or $u = V$ ...(17.32)

Equation (17.32) is the condition for maximum efficiency. Substituting equation (17.32) in equation (17.31), the value of maximum efficiency is obtained as

$$\eta_{max} = \frac{2\times u\times u}{(u+u)^2} = \frac{2u^2}{4u^2} = \frac{1}{2} = 0.5 \text{ or } 50\%. \quad \text{...(17.32A)}$$

---

* Change of velocity is the final velocity minus initial velocity of jet of water coming out from the orifice. The final velocity of the jet with respect to tank is $(V + u)$. This velocity is obtained by applying a velocity u to the whole system (*i.e.*, tank, water in the tank and jet of water) in a direction opposite to the motion of tank. Then final velocity of jet becomes as $(V + u)$ and initial velocity of water as u. Hence change of velocity is $(V + u) - u$.

Problem 17.27 *The head of water from the centre of the orifice which is fitted to one side of the tank is maintained at 2 m of water. The tank is not allowed to move and the diameter of orifice is 100 mm. Find the force exerted by the jet of water on the tank. Take C_v = 0.97 cm.*

Solution. Given :

Head of water, $H = 2$ m

Diameter of orifice, $d = 100$ mm $= 0.1$ m

$\therefore$ Area, $a = \frac{\pi}{4}d^2 = \frac{\pi}{4}(.1)^2 = .007854$ m^2

Value of $C_v = 0.97$

$\therefore$ Velocity of jet, $V = C_v \times \sqrt{2gH} = 0.97 \times \sqrt{2 \times 9.81 \times 2.0} = 6.07$ m/s

Force exerted on the tank is given by equation (17.29) as

$$F = \rho a V^2 = 1000 \times .007854 \times 6.072 = \textbf{289.3 N. Ans.}$$

Problem 17.28 *If in the above problem, the tank is fitted with frictionless wheels and allowed to move, determine*

(i) Propelling force on tank, *(ii) Work done by the propelling force per second, and*

(iii) Efficiency of propulsion.

The tank is moving with a velocity of 2 m/s.

Solution. Given :

$H = 2$ m, $d = 100$ mm, $a = .007854$ m^2, $C_v = 0.97$

and velocity of jet, $V = 6.07$ m/s.

Velocity of tank, $u = 2$ m/s.

(*i*) Propelling force is given by equation (17.30) as

$$F_x = \rho a(V + u) \times V$$
$$= 1000 \times .007854 \times (6.07 + 2.0) \times 6.07 = \textbf{384.65 N. Ans.}$$

(*ii*) Work done by the propelling force per second

$$= F_x \times u = 384.65 \times 2.0 = \textbf{769.3 N m/s. Ans.}$$

(*iii*) Efficiency of propulsion is given by equation (17.31) as

$$\eta = \frac{2Vu}{(V+u)^2} = \frac{2 \times 6.07 \times 2.0}{(6.07+2.0)^2} = \textbf{0.3728 or 37.28\%. Ans.}$$

17.5.2 Jet Propulsion of Ships. By the application of the jet propulsion principle, a ship is driven through water. A jet of water which is discharged at the back (also called stern) of the ship, exerts a propulsive force on the ship. The ship carries centrifugal pumps which draw water from the surrounding sea. This water is discharged through the orifice provided at the back of the ship in the form of a jet. The reaction of the jet coming out at the back of the ship propels the ship in the opposite direction of the jet. The water from the surrounding sea by the centrifugal pump is taken by the following two ways :

1. Through inlet orifices which are at right angles to the direction of the motion of the ship, and
2. Through the inlet orifices, which are facing the direction of motion of the ship.

1st Case. Jet propulsion of the ship when the inlet orifices are at right angles to the direction of the motion of the ship.

Fig. 17.26 shows a ship which is having the inlet orifices at right angles to its direction.

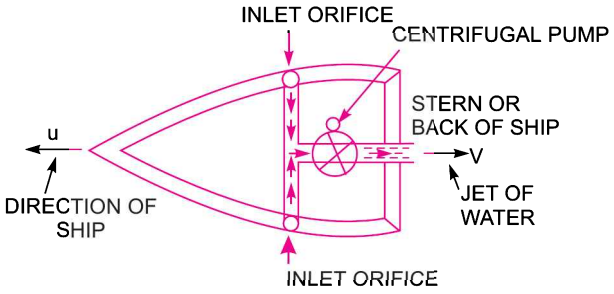


Fig. 17.26 *Inlet orifices are at right angles.*

Let V = Absolute velocity of jet of water coming at the back of the ship,
u = Velocity of the ship,
V_r = Relative velocity of jet with respect to ship
= $(V + u)$.

As the velocity V and u are in opposite direction and hence relative velocity will be equal to the sum of these two velocities.

Mass of water issuing from the orifice at the back of the ship = $\rho a V_r = \rho a(V + u)$,
where a = Area of the jet of water

$\therefore$ Propulsive force exerted on the ship

F = Mass of water issuing per sec × Change of velocity*

$= \rho a(V + u)\,[V_r - u] = \rho a(V + u)\,[(V + u) - u] = \rho a(V + u) \times V$ ...(17.33)

Work done per second $= F \times u = \rho a\,(V + u) \times V \times u$ ...(17.34)

The efficiency of propulsion, the condition of maximum efficiency and expression for maximum efficiency are given by equations (17.31), (17.32) and (17.32 *A*) respectively.

Note. (*i*) When the inlet orifices are at right angles to the direction of motion of the ship, then this case is also known as water is drawn **AMID SHIP** which means the water is drawn at the middle of the ship.

(*ii*) The centrifugal pump draws the water from the surrounding sea and discharges through orifice. The kinetic energy of the issuing jet is $\frac{1}{2}$ × mass × velocity ^{2} *i.e.*, $\frac{1}{2}\,[\rho a(V + u)] \times [V + u]^2 = \frac{1}{2}\,\rho a(V + u)^3$. This energy is provided by centrifugal pump *i.e.*, work is done by pump to provide this energy.

Problem 17.29 *Find the propelling force acting on a ship which takes water through inlet orifices which are at right angles to the direction of motion of ship, and discharges at the back through orifices having effective areas of 0.04 m^2. The water is flowing at the rate of 1000 litres/s and ship is moving with a velocity of 8 m/s.*

Solution. Given :

Effective areas of orifices, $a = 0.04$ m^2

Discharge of water, $Q = 1000$ litres/s $= 1$ m^3/s

$\therefore$ Velocity of jet relative to water $= \dfrac{Q}{a} = \dfrac{1}{.04} = \dfrac{100}{4} = 25$ m/s or $V_r = 25$ m/s

---

* To find the change of velocity, apply a velocity u to the whole system (*i.e.*, ship, jet of water and surrounding water in the sea) in a direction opposite to the motion of ship. Then final velocity of jet of water becomes as $(V + u)$. And velocity of water in sea becomes as u. Hence change of velocity becomes $(V + u) - u$.

Velocity of ship, $u = 8$ m/s

Now, $V_r = u + V$, where V = Absolute velocity of jet

$\therefore$ $25 = 8 + V$ or $V = 25 - 8 = 17$ m/s

Propelling force is given by equation (17.33) as,

$$F = \rho a\,(V + u) \times V$$
$$= 1000 \times .04 \times (17 + 8) \times 17 = \textbf{16999.94 N. Ans.}$$

Problem 17.30 *The water in a jet propelled boat is drawn amid-ship and discharged at the back with an absolute velocity of 20 m/s. The cross-sectional area of the jet at the back is 0.02 m^2 and the boat is moving in sea water with a speed of 30 km/hour. Determine :*

(i) Propelling force on the boat, (ii) Power required to drive the pump, and

(iii) Efficiency of the jet propulsion.

Solution. Given :

'Water is drawn amid-ship' means water is drawn at the middle of the ship and inlet orifices are at right angles to the motion of ship.

Absolute velocity of jet, $V = 20$ m/s

Area of the jet, $a = 0.02$ m^2

Speed of boat, $u = 30$ km/hr $= \dfrac{30 \times 1000}{60 \times 60} = 8.33$ m/s.

(*i*) Propelling force is given by equation (17.33) as

$$F = \rho a(V + u) \times V$$
$$= 1000 \times .02(20 + 8.33) \times 20 = \textbf{11332 N. Ans.}$$

(*ii*) Power required to drive the pump in kW

$$= \frac{\text{Work done per sec}}{1000} = \frac{F \times u}{1000} = \frac{11332 \times 8.33}{1000} = \textbf{94.395 kW. Ans.}$$

(*iii*) Efficiency of the jet propulsion is given by (17.31) as

$$\eta = \frac{2Vu}{(V+u)^2} = \frac{2 \times 20 \times 8.33}{(20+8.33)^2} = \textbf{0.415 or 41.5\%. Ans.}$$

Problem 17.31 *A small ship is fitted with jets of total area 0.65 m^2. The velocity through the jet is 9 m/s and speed of the ship is 18 km p.h. in sea-water. The efficiencies of the engine and pump are 85% and 65% respectively. If the water is taken amid-ships, determine the propelling force and the overall efficiency, assuming the pipe losses to be 10% of the kinetic energy of the jets.*

Solution. Given :

Total area of jets, $a = 0.65$ m^2

Velocity through the jet relative to ship, $V_r = 9$ m/s

Speed of ship, $u = 18$ km/hour $= \dfrac{18 \times 1000}{60 \times 60}$ m/s $= 5$ m/s

Efficiency of the engine, $\eta_E = 85\% = 0.85$

Efficiency of the pump, $\eta_P = 65\% = 0.65$

Pipe losses, h_f = 10% of kinetic energy of the jet

$$= \frac{10}{100} \times \frac{V_r^2}{2g} = \frac{V_r^2}{20g}$$

Now, $V_r = u + V$, where V = Absolute velocity of jet

$\therefore$ $9 = 5 + V$ or $V = 9 - 5 = 4$ m/s.

(*i*) Propelling force is given by equation (17.33) as

$$F = \rho a\,(V + u) \times V$$

$$= 1000 \times 0.65 \times (4 + 5) \times 4 = \textbf{23400 N. Ans.}$$

(*ii*) Work done by the jets per second

$$= F \times u = 23400 \times 5 = 117000 \text{ Nm/s}$$

Weight of water issuing from the jets per second

$$= g \times \text{Mass of water per second}$$

$$= g \times \rho a V_r = 9.81 \times 1000 \times 0.65 \times 9 = 57388.5 \text{ N/s.}$$

The pump should have the output which will give the jet a relative velocity (V_r) and also overcome the pipe losses.

$\therefore$ Output of the pump per unit weight of water

$$= \text{Kinetic energy of jet + Pipe losses}$$

$$= \frac{V_r^2}{2g} + \frac{V_r^2}{20g} = \frac{V_r^2}{2g}\,(1 + 0.1) = \frac{V_r^2}{2g} \times 1.1$$

Input to the pump per unit weight of water

$$= \frac{\text{Output of pump}}{\text{Efficiency of pump}} = \frac{1.1\,V_r^2}{2g \times 0.65}$$

The input to the pump is equal to the output of the engine. Hence input to the engine per unit weight of water

$$= \frac{1.1\,V_r^2}{2g \times 0.65 \times \text{Efficiency of engine}}$$

$$= \frac{1.1\,V_r^2}{2g \times 0.65 \times 0.85} = \frac{1.1 \times 9^2}{2 \times 9.81 \times 0.65 \times 0.85} = 8.22 \text{ Nm/N}$$

$\therefore$ Total input to the engine = Weight of water $\times$ Input per unit weight of water

$$= 57388.5 \times 8.22 = 471733.5 \text{ Nm}$$

$\therefore$ Overall efficiency, $\eta_o = \dfrac{\text{Work done by jets}}{\text{Total input to engine}} = \dfrac{117000}{471733.5} = 0.248 = \textbf{24.80\%. Ans.}$

Problem 17.32 *A jet propelled boat, moving with a velocity of 5 m/s, draws water amid-ship. The water is discharged through two jets provided at the back of the ship. The diameter of each jet is 150 mm. The total resistance offered to the motion of the boat is 4905 N (500 × 9.81 N). Determine :*

(i) Volume of water drawn by the pump per second, and

(ii) Efficiency of the jet propulsion.

Solution. Given :

Velocity of boat, $u = 5$ m/s

Diameter of each jet, $d = 150$ mm $= 0.15$ m

Area of each jet $= \frac{\pi}{4}(.15)^2 = 0.01767$ m^2

$\therefore$ Total area of the jets, $a = 2 \times .01767 = .03534$ m^2

Total resistance to motion $= 4905$ N $(500 \times 9.81$ N$)$

The propelling force must be equal to the resistance to the motion.

$\therefore$ Propelling force, $F = 4905$ N or $(500 \times 9.81$ N$)$

Propelling force is given by equation (17.33) as

$$F = \rho a\,(V + u)\,V$$

$$500 \times 9.81 = 1000 \times 0.03534 \times (V + 5) \times V$$

or $$500 = \frac{1000}{9.81} \times .03534 \times (V + 5) \times V$$

$$= 3.6\,(V + 5)\,V = 3.6V^2 + 3.6 \times 5V = 3.6\,V^2 + 18V$$

or $$3.6\,V^2 + 18V - 500 = 0$$

The above equation is quadratic and its solution is

$$V = \frac{-18 \pm \sqrt{18^2 + 4 \times 3.6 \times 500}}{2 \times 3.6} = \frac{-18 \pm 86.74}{7.2}$$

$$= \frac{86.74 - 18}{2} = 34.37 \text{ m/s} \qquad \text{[–ve value is not possible]}$$

(*i*) Volume of water drawn by the pump per second is equal to the volume of water discharged through the orifices at the back in the form of jets and this volume

$$= aV_r = a(V + u)$$

$$= .03534 \times (34.37 + 5.0) = \mathbf{1.39\ m^3/s.}\ \textbf{Ans.}$$

(*ii*) Efficiency of the jet propulsion is given by equation (17.31) as

$$\eta = \frac{2Vu}{(V + u)^2} = \frac{2 \times 34.37 \times 5.0}{(34.37 + 5.0)^2} = .2217 \text{ or } \mathbf{22.17\%.}\ \textbf{Ans.}$$

2nd Case. Jet propulsion of ship when the inlet orifices face the direction of motion of the ship. Fig. 17.27 shows a ship which is having the inlet orifices facing the direction of the motion of the ship. In this case the expression for propelling force and work done per second will be same as in the 1st case in which inlet orifices are at right angles to the ship. But the energy supplied by the jet will be different,

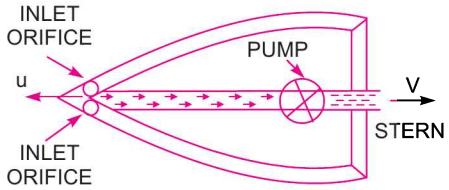


Fig. 17.27 *Inlet orifices facing the direction of ship.*

as in this case the water enters with a velocity equal to the velocity of the ship, *i.e.,* with a velocity u. Hence the expression for the energy supplied by the jet.

$$= \frac{1}{2} \text{ (Mass of water supplied per sec)} \times [V_r^2 - u^2]$$

$$= \frac{1}{2} (\rho a V_r) \times [V_r^2 - u^2]$$

where $V_r = (V + u)$ as in the previous case

$$\therefore \quad \text{K.E. supplied by jet} \quad = \frac{1}{2}\rho a(V + u)\,[(V + u)^2 - u^2] \qquad ...(17.35)$$

$$\therefore \quad \text{Efficiency of propulsion, } \eta = \frac{\text{Work done per sec by jet}}{\text{Energy supplied by jet}}$$

$$= \frac{\rho a(V + u) \times V \times u}{\frac{1}{2}\rho a(V + u)\left[(V + u)^2 - u^2\right]}$$

$$\{\because \text{ From equation (17.34) work done} = \rho a\,(V + u)\,V \times u\}$$

$$= \frac{2V \times u}{(V + u)^2 - u^2} = \frac{2Vu}{V^2 + u^2 + 2Vu - u^2} = \frac{2Vu}{V^2 + 2Vu} = \frac{2u}{V + 2u} \qquad ...(17.36)$$

Problem 17.33 *The water in a jet propelled boat is drawn through inlet openings facing the direction of motion of the ship. The boat is moving in sea-water with a speed of 30 km/hour. The absolute velocity of the jet of the water discharged at the back is 20 m/s and the area of the jet of water is 0.03 m^2. Find the propelling force and efficiency of propulsion.*

Solution. Given :

Speed of boat, $u = 30 \text{ km/hr} = \dfrac{30 \times 1000}{60 \times 60} = 8.33 \text{ m/s}$

Absolute velocity of jet, $V = 20$ m/s

Area of the jet, $a = .03 \text{ m}^2$.

(*i*) Propelling force is given by equation (17.33) as

$$F = \rho a\,(V + u) \times V = 1000 \times .03 \times (20 + 8.33) \times 20 = 16997.98 \text{ N.}$$

(*ii*) Efficiency of propulsion is given by equation (17.36) as

$$\eta = \frac{2u}{V + 2u} = \frac{2 \times 8.33}{20 + 2 \times 8.33} = \mathbf{0.4544} \text{ or } \mathbf{45.44\%}. \textbf{ Ans.}$$

HIGHLIGHTS

1. The force exerted by a jet of water on a stationary plate in the direction of the jet is given by

$F_x = \rho a V^2$ ...for a vertical plate

$= \rho a V^2 \sin^2 \theta$ ...for an inclined plate

$= \rho a V^2 (1 + \cos \theta)$ ...for a curved plate and jet strikes at the centre

$= 2\rho a V^2 \cos \theta$ ...for a curved plate and jet strikes at one of the tips of the jet.

where V = Velocity of the jet,

θ = Angle between the jet and the plate for inclined plate,

= Angle made by the jet with the direction of motion for curved plates.

2. When a jet of water strikes a vertical hinged plate, the angle of swing about the hinge is given by

$$\sin \theta = \frac{\rho a V^2}{W}$$

where V = Velocity of the jet of water, W = Weight of the hinged plate.

3. The force exerted by a jet of water on a moving plate, in the direction of the motion of the plate, is given by

$F_x = \rho a (V - u)^2$ ...for a moving vertical plate,

$= \rho a (V - u)^2 \sin^2 \theta$ ...for an inclined moving plate,

$= \rho a (V - u)^2 (1 + \cos \theta)$ ...when jet strikes the curved plate at the centre

4. When a jet of water strikes a curved moving plate at one of its tips and comes out at the other tip, the force exerted and work done are obtained from velocity triangles at inlet and outlet. The expression for force and work done are

$$F_x = \rho a V_{r_1} \left[V_{w_1} \pm V_{w_2} \right]$$

$$\text{Work done per second} = \rho a V_{r_1} \left[V_{w_1} \pm V_{w_2} \right] \times u$$

+ve sign is taken when β is an acute angle. If β is an obtuse angle then –ve sign is taken. If β is 90°, $V_{w_2} = 0$.

$$\text{Work done per second per kg of fluid} = \frac{1}{g}\left[V_{w_1} \pm V_{w_2} \right].$$

5. For a series of vanes, the force and work done are given as $F_x = \rho a V_{r_1} \left[V_{w_1} \pm V_{w_2} \right]$

$$\text{Work done/sec} = \rho a V_{r_1} \left[V_{w_1} \pm V_{w_2} \right] \times u$$

$$\text{Work done/sec per kg} = \frac{1}{g}\left[V_{w_1} \pm V_{w_2} \right] \times u.$$

6. Efficiency of a series of vanes is given as $\eta = \dfrac{2u(V-u)}{V^2}$

and condition of max. η is $u = \dfrac{V}{2}$

Max. $\eta = 50\%$.

7. For a curved radial vane, the work done per second $= \rho a V_1 \left[V_{w_1} u_1 \pm V_{w_2} u_2 \right]$

where V_1 = Absolute velocity of jet at inlet, V_{w_1} = Velocity of whirl at inlet

u_1 = Tangential velocity of vane at inlet, V_{w_2} = Velocity of whirl at outlet

u_2 = Tangential velocity of vane at outlet.

8. For a curved radial vane the efficiency is given by

$$\eta = \frac{\rho a V_1 \left[V_{w_1} u_1 \pm V_{w_2} u_2\right]}{\frac{1}{2}(\rho a V_1) \times V_1^2} = \frac{2\left[V_{w_1} u_1 \pm V_{w_2} u_2\right]}{V_1^2}.$$

9. Jet propulsion means the propulsion of a vessel with the help of the jet. The reaction of the jet is used for propelling the vessel. The propelling force exerted on a tank with a orifice is given by

$$F_x = \rho a\,(V + u) \times V$$

where V = Absolute velocity of the jet of water, u = Velocity of the tank.

10. The efficiency of propulsion is given by $\eta = \dfrac{2Vu}{(V+u)^2}$ and u = V for maximum efficiency

$\therefore$ Maximum $\eta = 50\%$.

11. Ships are also propelled by jets. The intake water by the centrifugal pump is taken by two ways. In one case, the water is taken from orifices which are at right angles to the direction of the motion of the ship and in the other case the water is taken through orifices which are facing the direction of motion of the ship.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define the terms : (*a*) Impact of jets, and (*b*) Jet propulsion.

2. Obtain an expression for the force exerted by a jet of water on a fixed vertical plate in the direction of the jet.

3. Show that the force exerted by a jet of water on an inclined fixed plate in the direction of the jet is given by,

$$F_x = \rho a V^2 \sin^2 \theta$$

where a = Area of the jet, V = Velocity of the jet

and θ = Inclination of the plate with the jet.

4. Prove that the force exerted by a jet of water on a fixed semi-circular plate in the direction of the jet when the jet strikes at the centre of the semi-circular plate is two times the force exerted by the jet on an fixed vertical plate.

5. Show that the angle of swing of a vertical hinged plate is given by $\sin\theta = \dfrac{\rho a V^2}{W}$

where V = Velocity of the jet striking the plate, a = Area of the jet, and W = Weight of the plate.

6. Differentiate between : (*i*) the force exerted by a jet of water on a fixed vertical plate and moving vertical plate, and (*ii*) the force exerted by a jet on a single curved moving plate and a series of curved moving plate.

7. Prove that the work done per second on a series of moving curved vanes by a jet of water striking at one of the tips of the vane is given by,

$$\text{Work done/sec} = \rho a V_1 \left[V_{w_1} \pm V_{w_2}\right] \times u.$$

8. Find an expression for the efficiency of a series of moving curved vanes when a jet of water strikes the vanes at one of its tips. Prove that maximum efficiency is when $u = V$ and the value of maximum efficiency is 50%.

9. Show that for a curved radial vane, the work done per second is given by, $\rho a V_1 \left[V_{w_1} u_1 \pm V_{w_2} u_2 \right]$.
10. Find an expression for the propelling force and the work done per second on a tank which is provided with an orifice through which jet of water is coming out and tank is free to move.
11. Show that the efficiency of a free jet striking normally on a series of flat plates mounted on the periphery of a wheel can never exceed 50%.
12. Show that the force exerted by a jet of water on moving inclined plate in the direction of jet is given by

$$F_x = \rho a (V - u)^2 \sin^2\theta$$

where a = area of jet,
θ = inclination of the plate with the jet, and
V = velocity of jet. *(J.N.T.U., Hyderabad, S 2002)*

(B) NUMERICAL PROBLEMS

1. Find the force exerted by a jet of water of diameter 100 mm on a stationary flat plate, when the jet strikes the plate normally with a velocity of 30 m/s. [**Ans.** 7068.6 N]
2. A jet of water of diameter 50 mm moving with a velocity of 20 m/s strikes a fixed plate in such a way that the angle between the jet and the plate is 60°. Find the force exerted by the jet on the pate (*i*) in the direction normal to the plate, and (*ii*) in the direction of the jet. [**Ans.** (*i*) 680.13 N, (*ii*) 589 N]
3. A jet of water of diameter 100 mm moving with a velocity of 30 m/s strikes a curved fixed symmetrical plate at the centre. Find the force exerted by the jet of water in the direction of the jet, if the jet is deflected through an angle of 120° at the outlet of the curved plate. [**Ans.** 10602.7 N]
4. A jet of water of the diameter 100 mm moving with a velocity of 20 m/s strikes a curved fixed plate tangentially at one end at an angle of 30° to the horizontal. The jet leaves the plate at an angle of 20° to the horizontal. Find the force exerted by the jet on the plate in the horizontal and vertical directions. [**Ans.** 5672.34 N, 496.3 N]
5. A jet of water of 30 mm diameter, moving with a velocity of 15 m/s, strikes a hinged square plate of weight 245.25 N at the centre of the plate. The plate is of uniform thickness. Find the angle through which the plate will swing. [**Ans.** $\theta = 40° 25.6'$]
6. A plate is acted upon at its centre by a jet of water of diameter 20 mm with a velocity of 20 m/s. The plate is hinged and is deflected through an angle of 15°. Find the weight of the plate. If the plate is not allowed to swing, what will be the force required at the lower edge of the plate to keep the plate in vertical position. [**Ans.** 485.5 N, 62.8 N]
7. A jet of water of diameter 150 mm strikes a flat plate normally with a velocity of 12 m/s. The plate is moving with a velocity of 6 m/s in the direction of the jet and away from the jet. Find : (*i*) the force exerted by the jet on the plate, (*ii*) work done by the jet on the plate per second, (*iii*) power of the jet, and (*iv*) efficiency of the jet. [**Ans.** (*i*) 636.3 N, (*ii*) 3817.6 Nm/s, (*iii*) 3.82 kW, (*iv*) 25%]
8. If in the problem 7, the jet strikes the plate in such a way that the normal on the plate makes an angle of 30° to the axis of the jet, find : (*i*) The normal force exerted on the plate, (*ii*) power, and (*iii*) efficiency of the jet. [**Ans.** (*i*) 551 N, (*ii*) 2.86 kW, (*iii*) 18.74%]
9. A jet of water of diameter 100 mm strikes a curved plate at its centre with a velocity of 15 m/s. The curved plate is moving with a velocity of 7 m/s in the direction of the jet. The jet is deflected through an angle of 150°. Assuming the plate smooth find : (*i*) force exerted on the plate in the direction of the jet, (*ii*) power of the jet, and (*iii*) efficiency. [**Ans.** (*i*) 938 N (*ii*) 6.56 kW, (*iii*) 49.53%]
10. A jet of water having a velocity of 30 m/s strikes a curved vane, which is moving with a velocity of 15 m/s. The jet makes an angle of 30° with the direction of motion of vane at inlet and leaves at an angle of 120° to the direction of motion of vane at outlet. Calculate : (*i*) Vane angles, if the water enters and leaves the vane without shock, (*ii*) Work done per second per unit weight of water striking the vanes per second. [**Ans.** (*i*) 53° 47.7′, 15° 41′, (*ii*) 44.15 Nm/N]

11. A jet of water of diameter 50 mm, having a velocity of 30 m/s strikes a curved vane which is moving with a velocity of 15 m/s in the direction of the jet. The jet leaves the vane at an angle of 60° to the direction of motion of vanes at outlet. Determine : (*i*) the force exerted by the jet on the vane in the direction of motion, (*ii*) work done per second by the jet. [**Ans.** (*i*) 662.5 N, (*ii*) 9937.5 Nm/s]

12. A jet of water having a velocity of 20 m/s strikes a curved vane which is moving with a velocity of 9 m/s. The vane is symmetrical and is so shaped that the jet is deflected through 120°. Find the angle of the jet at inlet of the vane so that there is no shock. What is the absolute velocity of the jet at outlet in magnitude and direction and the work done per second per unit weight of water strikings ? Assume the vane to be smooth. [**Ans.** 17°, 5.95 m/s, $\beta = 79° 6'$, 18.57 Nm/N]

13. A jet of water, having a velocity of 15 m/s, strikes a curved vane which is moving with a velocity of 6 m/s in the same direction as that of the jet at inlet. The vane is so shaped that the jet is deflected through 135°. The diameter of the jet is 150 mm. Assuming the vane to be smooth, find : (*i*) the force exerted by the jet on the vane in the direction of motion, (*ii*) power of the vane, and (*iii*) efficiency of the vane. [**Ans.** (*i*) 2443.5 N, (*ii*) 14.65 kW, (*iii*) 49.16%]

14. If in the above problem, the jet of water instead of striking a single plate, strikes a series of curved vanes, find : (*i*) force exerted by the jet on the vanes in the direction the motion, (*ii*) power of the vane, and (*iii*) efficiency of the vane. [**Ans.** (*i*) 4072.5 N, (*ii*) 24.43 kW, (*iii*) 81.9%]

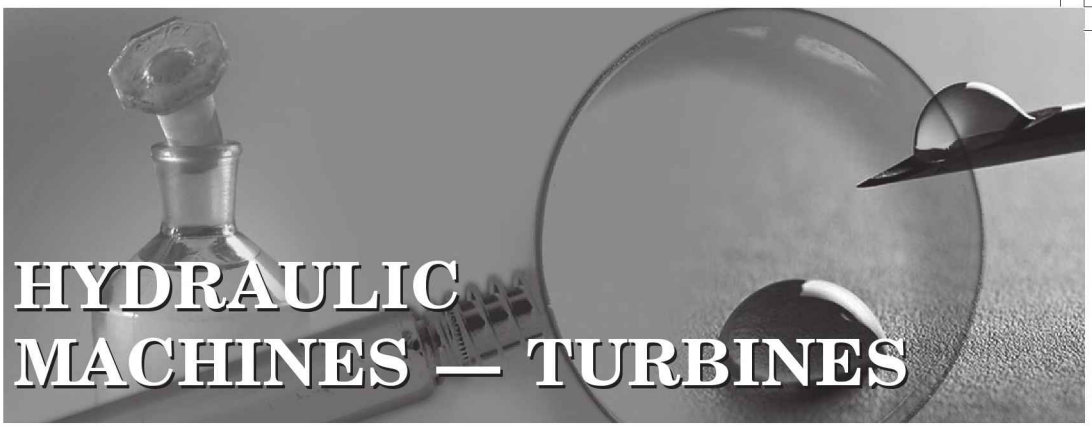
15. A jet of water having a velocity of 30 m/s, strikes a series of radial curved vanes mounted on a wheel which is rotating at 300 r.p.m. The jet makes an angle of 30° with the tangent to wheel at inlet and leaves the wheel with a velocity of 4 m/s at an angle of 120° to the tangent to the wheel at outlet. Water is flowing from outward in a radial direction. The outer and inner radii of the wheel are 0.6 m and 0.3 m respectively. Determine : (*i*) vane angles at inlet and outlet, (*ii*) work done per second per kg of water, and (*iii*) efficiency of the wheel. [**Ans.** (*i*) 42° 10.7′, 27°17.8′, (*ii*) 52.92, (*iii*) 56.5%]

16. The head of water from the centre of the orifice fitted to a tank is maintained at 6 m of water. The diameter of the orifice is 150 mm. The tank is fitted with frictionless wheels at the bottom and the tank is moving with a velocity of 4 m/s due to the reaction of the jet coming out from the orifice. Determine : (*i*) propelling force on the tank, (*ii*) work done per second, and (*iii*) efficiency of propulsion [**Ans.** (*i*) 2847.3 N, (*ii*) 11389 Nm/s, (*iii*) $\eta = 39.36\%$]

17. The water in a jet propelled boat is drawn mid-ship and is discharged at the back with an absolute velocity of 30 m/s. The cross-sectional area of the jet at the back is 0.04 m^2 and the boat is moving in sea-water with a speed of 30 km/hour. Determine : (*i*) propelling force of the boat, (*ii*) power, and (*iii*) efficiency of the jet propulsion. [**Ans.** (*i*) 45995.6 N, (*ii*) 383.14 kW, (*iii*) 34.02%]

18. The water in a jet propelled boat is drawn through inlet openings facing the direction of motion of the ship. The boat is moving in sea-water with a speed of 40 km/hr. The absolute velocity of the jet of the water discharged at the back is 40 m/s and the area of the jet of water is 0.04 m^2. Find the propelling force and efficiency of propulsion. [**Ans.** 81775.3 N, $\eta = 35.71\%$]

18 CHAPTER



HYDRAULIC MACHINES — TURBINES

▶ 18.1 INTRODUCTION

Hydraulic machines are defined as those machines which convert either hydraulic energy (energy possessed by water) into mechanical energy (which is further converted into electrical energy) or mechanical energy into hydraulic energy. The hydraulic machines, which convert the hydraulic energy into mechanical energy, are called *turbines* while the hydraulic machines which convert the mechanical energy into hydraulic energy are called *pumps*. Thus the study of hydraulic machines consists of study of turbines and pumps. Turbines consists of mainly study of Pelton turbine, Francis Turbine and Kaplan Turbine while pumps consist of study of centrifugal pump and reciprocating pumps.

▶ 18.2 TURBINES

Turbines are defined as the hydraulic machines which convert hydraulic energy into mechanical energy. This mechanical energy is used in running an electric generator which is directly coupled to the shaft of the turbine. Thus the mechanical energy is converted into electrical energy. The electric power which is obtained from the hydraulic energy (energy of water) is known as *Hydroelectric power*. At present the generation of hydroelectric power is the cheapest as compared by the power generated by other sources such as oil, coal etc.

▶ 18.3 GENERAL LAYOUT OF A HYDROELECTRIC POWER PLANT

Fig. 18.1 shows a general layout of a hydroelectric power plant which consists of :

(*i*) A dam constructed across a river to store water.

(*ii*) Pipes of large diameters called penstocks, which carry water under pressure from the storage reservoir to the turbines. These pipes are made of steel or reinforced concrete.

(*iii*) Turbines having different types of vanes fitted to the wheels.

(*iv*) Tail race, which is a channel which carries water away from the turbines after the water has worked on the turbines. The surface of water in the tail race channel is also known as tail race.

▶ 18.4 DEFINITIONS OF HEADS AND EFFICIENCIES OF A TURBINE

1. Gross Head. The difference between the head race level and tail race level when no water is flowing is known as Gross Head. It is denoted by 'H_g' in Fig. 18.1.

2. Net Head. It is also called effective head and is defined as the head available at the inlet of the turbine. When water is flowing from head race to the turbine, a loss of head due to friction between the water and penstocks occurs. Though there are other losses also such as loss due to bend, pipe fittings, loss at the entrance of penstock etc., yet they are having small magnitude as compared to head loss due to friction. If 'h_f'is the head loss due to friction between penstocks and water then net heat on turbine is given by

$$H = H_g - h_f \qquad ...(18.1)$$

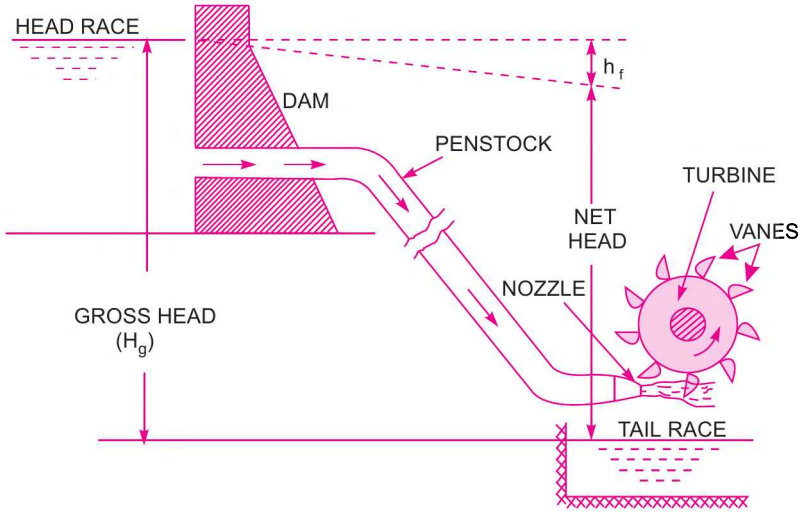


Fig. 18.1 *Layout of a hydroelectric power plant.*

where H_g = Gross head, $h_f = \dfrac{4 \times f \times L \times V^2}{D \times 2g}$,

in which V = Velocity of flow in penstock,

L = Length of penstock,

D = Diameter of penstock.

3. Efficiencies of a Turbine. The following are the important efficiencies of a turbine.

(*a*) Hydraulic Efficiency, η_h (*b*) Mechanical Efficiency, η_m

(*c*) Volumetric Efficiency, η_v and (*d*) Overall Efficiency, η_o

(*a*) **Hydraulic Efficiency (η_h).** It is defined as the ratio of power given by water to the runner of a turbine (runner is a rotating part of a turbine and on the runner vanes are fixed) to the power supplied by the water at the inlet of the turbine. The power at the inlet of the turbine is more and this power goes on decreasing as the water flows over the vanes of the turbine due to hydraulic losses as the vanes are not smooth. Hence, the power delivered to the runner of the turbine will be less than the power available at the inlet of the turbine. Thus, mathematically, the hydraulic efficiency of a turbine is written as

$$\eta_h = \frac{\text{Power delivered to runner}}{\text{Power supplied at inlet}} = \frac{\text{R.P.}}{\text{W.P.}} \qquad ...(18.2)$$

where R.P. = Power delivered to runner *i.e.*, runner power

$$= \frac{W}{g}\frac{\left[V_{w_1} \pm V_{w_2}\right] \times u}{1000} \text{ kW} \qquad \text{...for Pelton Turbine}$$

$$= \frac{W}{g}\frac{\left[V_{w_1}u_1 \pm V_{w_2}u_2\right]}{1000} \text{ kW} \qquad \text{...for a radial flow turbine}$$

W.P. = Power supplied at inlet of turbine and also called water power

$$= \frac{W \times H}{1000} \text{ kW} \qquad ...(18.3)$$

where W = Weight of water striking the vanes of the turbine per second
$= \rho g \times Q$ in which Q = Volume of water/s,
V_{w_1} = Velocity of whirl at inlet,
V_{w_2} = Velocity of whirl at outlet,
u = Tangential velocity of vane,
u_1 = Tangential velocity of vane at inlet for radial vane,
u_2 = Tangential velocity of vane at outlet for radial vane,
H = Net head on the turbine.

Power supplied at the inlet of turbine in S.I.units is known as water power. It is given by

$$\text{W.P.} = \frac{\rho \times g \times Q \times H}{1000} \text{ kW} \qquad ...(18.3A)$$

For water $\rho = 1000 \text{ kg/m}^3$

$$\therefore \quad \text{W.P.} = \frac{1000 \times g \times Q \times H}{1000} = g \times Q \times H \text{ kW} \qquad ...(18.3B)$$

The relation (18.3*B*) is only used when the flowing fluid is water. If the flowing fluid is other than the water, then relation (18.3*A*) is used.

(*b*) **Mechanical Efficiency (η_m).** The power delivered by water to the runner of a turbine is transmitted to the shaft of the turbine. Due to mechanical losses, the power available at the shaft of the turbine is less than the power delivered to the runner of a turbine. The ratio of the power available at the shaft of the turbine (known as S.P. or B.P.) to the power delivered to the runner is defined as mechanical efficiency. Hence, mathematically, it is written as

$$\eta_m = \frac{\text{Power at the shaft of the turbine}}{\text{Power delivered by water to the runner}} = \frac{\text{S.P.}}{\text{R.P.}} \qquad ...(18.4)$$

(*c*) **Volumetric Efficiency (η_v).** The volume of the water striking the runner of a turbine is slightly less than the vloume of the water supplied to the turbine. Some of the volume of the water is discharged to the tail race without striking the runner of the turbine. Thus the ratio of the volume of the water actually striking the runner to the volume of water supplied to the turbine is defined as volumetric efficiency. It is written as

$$\eta_v = \frac{\text{Volume of water actually striking the runner}}{\text{Volume of water supplied to the turbine}} \qquad ...(18.5)$$

(*d*) **Overall Efficiency (η_o).** It is defined as the ratio of power available at the shaft of the turbine to the power supplied by the water at the inlet of the turbine. It is written as :

$$\eta_o = \frac{\text{Volume available at the shaft of the turbine}}{\text{Power supplied at the inlet of the turbine}} = \frac{\text{Shaft power}}{\text{Water power}}$$

$$= \frac{\text{S.P.}}{\text{W.P.}}$$

$$= \frac{\text{S.P.}}{\text{W.P.}} \times \frac{\text{R.P.}}{\text{R.P.}} \qquad \text{(where R.P. = Power delivered to runner)}$$

$$= \frac{\text{S.P.}}{\text{R.P.}} \times \frac{\text{R.P.}}{\text{W.P.}}$$

$$= \eta_m \times \eta_h \qquad \left(\because \text{From equation (18.4)}, \frac{\text{S.P.}}{\text{R.P.}} = \eta_m \text{ and from equation (18.2)}, \frac{\text{R.P.}}{\text{W.P.}} = \eta_h \right) \qquad ...(18.6)$$

If shaft power (S.P.) is taken in kW then water power should also be taken in kW. Shaft power is commonly represented by P. But from equation (18.3*A*),

$$\text{Water power in kW} = \frac{\rho \times g \times Q \times H}{1000}, \text{ where } \rho = 1000 \text{ kg/m}^3$$

$$\therefore \quad \eta_o = \frac{\text{Shaft power in kW}}{\text{Water power in kW}} = \frac{P}{\left(\frac{\rho \times g \times Q \times H}{1000}\right)} \qquad ...(18.6A)$$

where P = Shaft power.

▶ 18.5 CLASSIFICATION OF HYDRAULIC TURBINES

The hydraulic turbines are classified according to the type of energy available at the inlet of the turbine, direction of flow through the vanes, head at the inlet of the turbine and specific speed of the turbines. Thus the following are the important classifications of the turbines :

1. According to the type of energy at inlet :
   (*a*) Impulse turbine, and (*b*) Reaction turbine.
2. According to the direction of flow through runner :
   (*a*) Tangential flow turbine, (*b*) Radial flow turbine,
   (*c*) Axial flow turbine, and (*d*) Mixed flow turbine.
3. According to the head at the inlet of turbine :
   (*a*) High head turbine, (*b*) Medium head turbine, and
   (*c*) Low head turbine.
4. According to the specific speed of the turbine :
   (*a*) Low specific speed turbine, (*b*) Medium specific speed turbine, and
   (*c*) High specific speed turbine.

If at the inlet of the turbine, the energy available is only kinetic energy, the turbine is known as **impulse turbine.** As the water flows over the vanes, the pressure is atmospheric from inlet to outlet of

the turbine. If at the inlet of the turbine, the water possesses kinetic energy as well as pressure energy, the turbine is known as **reaction turbine**. As the waters flows through the runner, the water is under pressure and the pressure energy goes on changing into kinetic energy. The runner is completely enclosed in an air-tight casing and the runner and casing is completely full of water.

If the water flows along the tangent of the runner, the turbine is known as **tangential flow turbine.** If the water flows in the radial direction through the runner, the turbine is called **radial flow turbine**. If the water flows from outwards to inwards, radially, the turbine is known as **inward** radial flow turbine, on the other hand, if water flows radially from inwards to outwards, the turbine is known as **outward** radial flow turbine. If the water flows through the runner along the direction parallel to the axis of rotation of the runner, the turbine is called **axial flow** turbine. If the water flows through the runner in the radial direction but leaves in the direction parallel to axis of rotation of the runner, the turbine is called **mixed flow** turbine.

▶ 18.6 PELTON WHEEL (OR TURBINE)

The Pelton wheel or Pelton turbine is a tangential flow impulse turbine. The water strikes the bucket along the tangent of the runner. The energy available at the inlet of the turbine is only kinetic energy. The pressure at the inlet and outlet of the turbine is atmospheric. This turbine is used for high heads and is named after L.A. Pelton, an American Engineer.

Fig. 18.1 shows the layout of a hydroelectric power plant in which the turbine is Pelton wheel . The water from the reservoir flows through the penstocks at the outlet of which a nozzle is fitted. The nozzle increases the kinetic energy of the water flowing through the penstock. At the outlet of the nozzle, the water comes out in the form of a jet and strikes the buckets (vanes) of the runner. The main parts of the Pelton turbine are :

1. Nozzle and flow regulating arrangement (spear),
2. Runner and buckets,
3. Casing, and
4. Breaking jet.

1. Nozzle and Flow Regulating Arrangement. The amount of water striking the buckets (vanes) of the runner is controlled by providing a spear in the nozzle as shown in Fig. 18.2. The spear is a conical needle which is operated either by a hand wheel or automatically in an axial direction depending upon the size of the unit. When the spear is pushed forward into the nozzle the amount of water striking the runner is reduced. On the other hand, if the spear is pushed back, the amount of water striking the runner increases.

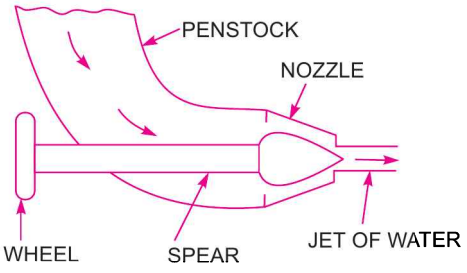


Fig. 18.2 *Nozzle with a spear to regulate flow.*

2. Runner with Buckets. Fig. 18.3 shows the runner of a Pelton wheel. It consists of a circular disc on the periphery of which a number of buckets evenly spaced are fixed. The shape of the buckets is of a double hemispherical cup or bowl. Each bucket is divided into two symmetrical parts by a dividing wall which is known as splitter.

The jet of water strikes on the splitter. The splitter divides the jet into two equal parts and the jet comes out at the outer edge of the bucket. The buckets are shaped in such a way that the jet gets deflected through 160° or 170°. The buckets are made of cast iron, cast steel bronze or stainless steel depending upon the head at the inlet of the turbine.

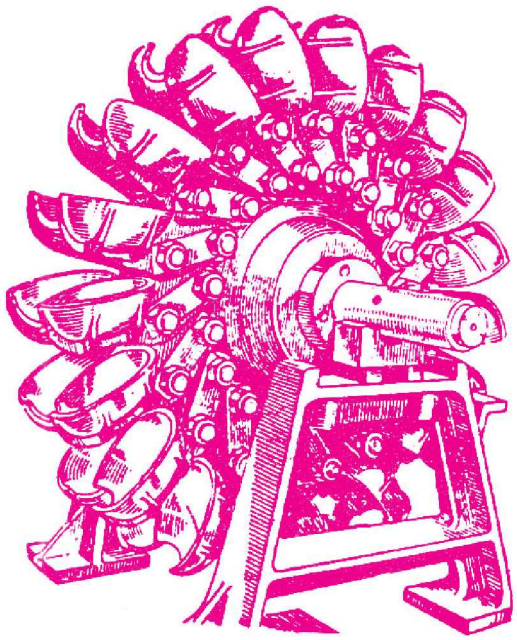


Fig. 18.3 *Runner of a pelton wheel.*

3. Casing. Fig. 18.4 shows a Pelton turbine with a casing. The function of the casing is to prevent the splashing of the water and to discharge water to tail race. It also acts as safeguard against accidents. It is made of cast iron or fabricated steel plates. The casing of the Pelton wheel does not perform any hydraulic function.

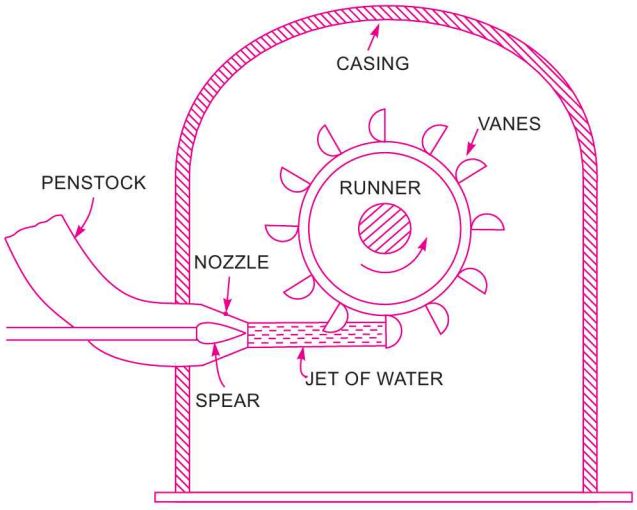


Fig. 18.4 *Pelton turbine.*

4. Breaking Jet. When the nozzle is completely closed by moving the spear in the forward direction, the amount of water striking the runner reduces to zero. But the runner due to inertia goes on revolving for a long time. To stop the runner in a short time, a small nozzle is provided which directs the jet of water on the back of the vanes. This jet of water is called breaking jet.

18.6.1 Velocity Triangles and Work done for Pelton Wheel. Fig. 18.5 shows the shape of the vanes or buckets of the Pelton wheel. The jet of water from the nozzle strikes the bucket at the splitter, which splits up the jet into two parts. These parts of the jet, glides over the inner surfaces and comes out at the outer edge. Fig. 18.5 (*b*) shows the section of the bucket at *Z-Z*. The splitter is the inlet tip and outer edge of the bucket is the outlet tip of the bucket. The inlet velocity triangle is drawn at the splitter and outlet velocity triangle is drawn at the outer edge of the bucket, by the same method as explained in Chapter 17.

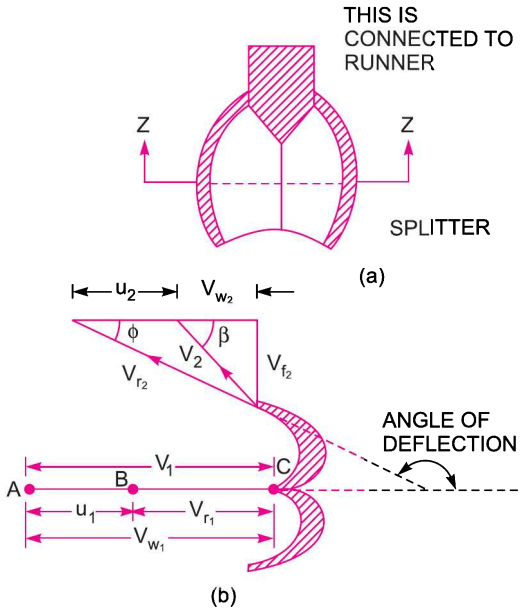


Fig. 18.5 *Shape of bucket.*

Let H = Net head acting on the Pelton wheel
$= H_g - h_f$

where H_g = Gross head and $h_f = \dfrac{4fLV^2}{D^* \times 2g}$

where D^* = Dia. of Penstock, N = Speed of the wheel in r.p.m.,
D = Diameter of the wheel, d = Diameter of the jet.

Then V_1 = Velocity of jet at inlet = $\sqrt{2gH}$ ...(18.7)

$$u = u_1 = u_2 = \frac{\pi DN}{60}.$$

The velocity triangle at inlet will be a straight line where

$$V_{r_1} = V_1 - u_1 = V_1 - u$$

$$V_{w_1} = V_1$$

$$\alpha = 0^\circ \text{ and } \theta = 0^\circ$$

From the velocity triangle at outlet, we have

$$V_{r_2} = V_{r_1} \text{ and } V_{w_2} = V_{r_2} \cos \phi - u_2.$$

The force exerted by the jet of water in the direction of motion is given by equation (17.19) as

$$F_x = \rho a V_1 [V_{w_1} + V_{w_2}] \quad ...(18.8)$$

As the angle β is an acute angle, +ve sign should be taken. Also this is the case of series of vanes, the mass of water striking is $\rho a V_1$ and not $\rho a V_{r_1}$. In equation (18.8), 'a' is the area of the jet which is given as

$$a = \text{Area of jet} = \frac{\pi}{4} d^2.$$

Now work done by the jet on the runner per second

$$= F_x \times u = \rho a V_1 [V_{w_1} + V_{w_2}] \times u \text{ Nm/s} \quad ...(18.9)$$

Power given to the runner by the jet

$$= \frac{\rho a V_1 \left[V_{w_1} + V_{w_2}\right] \times u}{1000} \text{ kW} \quad ...(18.10)$$

Work done/s per unit weight of water striking/s

$$= \frac{\rho a V_1 \left[V_{w_1} + V_{w_2}\right] \times u}{\text{Weight of water striking/s}}$$

$$= \frac{\rho a V_1 \left[V_{w_1} + V_{w_2}\right] \times u}{\rho a V_1 \times g} = \frac{1}{g} \left[V_{w_1} + V_{w_2}\right] \times u \quad ...(18.11)$$

The energy supplied to the jet at inlet is in the form of kinetic energy and is equal to $\frac{1}{2} m V^2$

$\therefore$ K.E. of jet per second $= \frac{1}{2} (\rho a V_1) \times V_1^2$

$\therefore$ **Hydraulic efficiency**, $\eta_h = \frac{\text{Work done per second}}{\text{K.E. of jet per second}}$

$$= \frac{\rho a V_1 \left[V_{w_1} + V_{w_2}\right] \times u}{\frac{1}{2} (\rho a V_1) \times V_1^2} = \frac{2 \left[V_{w_1} + V_{w_2}\right] \times u}{V_1^2} \quad ...(18.12)$$

Now $V_{w_1} = V_1$, $V_{r_1} = V_1 - u_1 = (V_1 - u)$

$\therefore$ $V_{r_2} = (V_1 - u)$

and $V_{w_2} = V_{r_2} \cos \phi - u_2 = V_{r_2} \cos \phi - u = (V_1 - u) \cos \phi - u$

Substituting the values of V_{w_1} and V_{w_2} in equation (18.12),

$$\eta_h = \frac{2 \left[V_1 + (V_1 - u) \cos \phi - u\right] \times u}{V_1^2}$$

$$= \frac{2 \left[V_1 - u + (V_1 - u) \cos \phi\right] \times u}{V_1^2} = \frac{2 (V_1 - u) [1 + \cos \phi] u}{V_1^2}. \quad ...(18.13)$$

The efficiency will be maximum for a given value of V_1 when

$$\frac{d}{du}(\eta_h) = 0 \quad \text{or} \quad \frac{d}{du}\left[\frac{2u(V_1 - u)(1 + \cos\phi)}{V_1^2}\right] = 0$$

or $$\frac{(1+\cos\phi)}{V_1^2}\frac{d}{du}(2uV_1 - 2u^2) = 0 \quad \text{or} \quad \frac{d}{du}[2uV_1 - 2u^2] = 0 \quad \left(\because \frac{1+\cos\phi}{V_1^2} \neq 0\right)$$

or $$2V_1 - 4u = 0 \quad \text{or} \quad u = \frac{V_1}{2} \qquad ...(18.14)$$

Equation (18.14) states that hydraulic efficiency of a Pelton wheel will be maximum when the velocity of the wheel is half the velocity of the jet of water at inlet. The expression for maximum efficiency will be obtained by substituting the value of $u = \frac{V_1}{2}$ in equation (18.13).

$$\therefore \quad \text{Max. } \eta_h = \frac{2\left(V_1 - \frac{V_1}{2}\right)(1+\cos\phi) \times \frac{V_1}{2}}{V_1^2}$$

$$= \frac{2 \times \frac{V_1}{2}(1+\cos\phi)\frac{V_1}{2}}{V_1^2} = \frac{(1+\cos\phi)}{2}. \qquad ...(18.15)$$

18.6.2 Points to be Remembered for Pelton Wheel

(*i*) The velocity of the jet at inlet is given by $V_1 = C_v\sqrt{2gH}$

where C_v = Co-efficient of velocity = 0.98 or 0.99

H = Net head on turbine

(*ii*) The velocity of wheel (u) is given by $u = \phi\sqrt{2gH}$

where ϕ = Speed ratio. The value of speed ratio varies from 0.43 to 0.48.

(*iii*) The angle of deflection of the jet through buckets is taken at 165° if no angle of deflection is given.

(*iv*) The mean diameter or the pitch diameter D of the Pelton wheel is given by

$$u = \frac{\pi DN}{60} \text{ or } D = \frac{60u}{\pi N}.$$

(*v*) **Jet Ratio.** It is defined as the ratio of the pitch diameter (D) of the Pelton wheel to the diameter of the jet (d). It is denoted by 'm' and is given as

$$m = \frac{D}{d} \text{ (= 12 for most cases)} \qquad ...(18.16)$$

(*vi*) Number of buckets on a runner is given by

$$Z = 15 + \frac{D}{2d} = 15 + 0.5\text{ m} \qquad ...(18.17)$$

where m = Jet ratio

(*vii*) **Number of Jets.** It is obtained by dividing the total rate of flow through the turbine by the rate of flow of water through a single jet.

Problem 18.1 *A Pelton wheel has a mean bucket speed of 10 metres per second with a jet of water flowing at the rate of 700 litres/s under a head of 30 metres. The buckets deflect the jet through an angle of 160°. Calculate the power given by water to the runner and the hydraulic efficiency of the turbine. Assume co-efficient of velocity as 0.98.*

Solution. Given :

Speed of bucket, $u = u_1 = u_2 = 10$ m/s

Discharge, $Q = 700$ litres/s $= 0.7$ m^3/s, Head of water, $H = 30$ m

Angle of deflection $= 160°$

$\therefore$ Angle, $\phi = 180° - 160° = 20°$

Co-efficient of velocity, $C_v = 0.98$.

The velocity of jet, $V_1 = C_v\sqrt{2gH} = 0.98\ \sqrt{2 \times 9.81 \times 30} = 23.77$ m/s

$\therefore$ $V_{r_1} = V_1 - u_1 = 23.77 - 10$

$= 13.77$ m/s

$V_{w_1} = V_1 = 23.77$ m/s

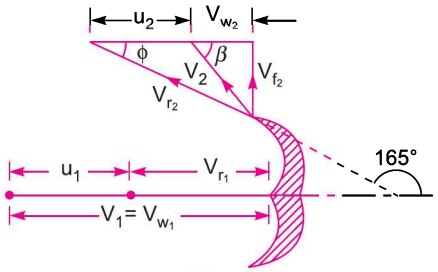


Fig. 18.6

From outlet velocity triangle,

$V_{r_2} = V_{r_1} = 13.77$ m/s

$V_{w_2} = V_{r_2} \cos \phi - u_2$

$= 13.77 \cos 20° - 10.0 = 2.94$ m/s

Work done by the jet per second on the runner is given by equation (18.9) as

$= \rho a V_1 \left[V_{w_1} + V_{w_2}\right] \times u$

$= 1000 \times 0.7 \times [23.77 + 2.94] \times 10$ ($\because$ $aV_1 = Q = 0.7$ m^3/s)

$= 186970$ Nm/s

$\therefore$ Power given to turbine $= \dfrac{186970}{1000} =$ **186.97 kW. Ans.**

The hydraulic efficiency of the turbine is given by equation (18.12) as

$$\eta_h = \frac{2\left[V_{w_1} + V_{w_2}\right] \times u}{V_1^2} = \frac{2[23.77 + 2.94] \times 10}{23.77 \times 23.77}$$

$=$ **0.9454** or **94.54%. Ans.**

Problem 18.2 *A Pelton wheel is to be designed for the following specifications :*
Shaft power = 11,772 kW ; Head = 380 metres ; Speed = 750 r.p.m. ; Overall efficiency = 86% ; Jet diameter is not to exceed one-sixth of the wheel diameter. Determine :

(i) The wheel diameter, (ii) The number of jets required, and
(iii) Diameter of the jet.

Take $K_{v_1} = 0.985$ and $K_{u_1} = 0.45$

Solution. Given :

Shaft power, S.P. $= 11{,}772$ kW

Head , $H = 380$ m

Speed, $N = 750$ r.p.m.

Overall efficiency, $\eta_0 = 86\%$ or 0.86

Ratio of jet dia. to wheel dia. $= \frac{d}{D} = \frac{1}{6}$

Co-efficient of velocity, $K_{v_1} = C_v = 0.985$

Speed ratio, $K_{u_1} = 0.45$

Velocity of jet, $V_1 = C_v\sqrt{2gH} = 0.985\sqrt{2 \times 9.81 \times 380} = 85.05$ m/s

The velocity of wheel, $u = u_1 = u_2$

$= \text{Speed ratio} \times \sqrt{2gH} = 0.45 \times \sqrt{2 \times 9.81 \times 380} = 38.85$ m/s

But $u = \frac{\pi DN}{60} \quad \therefore \quad 38.85 = \frac{\pi DN}{60}$

or $D = \frac{60 \times 38.85}{\pi \times N} = \frac{60 \times 38.85}{\pi \times 750} =$ **0.989 m. Ans.**

But $\frac{d}{D} = \frac{1}{6}$

$\therefore$ Dia. of jet, $d = \frac{1}{6} \times D = \frac{0.989}{6} =$ **0.165 m. Ans.**

Discharge of one jet, $q =$ Area of jet $\times$ Velocity of jet

$= \frac{\pi}{4} d^2 \times V_1 = \frac{\pi}{4}(.165) \times 85.05 \text{ m}^3/\text{s} = 1.818 \text{ m}^3/\text{s}$ ...(*i*)

Now $\eta_o = \frac{\text{S.P.}}{\text{W.P.}} = \frac{11772}{\frac{\rho g \times Q \times H}{1000}}$

$0.86 = \frac{11772 \times 1000}{1000 \times 9.81 \times Q \times 380}$, where Q = Total discharge

$\therefore$ Total discharge, $Q = \frac{11772 \times 1000}{1000 \times 9.81 \times 380 \times 0.86} = 3.672 \text{ m}^3/\text{s}$

$\therefore$ Number of jets $= \frac{\text{Total discharge}}{\text{Discharge of one jet}} = \frac{Q}{q} = \frac{3.672}{1.818} =$ **2 jets. Ans.**

Problem 18.3 *The penstock supplies water from a reservoir to the Pelton wheel with a gross head of 500 m. One third of the gross head is lost in friction in the penstock. The rate of flow of water through the nozzle fitted at the end of the penstock is 2.0 m^3/s. The angle of deflection of the jet is 165°. Determine the power given by the water to the runner and also hydraulic efficiency of the Pelton wheel. Take speed ratio = 0.45 and C_v = 1.0.*

Solution. Given :

Gross head, $H_g = 500$ m

Head lost in friction, $h_f = \frac{H_g}{3} = \frac{500}{3} = 166.7$ m

$\therefore$ Net head, $H = H_g - h_f = 500 - 166.7 = 333.30$ m

Discharge, $Q = 2.0$ m^3/s

Angle of deflection $= 165°$

$\therefore$ Angle, $\phi = 180° - 165° = 15°$

Speed ratio $= 0.45$

Co-efficient of velocity, $C_v = 1.0$

Velocity of jet, $V_1 = C_v\sqrt{2gH} = 1.0 \times \sqrt{2 \times 9.81 \times 333.3} = 80.86$ m/s

Velocity of wheel, $u =$ Speed ratio $\times \sqrt{2gH}$

or $u = u_1 = u_2 = 0.45 \times \sqrt{2 \times 9.81 \times 333.3} = 36.387$ m/s

$\therefore$ $V_{r_1} = V_1 - u_1 = 80.86 - 36.387$

$= 44.473$ m/s

Also $V_{w_1} = V_1 = 80.86$ m/s

From outlet velocity triangle, we have

$V_{r_2} = V_{r_1} = 44.473$

$V_{r_2} \cos \phi = u_2 + V_{w_2}$

or $44.473 \cos 15° = 36.387 + V_{w_2}$

or $V_{w_2} = 44.473 \cos 15° - 36.387 = 6.57$ m/s.

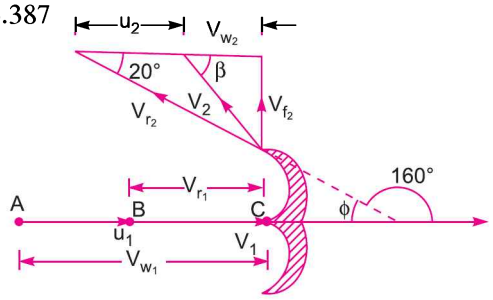


Fig. 18.7

Work done by the jet on the runner per second is given by equation (18.9) as

$$\rho a V_1 [V_{w_1} + V_{w_2}] \times u = \rho Q [V_{w_1} + V_{w_2}] \times u \qquad (\because \ aV_1 = Q)$$

$$= 1000 \times 2.0 \times [80.86 + 6.57] \times 36.387 = 6362630 \text{ Nm/s}$$

$\therefore$ Power given by the water to the runner in kW

$$= \frac{\text{Work done per second}}{1000} = \frac{6362630}{1000} = \mathbf{6362.63 \text{ kW. Ans.}}$$

Hydraulic efficiency of the turbine is given by equation (18.12) as

$$\eta_h = \frac{2\left[V_{w_1} + V_{w_2}\right] \times u}{V_1^2} = \frac{2[80.86 + 6.57] \times 36.387}{80.86 \times 80.86}$$

$= \mathbf{0.9731}$ or $\mathbf{97.31\%}$**. Ans.**

Problem 18.4 *A Pelton wheel is having a mean bucket diameter of 1 m and is running at 1000 r.p.m. The net head on the Pelton wheel is 700 m. If the side clearance angle is 15° and discharge through nozzle is 0.1 m^3/s, find :*

(i) Power available at the nozzle, and (ii) Hydraulic efficiency of the turbine.

Solution. Given :

Diameter of wheel, $D = 1.0$ m

Speed of wheel, $N = 1000$ r.p.m.

$\therefore$ Tangential velocity of the wheel, $u = \frac{\pi DN}{60} = \frac{\pi \times 1.0 \times 1000}{60} = 52.36$ m/s

Net head on turbine $H = 700$ m

Side clearance angle, $\phi = 15°$

Discharge, $Q = 0.1\ m^3/s$

Velocity of jet at inlet, $V_1 = C_v\sqrt{2gH} = 1 \times \sqrt{2 \times 9.81 \times 700}$

($\because$ Value of C_v is not given. Take it = 1.0)

or $V_1 = 117.19$ m/s

(*i*) Power available at the nozzle is given by equation (18.3) as

$$\text{W.P.} = \frac{W \times H}{1000} = \frac{\rho \times g \times Q \times H}{1000}$$

$$= \frac{1000 \times 9.81 \times 0.1 \times 700}{1000} = \textbf{686.7 kW. Ans.}$$

(*ii*) Hydraulic efficiency is given by equation (18.13) as

$$\eta_h = \frac{2(V_1 - u)(1 + \cos\phi)\, u}{V_1^2}$$

$$= \frac{2(117.19 - 52.36)(1 + \cos 15^\circ) \times 52.36}{117.19 \times 117.19}$$

$$= \frac{2 \times 64.83 \times 1.966 \times 52.36}{117.19 \times 117.19} = 0.9718 = \textbf{97.18 \%. Ans.}$$

Problem 18.5 *A Pelton wheel is working under a gross head of 400 m. The water is supplied through penstock of diameter 1 m and length 4 km from reservoir to the Pelton wheel. The co-efficient of friction for the penstock is given as .008. The jet of water of diameter 150 mm strikes the buckets of the wheel and gets deflected through an angle of 165°. The relative velocity of water at outlet is reduced by 15% due to friction between inside surface of the bucket and water. If the velocity of the buckets is 0.45 times the jet velocity at inlet and mechanical efficiency as 85% determine :*

(i) Power given to the runner, (ii) Shaft power,

(iii) Hydraulic efficiency and overall efficiency.

Solution. Given :

Gross head, $H_g = 400$ m

Diameter of penstock, $D = 1.0$ m

Length of penstock, $L = 4$ km $= 4 \times 1000 = 4000$ m

Co-efficient of friction, $f = .008$

Diameter of jet, $d = 150$ mm $= 0.15$ m

Angle of deflection $= 165°$

$\therefore$ Angle, $\phi = 180° - 165° = 15°$

Relative velocity at outlet, $V_{r_2} = 0.85\ V_{r_1}$

Velocity of bucket, $u = 0.45 \times$ Jet velocity

Mechanical efficiency, $\eta_m = 85\% = 0.85$

Let V^* = Velocity of water in penstock, and

V_1 = Velocity of jet of water.

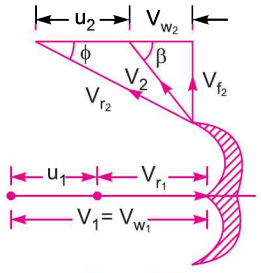


Fig. 18.8

Using continuity equation, we have

Area of penstock × V^* = Area of jet × V_1

or $$\frac{\pi}{4}D^2 \times V^* = \frac{\pi}{4}d^2 \times V_1$$

$$\therefore \quad V^* = \frac{d^2}{D^2} \times V_1 = \frac{0.15^2}{1.0^2} \times V_1 = .0225\ V_1 \qquad ...(i)$$

Applying Bernoulli's equation to the free surface of water in the reservoir and outlet of the nozzle, we get

$$H_g = \text{Head lost due to friction} + \frac{V_1^2}{2g}$$

or $$400 = \frac{4fLV^{*2}}{D \times 2g} + \frac{V_1^2}{2g} = \frac{4 \times .008 \times 4000 \times V^{*2}}{1.0 \times 2 \times 9.81} + \frac{V_1^2}{2g}$$

Substituting the value of V^* from equation (i), we get

$$400 = \frac{4 \times .008 \times 4000}{2 \times 9.81} \times (0.0225\ V_1)^2 + \frac{V_1^2}{2g}$$

$$= .0033\ V_1^2 + .051\ V_1^2 \text{ or } 400 = .0543\ V_1^2$$

$$\therefore \quad V_1 = \sqrt{\frac{400}{.0543}} = 85.83 \text{ m/s.}$$

Now velocity of bucket, $u_1 = 0.45\ V_1 = 0.45 \times 85.83 = 38.62$ m/s

From inlet velocity triangle, $V_{r_1} = V_1 - u_1 = 85.83 - 38.62 = 47.21$ m/s

$$V_{w_1} = V_1 = 85.83 \text{ m/s}$$

From outlet velocity triangle, $V_{r_2} = 0.85 \times V_{r_1} = 0.85 \times 47.21 = 40.13$ m/s

$$V_{w_2} = V_{r_2} \cos \phi - u_2 = 40.13 \cos 15° - 38.62$$

$$= 0.143 \text{ m/s} \qquad (\because \quad u = u_1 = u_2 = 38.62)$$

Discharge through nozzle is given as

$$Q = \text{Area of jet} \times \text{Velocity of jet} = a \times V_1$$

$$= \frac{\pi}{4}d^2 \times V_1 = \frac{\pi}{4}(.15)^2 \times 85.83 = 1.516 \text{ m}^3\text{/s}$$

Work done on the wheel per second is given by equation (18.9) as

$$= \rho a V_1 [V_{w_1} + V_{w_2}] \times u = \rho Q \left[V_{w_1} + V_{w_2}\right] \times u$$

$$= 1000 \times 1.516\ [85.83 + .143] \times 38.62 = 5033540 \text{ Nm/s}$$

(i) Power given to the runner in kW

$$= \frac{\text{Work done per second}}{1000} = \frac{5033540}{1000} = \mathbf{5033.54\ kW.\ Ans.}$$

(*ii*) Using equation (18.4) for mechanical efficiency,

$$\eta_m = \frac{\text{Power at the shaft}}{\text{Power given to the runner}} = \frac{\text{S.P.}}{5033.54}$$

$\therefore$ S.P. $= \eta_m \times 5033.54 = 0.85 \times 5033.54 =$ **4278.5 kW. Ans.**

(*iii*) Hydraulic efficiency is given by equation (18.12) as

$$\eta_h = \frac{2\left[V_{w_1} + V_{w_2}\right] \times u}{V_1^2}$$

$$= \frac{2\,[85.83 + .143] \times 38.62}{85.83 \times 85.83} = 0.9014 = \textbf{90.14\%. Ans.}$$

Overall efficiency is given by equation (18.6) as

$$\eta_0 = \eta_m \times \eta_h = 0.85 \times .9014 = 0.7662 \text{ or } \textbf{76.62\%. Ans.}$$

Problem 18.6 *A Pelton wheel nozzle, for which $C_v = 0.97$, is 400 m below the water surface of a lake. The jet diameter is 80 mm, the pipe diameter is 0.6 m, its length is 4 km and $f = 0.032$ in the formula $h_f = \dfrac{fLV^2}{2g \times D}$. The buckets, deflect the jet through 165° and they run at 0.48 times the jet speed, bucket friction reducing the relative velocity at outlet by 15% of the relative velocity at inlet. Mechanical efficiency = 90%. Find the flow rate and the shaft power developed by the turbine.*

Solution. Given :

$C_v = 0.97$

Gross head, $H_g = 400$ m

Dia. of jet, $d = 800 \text{ mm} = \dfrac{80}{1000}$ m

$= .08$ m

Dia. of pipe, $D = 0.6$ m

Length of pipe $L = 4 \text{ km} = 4000$ m

$f = .032$

Angle, $\phi = 180° - 165° = 15°$

Bucket speed, $u = 0.48$ times jet speed

Relative velocity at outlet $= 0.85$ times relative velocity at inlet

or $V_{r_2} = 0.85\ V_{r_1}$

Mechanical efficiency, $\eta_m = 0.90$.

Find. (*i*) Flow rate, and (*ii*) Shaft power, S.P.

Let V = Velocity of water in pipe, and

V_1 = Velocity of jet of water.

From continuity equation, we have

Area of pipe $\times V$ = Area of jet $\times V_1$

or $$\frac{\pi}{4}D^2 \times V = \frac{\pi}{4}d^2 \times V_1$$

$$= \frac{d^2}{D^2} \times V_1 = \left(\frac{.08}{0.60}\right)^2 \times V_1 = 0.0177\ V_1 \qquad ...(i)$$

Applying Bernoulli's equation to the free surface of water in the reservoir and the outlet of the nozzle, we get

Head at reservoir = Kinetic head of jet of water + Head lost due to friction in pipe + Head lost in nozzle

$$= \frac{V_1^2}{2g} + \frac{fLV^2}{D \times 2g} + \text{Head lost in nozzle} \qquad ...(ii)$$

Let V* = Theoretical velocity at the outlet of nozzle, V_1 = Actual velocity of jet of water

Then $$\frac{V_1}{V^*} = C_v \text{ or } V^* = \frac{V_1}{C_v}.$$

Now head lost in nozzle = Head corresponding to V^* – Head corresponding to V_1

$$= \frac{V^{*2}}{2g} - \frac{V_1^2}{2g} = \left(\frac{V_1}{C_v}\right)^2 \times \frac{1}{2g} - \frac{V_1^2}{2g} = \frac{V_1^2}{2g}\left(\frac{1}{C_v^2} - 1\right)$$

Substituting this value in equation (*ii*), we get

Head at reservoir $$= \frac{V_1^2}{2g} + \frac{fLV^2}{2g \times D} + \frac{V_1^2}{2g}\left(\frac{1}{C_v^2} - 1\right)$$

or $$400 = \frac{V_1^2}{2g} + \frac{0.032 \times 4000 \times V^2}{0.6 \times 2 \times 9.81} + \frac{V_1^2}{2g} \times \frac{1}{C_v^2} - \frac{V_1^2}{2g}$$

$$= \frac{0.032 \times 4000 \times (.0177 V_1)^2}{0.6 \times 2 \times 9.81} + \frac{V_1^2}{2 \times 9.81} \times \frac{1}{.97^2}$$

$$= .0034\ V_1^2 + .054\ V_1^2 \qquad (\because\ V = .0177\ V_1)$$

$$= 0.0574\ V_1^2$$

$\therefore$ $$V_1 = \sqrt{\frac{400}{.0574}} = 83.47 \text{ m/s}$$

Now velocity of bucket, $u_1 = 0.48 \times V_1 = 0.48 \times 83.47 = 40.06$ m/s

Refer to Fig. 18.8 (*a*), we have from inlet velocity triangle

$$V_{r_1} = V_1 - u_1 = 83.47 - 40.06 = 43.41 \text{ m/s}$$

$$V_{w_1} = V_1 = 83.47$$

From outlet velocity triangle,

$$V_{r_2} = 0.85\ V_{r_1} = 0.85 \times 43.41 = 36.898 \text{ m/s}$$

$$V_{w_2} = u_2 - V_{r_2} \cos \phi$$

$$= 40.06 - 36.898 \times \cos 15°$$

$$(\because\ u_1 = u_2 = 40.06)$$

$$= 4.42$$

Flow rate, Q = Area of jet × Velocity of jet

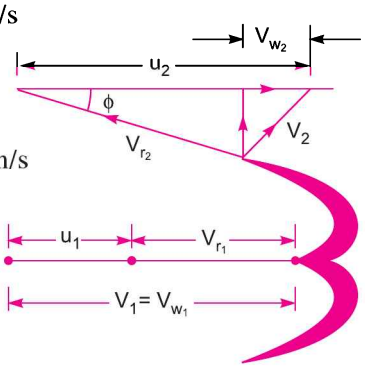


Fig. 18.8 (*a*)

$$= \frac{\pi}{4} d^2 \times V_1 = \frac{\pi}{4}(.08)^2 \times 83.47 = \mathbf{0.419\ m^3/s.\ Ans.}$$

Now using equation (18.4), $\eta_m = \dfrac{\text{S.P.}}{\text{Power given to the runner}}$

$\therefore$ S.P. $= \eta_m \times$ Power given to the runner

where power given to the runner in kW

$$= \frac{\text{Work done per second}}{1000}$$

Work done per second $= \dfrac{W}{g}\left[V_{w_1} - V_{w_2}\right] \times u$

(Here –ve sign is taken as V_{w_1} and V_{w_2} are in the same direction)

$$= \frac{\rho \times g \times Q}{g}[V_{w_1} - V_{w_2}] \times u_1 \qquad \{\because \quad u = u_1\}$$

$$= \frac{1000 \times 9.81 \times .419}{9.81}\ [83.47 - 4.42] \times 40.06 = 1326865\ \text{Nm/s}$$

$\therefore$ Power given to the runner $= \dfrac{1326865}{1000} = 1326.865$ kW

$\therefore$ S.P. $= \eta_m \times$ Power given to runner

$= 0.90 \times 1326.865 =$ **1194.18 kW. Ans.**

Problem 18.7 *A 137 mm diameter jet of water issuing from a nozzle impinges on the buckets of a Pelton wheel and the jet is deflected through an angle of 165° by the buckets. The head available at the nozzle is 400 m. Assuming co-efficient of velocity as 0.97, speed ratio as 0.46, and reduction in relative velocity while passing through buckets as 15%, find :*

(i) The force exerted by the jet on buckets in tangential direction,

(ii) The power developed.

Solution. Given :

Dia. of jet, $d = 137$ mm $= 0.137$ m

$\therefore$ Area of jet, $a = \dfrac{\pi}{4} \times 0.137 = 0.01474\ \text{m}^2$

Angle of deflection $= 165°$

$\therefore$ Angle, $\phi = 180° - 165° = 15°$

Head of water, $H = 400$ m

Co-efficient of velocity, $C_v = 0.97$

Speed ratio $= 0.46$

Relative velocity at outlet $= 0.85 \times$ relative velocity at inlet

or $V_{r_2} = 0.85\ V_{r_1}$

Now velocity of jet, $V_1 = C_v\sqrt{2gH} = 0.97\sqrt{2 \times 9.81 \times 400} = 85.93$ m/s

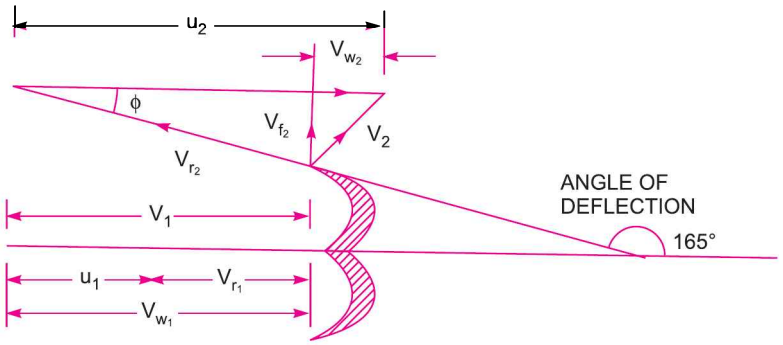


Fig. 18.8 (*b*)

Speed ratio $= \dfrac{u_1}{\sqrt{2gH}}$ or $0.46 = \dfrac{u_1}{\sqrt{2 \times 9.81 \times 400}}$

$\therefore \quad u_1 = 0.46 \times \sqrt{2 \times 9.81 \times 400} = 40.75$ m/s

Hence $V_{r_1} = V_1 - u_1 = 85.93 - 40.75 = 45.18$ m/s

and $V_{r_2} = 0.85\ V_{r_1} = 0.85 \times 45.18 = 38.40$ m/s

For Pelton turbine, $u_1 = u_2 = u = 40.75$ m/s

$V_{r_2} \cos \phi = 38.40 \times \cos 15° = 37.092$

Here $V_{r_2} \cos \phi$ is less than u_2. Hence velocity triangle at outlet will be as shown in Fig. 18.8 (*b*)

$\therefore \quad V_{w_2} = u_2 - V_{r_2} \cos \phi = 40.75 - 37.092 = 3.658$ m/s.

(*i*) Force exerted by jet on buckets in tangential direction is given by,

$$F_x = \rho a V_1 [V_{w_1} - V_{w_2}]$$

(Here –ve sign is taken as V_{w_1} and V_{w_2} are in the same direction)

$\therefore \quad F_x = 1000 \times 0.01474 \times 85.93\ (85.93 - 3.658)$ N = **104206 N. Ans.**

(*ii*) Power developed is given by,

$$\text{Power} = \frac{F_x \times u}{1000} \text{ kW} = \frac{104206 \times 40.75}{1000} = \textbf{4246.4 kW. Ans.}$$

Problem 18.8 *Two jets strike the buckets of a Pelton wheel, which is having shaft power as 15450 kW. The diameter of each jet is given as 200 mm. If the net head on the turbine is 400 m, find the overall efficiency of the turbine. Take $C_v = 1.0$.*

Solution. Given :

Number of jets $= 2$

Shaft power, S.P. $= 15450$ kW

Diameter of each jet, $d = 200$ mm $= 0.20$ m

$\therefore$ Area of each jet, $a = \dfrac{\pi}{4} d^2 = \dfrac{\pi}{4} (.2)^2 = 0.031416$ m^2

Net head, $H = 400$ m

Co-efficient of velocity, $C_v = 1.0$

Velocity of each jet, $V_1 = C_v\sqrt{2gH} = 1.0 \times \sqrt{2 \times 9.81 \times 400} = 88.58$ m/s

Discharge of each jet $= a \times V_1 = .031416 \times 88.58 = 2.78$ m^3/s

$\therefore$ Total discharge, $Q = 2 \times 2.78 = 5.56$ m^3/s

Power at the inlet of turbine,

$$\text{W.P.} = \frac{\rho \times g \times Q \times H}{1000} \text{ kW}$$

$$= \frac{1000 \times 9.81 \times 5.56 \times 400}{1000} = 21817.44 \text{ kW}$$

$\therefore$ Overall efficiency is given as

$$\eta_o = \frac{\text{S.P.}}{\text{W.P.}} = \frac{15450}{21817.44} = 0.708 = \mathbf{70.8\%}. \textbf{ Ans.}$$

Problem 18.9 *The water available for a Pelton wheel is 4 cumec and the total head from the reservoir to the nozzle is 250 metres. The turbine has two runners with two jets per runner. All the four jets have the same diameters. The pipe line is 3000 metres long. The efficiency of power transmission through the pipe line and the nozzle is 91% and efficiency of each runner is 90%. The velocity co-efficient of each nozzle is 0.975 and co-efficient of friction '4f' for the pipe is 0.0045. Determine :*

(i) The power developed by the turbine, (ii) The diameter of the jet, and

(iii) The diameter of the pipe line.

Solution. Given :

Total discharge, $Q = 4$ cumec $= 4.0$ m^3/s

Total or gross head, $H_g = 250$ m

Total number of jets $= 2 \times 2 = 4$

Length of pipe, $L = 3000$ m

Efficiency of the pipe line and nozzle = 91% or 0.91

Efficiency of runner* or $\eta_h = 90\%$ or 0.90

Co-efficient of velocity, $C_v = 0.975$

Co-efficient of friction, $4f = .0045$

Efficiency of power transmission through pipe lines and nozzle is given by

$$\eta = \frac{H_g - h_f}{H_g} \text{ or } 0.91 = \frac{250 - h_f}{250}$$

where h_f = Head lost due to friction.

$\therefore$ $h_f = 250 - 0.91 \times 250 = 22.5$ m

$\therefore$ Net head on the turbine, $H = H_g - h_f = 250 - 22.50 = 227.5$ m

Velocity of jet, $V_1 = C_v\sqrt{2gH} = 0.975\sqrt{2 \times 9.81 \times 227.5} = 65.14$ m/s.

---

* Efficiency of runner means the ratio of power delivered to the runner to the power at the inlet of turbine *i.e.*, hydraulic efficiency.

(*i*) Power at the inlet of the turbine is given as,

$$\text{W.P.} = \text{Kinetic energy of the jet/s}$$

$$= \frac{\frac{1}{2}mV_1^2}{1000} = \frac{\frac{1}{2}(\rho \times Q)V_1^2}{1000} = \frac{1}{2} \times 1000 \times \frac{4.0 \times 65.14^2}{1000} = 8486.44 \text{ kW}$$

But $$\eta_h = \frac{\text{Power developed by turbine}}{\text{W.P.}}$$

$$\therefore \quad 0.90 = \frac{\text{Power developed by turbine}}{8486.44}$$

$\therefore$ Power developed by turbine $= 0.90 \times 8486.44 =$ **7637.8 kW. Ans.**

(*ii*) Discharge per jet, $$q = \frac{\text{Total discharge}}{\text{No. of jets}} = \frac{4.0}{4.0} = 1.0 \text{ m}^3/\text{s}$$

But $q =$ Area of one jet $\times$ Velocity of jet

$$= \frac{\pi}{4}d^2 \times V_1, \qquad \text{where } d = \text{Diameter of each jet}$$

$$\therefore \quad 1.0 = \frac{\pi}{4}d^2 \times 65.14$$

$$\therefore \quad d = \sqrt{\frac{4 \times 1.0}{\pi \times 65.14}} = \textbf{0.14 m. Ans.}$$

(*iii*) Let $D =$ Diameter of pipe line

Then $$h_f = \frac{4 \times f \times L \times V^{*2}}{D \times 2g}, \qquad \text{where } V^* = \text{Velocity through pipe}$$

$$\therefore \quad V^* = \frac{Q}{\text{Area}} = \frac{Q}{\frac{\pi}{4}D^2} = \frac{4Q}{\pi D^2}$$

And $$h_f = \frac{.0045 \times 3000 \times \left(\frac{4Q}{\pi D^2}\right)^2}{D \times 2g}$$

or $$22.50 = \frac{.0045 \times 3000 \times 16 \times Q^2}{D \times 2 \times 9.81 \times \pi^2 \times D^4} = \frac{.0045 \times 3000 \times 16 \times (4)^2}{D^5 \times 2 \times 9.81 \times \pi^2} = \frac{17.84}{D^5}$$

$$\therefore \quad D^5 = \frac{17.84}{22.50} = 0.7933$$

$\therefore$ $D = (.7933)^{1/5} =$ **0.955 m. Ans.**

Problem 18.10 *The following data is related to a Pelton wheel :*

Head at the base of the nozzle = *80 m*

Diameter of the jet = *100 mm*

Discharge of the nozzle = *0.30 m^3/s*
Power at the shaft = *206 kW*
Power absorbed in mechanical resistance = *4.5 kW*
Determine (i) Power lost in nozzle and (ii) Power lost due to hydraulic resistance in the runner.

Solution. Given :

Head at the base of the nozzle, H_1 = 80 m

Diameter of the jet, d = 100 mm = 0.1 m

$\therefore$ Area of the jet, $a = \frac{\pi}{4} = (0.1)^2 = .007854$

Discharge of the nozzle, Q = 0.30 m^3/s

Shaft power, S.P. = 206 kW

Power absorbed in mechanical resistance = 4.5 kW

Now discharge Q = area of jet × velocity of jet = $a \times V_1$

$$0.30 = .007854 \times V_1$$

$$\therefore \quad V_1 = \frac{0.30}{.007854} = 38.197 \text{ m/s}$$

Power at the base of the nozzle in kW

$$= \frac{\rho \times g \times Q \times H_1}{1000} = \frac{1000 \times 9.81 \times 0.30 \times 80}{1000} = 235.44$$

Power corresponding to kinetic energy of the jet in kW

$$= \frac{1}{2}\frac{(m)\, V_1^2}{1000} = \frac{1}{2}\frac{(\rho \times aV_1)\, V_1^2}{1000} = \frac{1}{2}\frac{\rho \times Q \times V_1^2}{1000}$$

$$= \frac{1}{2} \times 1000 \times \frac{0.3 \times 38.197^2}{1000} = 218.85 \text{ kW}.$$

(*i*) Power at the base of the nozzle

= Power of the jet + Power lost in nozzle

or 235.44 = 218.85 + Power lost in nozzle

$\therefore$ Power lost in nozzle = 235.44 – 218.85 = **16.59 kW. Ans.**

(*ii*) Also power at the base of nozzle = power at the shaft + power lost in nozzle + power lost in runner + power lost due to mechanical resistance

$\therefore$ 235.44 = 206 + 16.59 + Power lost in runner + 4.5

$\therefore$ Power lost in runner = 235.44 – (206 + 16.59 + 4.5) = 235.44 – 227.09 = **8.35 kW. Ans.**

18.6.3 Design of Pelton Wheel. Design of Pelton wheel means the following data is to be determined :

1. Diameter of the jet (d),
2. Diameter of wheel (D),
3. Width of the buckets which is = 5 × d,
4. Depth of the buckets which is = 1.2 × d, and
5. Number of buckets on the wheel.

Size of buckets means the width and depth of the buckets.

Problem 18.11 *A Pelton wheel is to be designed for a head of 60 m when running at 200 r.p.m. The Pelton wheel develops 95.6475 kW shaft power. The velocity of the buckets = 0.45 times the velocity of the jet, overall efficiency = 0.85 and co-efficient of the velocity is equal to 0.98.*

Solution. Given :

Head, $H = 60$ m

Speed $N = 200$ r.p.m

Shaft power, S.P. = 95.6475 kW

Velocity of bucket, $u = 0.45 \times$ Velocity of jet

Overall efficiency, $\eta_o = 0.85$

Co-efficient of velocity, $C_v = 0.98$

Design of Pelton wheel means to find diameter of jet (d), diameter of wheel (D), Width and depth of buckets and number of buckets on the wheel.

(*i*) *Velocity of jet*, $V_1 = C_v \times \sqrt{2gH} = 0.98 \times \sqrt{2 \times 9.81 \times 60} = 33.62$ m/s

$\therefore$ Bucket velocity, $u = u_1 = u_2 = 0.45 \times V_1 = 0.45 \times 33.62 = 15.13$ m/s

But $u = \dfrac{\pi DN}{60}$, where D = Diameter of wheel

$\therefore$ $15.13 = \dfrac{\pi \times D \times 200}{60}$ or $D = \dfrac{60 \times 15.13}{\pi \times 200}$ = **1.44 m. Ans.**

(*ii*) *Diameter of the jet* (d)

Overall efficiency $\eta_o = 0.85$

But $\eta_o = \dfrac{\text{S.P.}}{\text{W.P.}} = \dfrac{95.6475}{\left(\dfrac{\text{W.P.}}{1000}\right)} = \dfrac{95.6475 \times 1000}{\rho \times g \times Q \times H}$ ($\because$ W.P. = ρgQH)

$$= \frac{95.6475 \times 1000}{1000 \times 9.81 \times Q \times 60}$$

$\therefore$ $Q = \dfrac{95.6475 \times 1000}{\eta_o \times 1000 \times 9.81 \times 60} = \dfrac{95.6475 \times 1000}{0.85 \times 1000 \times 9.81 \times 60} = 0.1912$ m^3/s.

But the discharge, Q = Area of jet $\times$ Velocity of jet

$\therefore$ $0.1912 = \dfrac{\pi}{4} d^2 \times V_1 = \dfrac{\pi}{4} d^2 \times 33.62$

$\therefore$ $d = \sqrt{\dfrac{4 \times 0.1912}{\pi \times 33.62}} = 0.085$ m = **85 mm. Ans.**

(*iii*) *Size of buckets*

Width of buckets $= 5 \times d = 5 \times 85 = 425$ mm

Depth of buckets $= 1.2 \times d = 1.2 \times 85 =$ **102 mm. Ans.**

(*iv*) *Number of buckets on the wheel* is given by equation (18.17) as

$$Z = 15 + \frac{D}{2d} = 15 + \frac{1.44}{2 \times .085} = 15 + 8.5 = \textbf{23.5 say 24. Ans.}$$

Problem 18.12 *Determine the power given by the jet of water to the runner of a Pelton wheel which is having tangential velocity as 20 m/s. The net head on the turbine is 50 m and discharge through the jet water is 0.03 m^3/s. The side clearance angle is 15° and take C_v = 0.975.*

Solution. Given :

Tangential velocity of wheel, $u = u_1 = u_2 = 20$ m/s

Net head, $H = 50$ m

Discharge , $Q = 0.03$ m^3/s

Side clearance angle, $\phi = 15°$

Co-efficient of velocity, $C_v = 0.975$

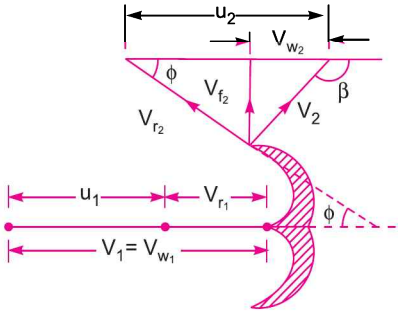


Fig. 18.9

Velocity of the jet, $V_1 = C_v \times \sqrt{2gH}$

$= 0.975 \times \sqrt{2 \times 9.81 \times 50}$

$= 30.54$ m/s

From inlet triangle, $V_{w_1} = V_1 = 30.54$ m/s

$V_{r_1} = V_{w_1} - u_1 = 30.54 - 20.0 = 10.54$ m/s

From outlet velocity triangle, we have

$V_{r_2} = V_{r_1} = 10.54$ m/s

$V_{r_2} \cos \phi = 10.54 \cos 15° = 10.18$ m/s

As $V_{r_2} \cos \phi$ is less than u_2, the velocity triangle at outlet will be as shown in Fig. 18.9.

$\therefore$ $V_{w_2} = u_2 - V_{r_2} \cos \phi = 20 - 10.18 = 9.82$ m/s.

Also as β is an obtuse angle, the work done per second on the runner,

$= \rho a V_1 \left[V_{w_1} - V_{w_2}\right] \times u = \rho Q \left[V_{w_1} - V_{w_2}\right] \times u$

$= 1000 \times .03 \times [30.54 - 9.82] \times 20 = 12432$ Nm/s

$\therefore$ Power given to the runner in kW $= \dfrac{\text{Work done per second}}{1000} = \dfrac{12432}{1000} =$ **12.432 kW. Ans.**

Problem 18.13 *The three-jet Pelton turbine is required to generate 10,000 kW under a net head of 400 m. The blade angle at outlet is 15° and the reduction in the relative velocity while passing over the blade is 5%. If the overall efficiency of the wheel is 80%, C_v = 0.98 and speed ratio = 0.46, then find: (i) the diameter of the jet, (ii) total flow in m^3/s and (iii) the force exerted by a jet on the buckets.*

If the jet ratio is not to be less than 10, find the speed of the wheel for a frequency of 50 hertz/sec and the corresponding wheel diameter.

Solution. Given :

No. of jets $= 3$

Total power, $P = 10000$ kW

Net head, $H = 400$ m

Blade angle at outlet, $\phi = 15°$

Relative velocity at outlet $= 0.95$ of relative velocity at inlet

or $V_{r_2} = 0.95\ V_{r_1}$

Overall efficiency, $\eta_o = 0.80$

Value of $C_v = 0.98$

Speed ratio $= 0.46$

Frequency, $f = 50$ hertz/sec

Now using equation (18.6 *A*), $\eta_o = \dfrac{P}{\left(\dfrac{\rho \times g \times Q \times H}{1000}\right)}$

where Q = Total discharge through three nozzles and $\rho = 1000$ kg/m^3

$$\therefore \quad 0.80 = \frac{10000}{\left(\dfrac{1000 \times 9.81 \times Q \times 400}{1000}\right)}$$

$$\therefore \quad Q = \frac{10000}{0.8 \times 9.81 \times 400} = \textbf{3.18 m}^3\textbf{/s. Ans.}$$

Discharge through one nozzle $= \dfrac{3.18}{3} = 1.06$ m^3/s.

(*i*) **Diameter of the jet (d).**

Discharge through one nozzle = Area of one jet × Velocity

But velocity of jet, $V_1 = C_v \times \sqrt{2gH} = 0.98 \times \sqrt{2 \times 9.81 \times 400} = 87$ m/s

$$\therefore \quad 1.06 = \frac{\pi}{4} d^2 \times 87$$

$$\therefore \quad d = \sqrt{\frac{4 \times 1.06}{\pi \times 87}} = 0.125 \text{ m} = \textbf{125 mm. Ans.}$$

(*ii*) **Total flow in m^3/s** $= 3.18$ m^3/s.

(*iii*) **Force exerted by a jet on the wheel.**

$$\text{Speed ratio} = \frac{u_1}{\sqrt{2gH}}$$

$$\therefore \quad u_1 = \text{Speed ratio} \times \sqrt{2gH} = 0.46 \times \sqrt{2 \times 9.81 \times 400} = 40.75 \text{ m/s}$$

Now $V_{r_1} = V_1 - u_1 = 87 - 40.75 = 46.25$ m/s

and $V_{r_2} = 0.95\ V_{r_1} = 0.95 \times 46.25 = 44.0$ m/s

$V_{w_1} = V_1 = 87$ m/s

$V_{w_2} = V_{r_2} \cos \phi - u_2 = 44 \times \cos 15° - 40.75$ ($\because \quad u_1 = u_2 = 40.75$ m/s)

$= 1.75$ m/s

Force exerted by a single jet on the buckets

$= \rho \times$ discharge through one jet $\times (V_{w_1} + V_{w_2})$

$= 1000 \times 1.06\ (87 + 1.75) = 94075$ N = **94.075 kN. Ans.**

(*iv*) **Jet ratio** $= 10 \text{ or } \frac{D}{d} = 10$

$\therefore$ Dia. of wheel, $D = 10 \times d = 10 \times 0.125 = 1.25$ m

But, $u_1 = \frac{\pi DN}{60}$

$\therefore$ $N = \frac{60 \times u_1}{\pi \times D} = \frac{60 \times 40.75}{\pi \times 1.25} = 620$ r.p.m.

Now using the relation, $N = \frac{60 \times f}{p}$

where f = frequency in hertz per second,

p = pairs of poles, and N = speed.

$\therefore$ $p = \frac{60 \times f}{N} = \frac{60 \times 50}{620} = 4.85$

Take the next whole number *i.e.*, 5. Hence, pairs of poles are 5.

Now corresponding to five pairs of poles, the speed of the turbine will become as given below :

$$N = \frac{60 \times f}{p} = \frac{60 \times 50}{5} = 600 \text{ r.p.m.}$$

But $u = \frac{\pi DN}{60}$

As the peripheral velocity is constant. Hence with the change of speed, diameter of wheel will change.

$\therefore$ $D = \frac{60 \times u}{\pi \times N} = \frac{60 \times 40.75}{\pi \times 600} = 1.3$ m

$\therefore$ Jet ratio becomes $= \frac{D}{d} = \frac{1.30}{0.125} > 10$

Hence the given condition is satisfied.

▶ 18.7 RADIAL FLOW REACTION TURBINES

Radial flow turbines are those tubines in which the water flows in the radial direction. The water may flow radially from outwards to inwards (*i.e.*, towards the axis of rotation) or from inwards to outwards. If the water flows from outwards to inwards through the runner, the turbine is known as inward radial flow turbine. And if the water flows from inwards to outwards, the turbine is known as outward radial flow turbine.

Reaction turbine means that the water at the inlet of the turbine possesses kinetic energy as well as pressure energy. As the water flows through the runner, a part of pressure energy goes on changing into kinetic energy. Thus the water through the runner is under pressure. The runner is completely enclosed in an air-tight casing and casing and the runner is always full of water.

18.7.1 Main Parts of a Radial Flow Reaction Turbine. The main parts of a radial flow reaction turbine are :

1. Casing,
2. Guide mechanism,
3. Runner, and
4. Draft-tube.

1. Casing. As mentioned above that in case of reaction turbine, casing and runner are always full of water. The water from the penstocks enters the casing which is of spiral shape in which area of cross-section of the casing goes on decreasing gradually. The casing completely surrounds the runner of the turbine. The casing as shown in Fig. 18.10 is made of spiral shape, so that the water may enter the runner at constant velocity throughout the circumference of the runner. The casing is made of concrete, cast steel or plate steel.

2. Guide Mechanism. It consists of a stationary circular wheel all round the runner of the turbine. The stationary guide vanes are fixed on the guide mechanism. The guide vanes allow the water to strike the vanes fixed on the runner without shock at inlet. Also by a suitable arrangement, the width between two adjacent vanes of guide mechanism can be altered so that the amount of water striking the runner can be varied.

3. Runner. It is a circular wheel on which a series of radial curved vanes are fixed. The surface of the vanes are made very smooth. The radial curved vanes are so shaped that the water enters and leaves the runner without shock. The runners are made of cast steel, cast iron or stainless steel. They are keyed to the shaft.

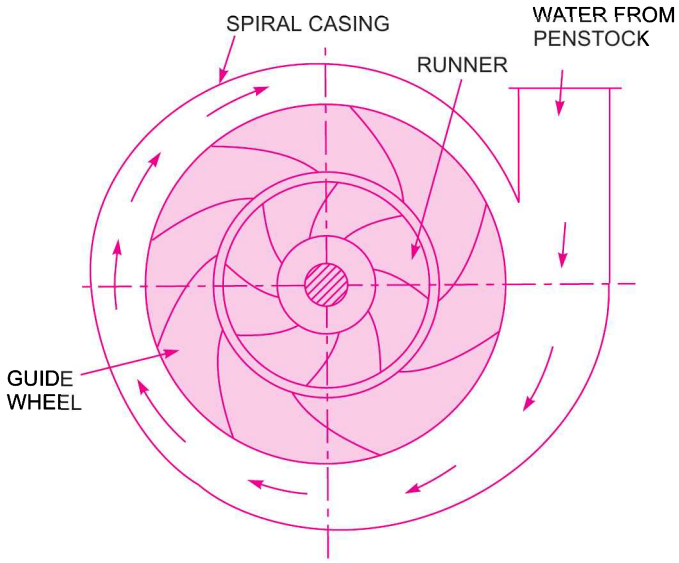


Fig. 18.10 *Main parts of a radial reaction turbines.*

4. Draft-tube. The pressure at the exit of the runner of a reaction turbine is generally less than atmospheric pressure. The water at exit cannot be directly discharged to the tail race. A tube or pipe of gradually increasing area is used for discharging water from the exit of the turbine to the tail race. This tube of increasing area is called draft tube.

18.7.2 Inward Radial Flow Turbine. Fig. 18.11 shows inward radial flow turbine, in which case the water from the casing enters the stationary guiding wheel. The guiding wheel consists of guide vanes which direct the water to enter the runner which consists of moving vanes. The water flows over the moving vanes in the inward radial direction and is discharged at the inner diameter of the runner. The outer diameter of the runner is the inlet and the inner diameter is the outlet.

Velocity Triangles and Work done by Water on Runner. In Chapter 17 (Art. 17.4.6), we have discussed in detail the force exerted by the water on the radial curved vanes fixed on a wheel. From the force exerted on the vanes,the work done by water, the horse power given by the water to the vanes and

efficiency of the vanes can be obtained. Also we have drawn velocity triangles at inlet and outlet of the moving radial vanes in Fig. 17.23. From the velocity triangles, the work done by the water on the runners, horse power and efficiency of the turbine can be obtained.

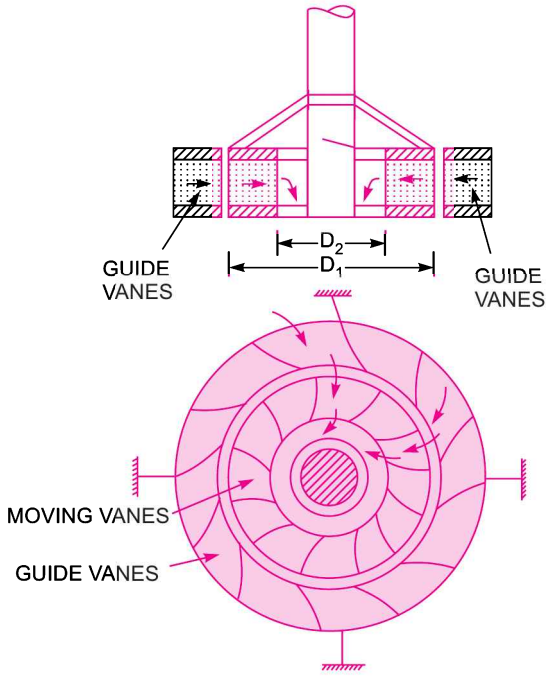


Fig. 18.11 *Inward radial flow turbine.*

The work done per second on the runner by water is given by equation (17.26) as

$$= \rho a V_1 [V_{w_1} u_1 \pm V_{w_2} u_2]$$

$$= \rho Q [V_{w_1} u_1 \pm V_{w_2} u_2] \qquad (\because \quad aV_1 = Q) \quad ...(18.18)$$

The equation (18.18) also represents the energy transfer per second to the runner.

where V_{w_1} = Velocity of whirl at inlet,

V_{w_2} = Velocity of whirl at outlet,

u_1 = Tangential velocity of wheel at inlet

$= \dfrac{\pi D_1 \times N}{60}$, where D_1 = Outer dia. of runner,

u_2 = Tangential velocity of wheel at outlet

$= \dfrac{\pi D_2 \times N}{60}$, where D_2 = Inner dia. of runner, N = Speed of the turbine in .r.p.m.

The work done per second per unit weight of water per second.

$$= \frac{\text{Work done per second}}{\text{Weight of water striking per second}}$$

$$= \frac{\rho Q \left[V_{w_1} u_1 \pm V_{w_2} u_2\right]}{\rho Q \times g} = \frac{1}{g}\left[V_{w_1} u_1 \pm V_{w_2} u_2\right] \qquad ...(18.19)$$

The equation (18.19) represents the energy transfer per unit weight/s to the runner. This equation is known by **Euler's equation** of hydrodynamics machines. This is also known as fundamental equation of hydrodynamic machines. This equation was given by Swiss scientist *L.* Euler.

In equation (18.19), +ve sign is taken if angle β is an acute angle. If β is an obtuse angle then –ve sign is taken. If β = 90°, then $V_{w_2} = 0$ and work done per second per unit weight of water striking/s become as

$$= \frac{1}{g} V_{w_1} u_1 \qquad ...(18.20)$$

Hydraulic efficiency is obtained from equation (18.2) as

$$\eta_h = \frac{\text{R.P.}}{\text{W.P.}} = \frac{\frac{W}{1000g}[V_{w_1} u_1 \pm V_{w_2} u_2]}{\frac{W \times H}{1000}} = \frac{(V_{w_1} u_1 \pm V_{w_2} u_2)}{gH} \qquad ...(18.20A)$$

where R.P. = Runner power *i.e.*, power delivered by water to the runner
W.P. = Water power

If the discharge is radial at outlet, then $V_{w_2} = 0$

$$\eta_h = \frac{V_{w_1} u_1}{gH} \qquad ...(18.20B)$$

18.7.3 Degree of Reaction. Degree of reaction is defined as the ratio of pressure energy change inside a runner to the total energy change inside the runner. It is represented by 'R'. Hence mathematically it can be written as

$$R = \frac{\text{Change of pressure energy inside the runner}}{\text{Change of total energy inside the runner}} \qquad ...(18.20C)$$

The equation (18.19) which is the fundamental equation of hydrodynamic machines, represents the energy transfer per unit weight to the runner. This is also known as the total energy change inside the runner per unit weight.

∴ Change of total energy per unit weight inside the runner

$$= \frac{1}{g}[V_{w_1} u_1 \pm V_{w_2} u_2]$$

Let H_e = Change of total energy per unit weight inside the runner.

Then $$H_e = \frac{1}{g}[V_{w_1} u_1 \pm V_{w_2} u_2] \qquad ...(18.20D)$$

Let us find the values of $V_{w_1} u_1$ and $V_{w_2} u_2$ from inlet and outlet velocity triangles.

Now from inlet velocity triangle, we know that [Refer to Fig. 18.11(*a*)]

$$V_{w_1} = u_1 + V_{r_1 x}, \quad \text{where } V_{r_1 x} = V_{r_1} \cos\theta = \sqrt{V_{r_1}^2 - V_{f_1}^2}$$

$$= u_1 + \sqrt{V_{r_1}^2 - V_{f_1}^2}$$

$$= u_1 + \sqrt{V_{r_1}^2 - (V_1^2 - V_{w_1}^2)} \quad [\because \text{From triangle } ABC,\ V_{f_1}^2 = V_1^2 - V_{w_1}^2]$$

$$\therefore \qquad (V_{w_1} - u_1) = \sqrt{V_{r_1}^2 - (V_1^2 - V_{w_1}^2)}$$

Squaring both sides, we get

$$(V_{w_1} - u_1)^2 = V_{r_1}^2 - (V_1^2 - V_{w_1}^2)$$

or $$V_{w_1}^2 + u_1^2 - 2V_{w_1}u_1 = V_{r_1}^2 - V_1^2 + V_{w_1}^2$$

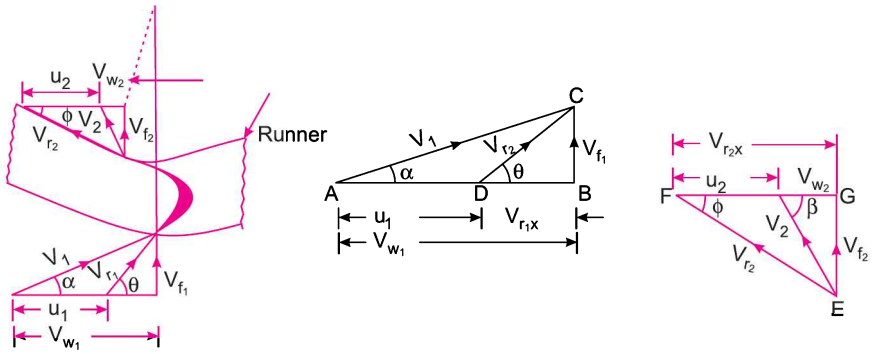


Fig. 18.11 (*a*)

or $$V_{w_1}^2 + u_1^2 - V_{r_1}^2 + V_1^2 - V_{w_1}^2 = 2V_{w_1}u_1$$

or $$u_1^2 - V_{r_1}^2 + V_1^2 = 2V_{w_1}u_1$$

or $$2V_{w_1}u_1 = u_1^2 - V_{r_1}^2 + V_1^2$$

or $$V_{w_1}u_1 = \frac{1}{2}[u_1^2 - V_{r_1}^2 + V_1^2] \quad ...(i)$$

Similarly from outlet triangle, we know that [Refer to Fig. 18.11(*a*)]

$$V_{w_2} = V_{r_2x} - u_2$$

$$= \sqrt{V_{r_2}^2 - V_{f_2}^2} - u_2, \text{ where } V_{r_2x} = V_{r_2}\cos\theta = \sqrt{V_{r_2}^2 - V_{f_2}^2}$$

$$= \sqrt{V_{r_2}^2 - (V_2^2 - V_{w_2}^2)} - u_2 \quad \therefore \quad V_{f_2}^2 = V_2^2 - V_{w_2}^2$$

$\therefore$ $$V_{w_2} + u_2 = \sqrt{V_{r_2}^2 - V_2^2 + V_{w_2}^2}$$

Squaring both sides, we get

$$(V_{w_2} + u_2)^2 = V_{r_2}^2 - V_2^2 + V_{w_2}^2$$

or $$V_{w_2}^2 + u_2^2 + 2V_{w_2}u_2 = V_{r_2}^2 - V_2^2 + V_{w_2}^2$$

or $$2V_{w_2}u_2 = V_{r_2}^2 - V_2^2 + V_{w_2}^2 - V_{w_2}^2 - u_2^2$$

or $$2V_{w_2}u_2 = V_{r_2}^2 - V_2^2 - u_2^2$$

or $$V_{w_2}u_2 = \frac{1}{2}[V_{r_2}^2 - V_2^2 - u_2^2] \quad ...(ii)$$

In the above case of velocity triangles under consideration, the change of total energy per unit weight inside the runner is equal to $\frac{1}{g}[V_{w_1}u_1 + V_{w_2}u_2]$

Substituting the values of $V_{w_1}u_1$ and $V_{w_2}u_2$ from equations (*i*) and (*ii*) into equation (18.20 *D*), we get Change of total energy per unit weight inside the runner as

$$H_e = \frac{1}{g}\left[\frac{1}{2}\left(u_1^2 - V_{r_1}^2 + V_1^2\right) + \frac{1}{2}\left(V_{r_2}^2 - V_2^2 - u_2^2\right)\right]$$

$$= \frac{1}{2g}\left[\left(u_1^2 - u_2^2\right) + \left(V_1^2 - V_2^2\right) + \left(V_{r_2}^2 - V_{r_1}^2\right)\right]$$

$$= \frac{V_1^2 - V_2^2}{2g} + \frac{u_1^2 - u_2^2}{2g} + \frac{V_{r_2}^2 - V_{r_1}^2}{2g} \qquad ...(18.20E)$$

The above equation consists of three terms. The first term represents the change in kinetic energy of the fluid per unit weight and the second term represents the change of energy per unit weight due to centrifugal action. The third term represents the change in static pressure energy per unit weight, as per Bernoulli's equation applied to relative flow through runner passage by reducing the rotating system into stationary system. We know that the energy change due to centrifugal action takes place in the form of pressure energy. [When a container containing a liquid is rotated, then due to centrifugal action there is change of pressure energy *i.e.*, $h = \frac{\Delta p}{\rho g} = \frac{u_2^2 - u_1^2}{2g}\Big]$. Hence, the last two terms in equation (18.20*E*) represents the change in pressure energy inside the runner passage per unit weight.

$\therefore$ Change in pressure energy inside the runner per unit weight $= \frac{u_1^2 - u_2^2}{2g} + \frac{V_{r_2}^2 - V_{r_1}^2}{2g}$ ...(*iii*)

Now the equation (18.20*C*) becomes as

$$R = \frac{\text{Change of pressure energy inside the runner per unit weight}}{\text{Change of total energy inside the runner per unit weight}}$$

$$= \left(\frac{\left(u_1^2 - u_2^2\right)}{2g} + \frac{V_{r_2}^2 - V_{r_1}^2}{2g}\right) \Bigg/ \left[\left(\frac{V_1^2 - V_2^2}{2g}\right) + \left(\frac{u_1^2 - u_2^2}{2g}\right) + \left(\frac{V_{r_2}^2 - V_{r_1}^2}{2g}\right)\right]$$

or $$R = \frac{\left(u_1^2 - u_2^2\right) + \left(V_{r_2}^2 - V_{r_1}^2\right)}{\left(V_1^2 - V_2^2\right) + \left(u_1^2 - u_2^2\right) + \left(V_{r_2}^2 - V_{r_1}^2\right)} \qquad ...(18.20F)$$

or $$R = \frac{\left(V_1^2 - V_2^2\right) + \left(u_1^2 - u_2^2\right) + \left(V_{r_2}^2 - V_{r_1}^2\right) - \left(V_1^2 - V_2^2\right)}{\left(V_1^2 - V_2^2\right) + \left(u_1^2 - u_2^2\right) + \left(V_{r_2}^2 - V_{r_1}^2\right)}$$

$$= 1 - \frac{\left(V_1^2 - V_2^2\right)}{\left(V_1^2 - V_2^2\right) + \left(u_1^2 - u_2^2\right) + \left(V_{r_2}^2 - V_{r_1}^2\right)} \qquad ...(18.20G)$$

From equation (18.20*E*) , we know that

$$H_e = \frac{V_1^2 - V_2^2}{2g} + \frac{u_1^2 - u_2^2}{2g} + \frac{V_{r_2}^2 - V_{r_1}^2}{2g}$$

or $$2gH_e = (V_1^2 - V_2^2) + (u_1^2 - u_2^2) + (V_{r_2}^2 - V_{r_1}^2)$$

Now the equation (18.20*G*) can be written as

$$R = 1 - \frac{\left(V_1^2 - V_2^2\right)}{2g\ H_e} \qquad ...(18.20H)$$

Values of *R* for Pelton turbine and other actual reaction turbines

(*i*) *For a Pelton turbine,*

$$u_1 = u_2 \text{ and } V_{r_2} = V_{r_1}$$

∴ From equation (18.20*G*)

$$R = 1 - \frac{\left(V_1^2 - V_2^2\right)}{\left(V_1^2 - V_2^2\right)} = 1 - 1 = 0$$

(*ii*) *For an actual reaction turbine,* generally, the angle β is 90° so that the loss of kinetic energy at outlet is minimum (*i.e.*, V_2 is minimum).

Hence in outlet velocity triangle, V_{w_2} becomes zero

(*i.e.*, $V_{w_2} = 0$). Also $V_2 = V_{f_2}$ [Refer to Fig. 18.11(*b*)]

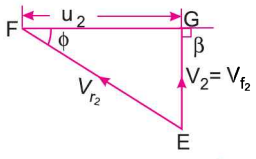


Fig. 18.11 (*b*)

Also there is not much change in velocity of flow. This means $V_{f_1} = V_{f_2}$

From equation (18.20*D*), we know that

$$H_e = \frac{1}{g}[V_{w_1}u_1 + V_{w_2}u_2]$$

$$= \frac{1}{g}V_{w_1}u_1 \qquad \left(\because\ V_{w_2} = 0\right)$$

$$= \frac{1}{g}[V_{f_1}\cot\alpha]\,[V_{f_1}\cot\alpha - V_{f_1}\cot\theta] \qquad \text{[Refer to Fig. 18.11(}a\text{)]}$$

$$[\because\ V_{w_1} = V_{f_1}\cot\alpha \text{ and } u_1 = V_{w_1} - V_{f_1}\cot\theta = V_{f_1}\cot\alpha - V_{f_1}\cot\theta]$$

$$= \frac{1}{g}V_{f_1}^2\cot\alpha\,[\cot\alpha - \cot\theta\,]$$

Now $$V_1^2 - V_2^2 = (V_{f_1}\ \text{cosec}\ \alpha)^2 - V_{f_2}^2 \qquad (\because\ V_2 = V_{f_2})$$

$$= V_{f_1}^2\ \text{cosec}^2\alpha - V_{f_1}^2 \qquad (\because\ V_{f_2} = V_{f_1})$$

or $$V_1^2 - V_2^2 = V_{f_1}^2\ (\text{cosec}^2\alpha - 1)$$

$$= V_{f_1}^2\cot^2\alpha \qquad (\because\ \text{cosec}^2\alpha - 1 = \cot^2\alpha)$$

Substituting the value of H_e and $(V_1^2 - V_2^2)$ in equation (18.20*H*), we get

$$R = 1 - \frac{V_{f_1}^2\cot^2\alpha}{2g \times [\frac{1}{g}V_{f_1}^2\cot\alpha(\cot\alpha - \cot\theta)]}$$

$$= 1 - \frac{\cot\alpha}{2(\cot\alpha - \cot\theta)} \qquad ...(18.20I)$$

18.7.4 Definitions. The following terms are generally used in case of reaction radial flow turbines which are defined as :

(*i*) **Speed Ratio.** The speed ratio is defind as = $\frac{u_1}{\sqrt{2gH}}$

where u_1 = Tangential velocity of wheel at inlet.

(*ii*) **Flow Ratio.** The ratio of the velocity of flow at inlet (V_{f_1}) to the velocity given $\sqrt{2gH}$ is known as flow ratio or it is given as

$$= \frac{V_{f_1}}{\sqrt{2gH}}, \text{ where } H = \text{Head on turbine}$$

(*iii*) **Discharge of the Turbine.** The discharge through a reaction radial flow turbine is given by

$$Q = \pi D_1 B_1 \times V_{f_1} = \pi D_2 \times B_2 \times V_{f_2} \qquad ...(18.21)$$

where D_1 = Diameter of runner at inlet,

B_1 = Width of runner at inlet,

V_{f_1} = Velocity of flow at inlet, and

D_2, B_2, V_{f_2} = Corresponding values at outlet.

If the thickness of vanes are taken into consideration, then the area through which flow takes place is given by $(\pi D_1 - n \times t)$

where n = Number of vanes on runner and t = Thickness of each vane

The discharge Q, then is given by $Q = (\pi D_1 - n \times t)\, B_1 \times V_{f_1}$ ...(18.22)

(*iv*) The head (H) on the turbine is given by $H = \frac{p_1}{\rho \times g} + \frac{V_1^2}{2g}$ ...(18.23)

where p_1 = Pressure at inlet.

(*v*) **Radial Discharge.** This means the angle made by absolute velocity with the tangent on the wheel is 90° and the component of the whirl velocity is zero. Radial discharge at outlet means $\beta = 90°$ and $V_{w_2} = 0$, while radial discharge at outlet means $\alpha = 90°$ and $V_{w_1} = 0$.

(*vi*) If there is no loss of energy when water flows through the vanes then we have

$$H - \frac{V_2^2}{2g} = \frac{1}{g}[V_{w_1} u_1 \pm V_{w_2} u_2]. \qquad ...(18.24)$$

Problem 18.14 *An inward flow reaction turbine has external and internal diameters as 1 m and 0.5 m respectively. The velocity of flow through the runner is constant and is equal to 1.5 m/s. Determine :*

(i) Discharge through the runner, and

(ii) Width of the turbine at outlet if the width of the turbine at inlet = 200 mm.

Solution. Given :

External diameter of turbine, $D_1 = 1$ m

Internal diameter of turbine, $D_2 = 0.5$ m

Velocity of flow at inlet and outlet, $V_{f_1} = V_{f_2} = 1.5$ m/s

Width of turbine at inlet, $B_1 = 200$ mm $= 0.20$ m

Let the width at outlet $= B_2$

Using equation (18.21) for discharge,

$$Q = \pi D_1 B_1 \times V_{f_1} = \pi \times 1 \times 0.20 \times 1.5 = \mathbf{0.9425\ m^3/s.\ Ans.}$$

Also $\pi D_1 B_1 V_{f_1} = \pi D_2 B_2 V_{f_2}$ or $D_1 B_1 = D_2 B_2$ $\left(\because \quad \pi V_{f_1} = \pi V_{f_2}\right)$

$$\therefore \quad B_2 = \frac{D_1 \times B_1}{D_2} = \frac{1 \times 0.20}{0.5} = 0.40 \text{ m} = \mathbf{400\ mm.\ Ans.}$$

Problem 18.15 *An inward flow reaction turbine has external and internal diameters as 0.9 m and 0.45 m respectively. The turbine is running at 200 r.p.m. and width of turbine at inlet is 200 mm. The velocity of flow through the runner is constant and is equal to 1.8 m/s. The guide blades make an angle of 10° to the tangent of the wheel and the discharge at the outlet of the turbine is radial. Draw the inlet and outlet velocity triangles and determine:*

(i) The absolute velocity of water at inlet of runner,
(ii) The velocity of whirl at inlet, (iii) The relative velocity at inlet,
(iv) The runner blade angles, (v) Width of the runner at outlet,
(vi) Mass of water flowing through the runner per second,
(vii) Head at the inlet of the turbine,
(viii) Power developed and hydraulic efficiency of the turbine.

Solution. Given :

External Dia., $D_1 = 0.9$ m
Internal Dia., $D_2 = 0.45$ m
Speed, $N = 200$ r.p.m.
Width at inlet, $B_1 = 200$ mm $= 0.2$ m
Velocity of flow, $V_{f_1} = V_{f_2} = 1.8$ m/s
Guide blade angle, $\alpha = 10°$
Discharge at outlet = Radial
$\therefore$ $\beta = 90°$ and $V_{w_2} = 0$

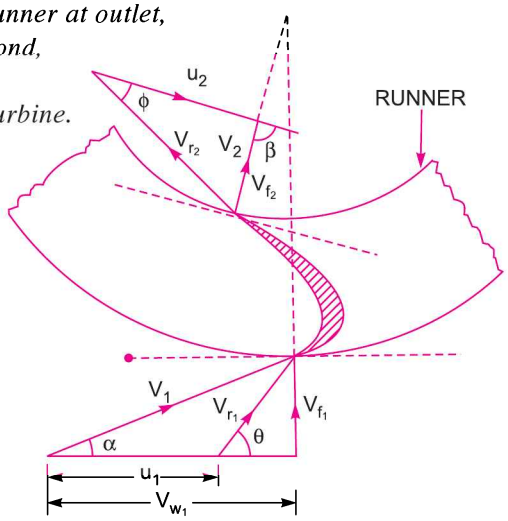


Fig 18.12

Tangential velocity of wheel at inlet and outlet are :

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times .9 \times 200}{60} = 9.424 \text{ m/s}$$

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times .45 \times 200}{60} = 4.712 \text{ m/s.}$$

(*i*) *Absolute velocity of water at inlet of the runner i.e.*, V_1

From inlet velocity triangle,

$$V_1 \sin \alpha = V_{f_1}$$

$$\therefore \quad V_1 = \frac{V_{f_1}}{\sin \alpha} = \frac{18}{\sin 10^\circ} = \textbf{10.365 m/s. Ans.}$$

(*ii*) *Velocity of whirl at inlet, i.e.*, V_{w_1}

$$V_{w_1} = V_1 \cos \alpha = 10.365 \times \cos 10° = \textbf{10.207 m/s. Ans.}$$

(*iii*) *Relative velocity at inlet, i.e.*, V_{r_1}

$$V_{r_1} = \sqrt{V_{r_1}^3 + (V_{w_1} - u_1)^2} = \sqrt{1.8^2 + (10.207 - 9.424)^2}$$

$$= \sqrt{3.24 + .613} = \textbf{1.963 m/s. Ans.}$$

(*iv*) *The runner blade angles* means the angle θ and ϕ

Now $$\tan \theta = \frac{V_{f_1}}{(V_{w_1} - u_1)} = \frac{1.8}{(10.207 - 9.424)} = 2.298$$

$\therefore$ $\theta = \tan^{-1} 2.298 = \textbf{66.48°}$ or $\textbf{66° 29'}$. **Ans.**

From outlet velocity triangle, we have

$$\tan \phi = \frac{V_{f_2}}{u_2} = \frac{1.8}{4.712} = \tan 20.9°$$

$\therefore$ $\phi = \mathbf{20.9°}$ or $\mathbf{20° 54.4'}$. **Ans.**

(*v*) *Width of runner at outlet, i.e.*, B_2

From equation (18.21), we have

$$\pi D_1 B_1 V_{f_1} = \pi D_2 B_2 V_{f_2} \text{ or } D_1 B_1 = D_2 B_2 \qquad (\because \ \pi V_{f_1} = \pi V_{f_2} \text{ as } V_{f_1} = V_{f_2})$$

$\therefore$ $$B_2 = \frac{D_1 B_1}{D_2} = \frac{0.90 \times 0.20}{0.45} = 0.40 \text{ m} = \mathbf{400 \text{ mm}}. \text{ Ans.}$$

(*vi*) *Mass of water flowing through the runner per second.*

The discharge, $Q = \pi D_1 B_1 V_{f_1} = \pi \times 0.9 \times 0.20 \times 1.8 = 1.0178 \text{ m}^3/\text{s}.$

$\therefore$ Mass $= \rho \times Q = 1000 \times 1.0178 \text{ kg/s} = \mathbf{1017.8 \text{ kg/s}}$. **Ans.**

(*vii*) *Head at the inlet of turbine, i.e., H.*

Using equation (18.24), we have

$$H - \frac{V_2^2}{2g} = \frac{1}{g}(V_{w_1} u_1 \pm V_{w_2} u_2) = \frac{1}{g}(V_{w_1} u_1) \qquad (\because \text{ Here } V_{w_2} = 0)$$

$$H = \frac{1}{g} V_{w_1} u_1 + \frac{V_2^2}{2g} = \frac{1}{9.81} \times 10.207 \times 9.424 + \frac{1.8^2}{2 \times 9.81} \quad (\because V_2 = V_{f_2})$$

$$= 9.805 + 0.165 = \mathbf{9.97 \text{ m}}. \text{ Ans.}$$

(*viii*) *Power developed, i.e.*, $P = \dfrac{\text{Work done per second on runner}}{1000}$

$$= \frac{\rho Q \left[V_{w_1} u_1\right]}{1000} \qquad \text{[Using equation (18.18)]}$$

$$= 1000 \times \frac{1.0178 \times 10.207 \times 9.424}{1000} = \mathbf{97.9 \text{ kW}}. \text{ Ans.}$$

Hydraulic efficiency is given by equation (18.20*B*) as

$$\eta_h = \frac{V_{w_1} u_1}{gH} = \frac{10.207 \times 9.424}{9.81 \times 9.97} = 0.9834 = \mathbf{98.34\%}. \text{ Ans.}$$

Problem 18.16 *A reaction turbine works at 450 r.p.m. under a head of 120 metres. Its diameter at inlet is 120 cm and the flow area is 0.4 m^2. The angles made by absolute and relative velocities at inlet are 20° and 60° respectively with the tangential velocity. Determine :*

(a) The volume flow rate, (b) The power developed, and

(c) Hydraulic efficiency.

Assume whirl at outlet to be zero.

Solution. Given :

Speed of turbine, $N = 450$ r.p.m.

Head, $H = 120$ m

Diameter at inlet, $D_1 = 120 \text{ cm} = 1.2 \text{ m}$

Flow area, $\pi D_1 \times B_1 = 0.4 \text{ m}^2$

Angle made by absolute velocity at inlet, $\alpha = 20°$

Angle made by the relative velocity at inlet, $\theta = 60°$

Whirl at outlet, $V_{w_2} = 0$

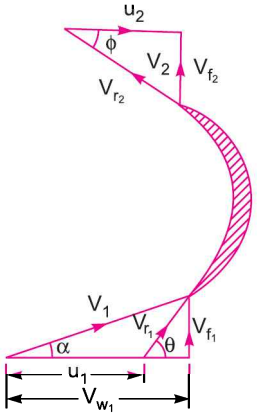


Fig. 18.13

Tangential velocity of the turbine at inlet,

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 1.2 \times 450}{60} = 28.27 \text{ m/s}$$

From inlet velocity triangle,

$$\tan \alpha = \frac{V_{f_1}}{V_{w_1}} \text{ or } \tan 20° = \frac{V_{f_1}}{V_{w_1}} \text{ or } \frac{V_{f_1}}{V_{w_1}} = \tan 20° = 0.364$$

$$\therefore \quad V_{f_1} = 0.364\ V_{w_1}$$

Also
$$\tan \theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{0.364\ V_{w_1}}{V_{w_1} - 28.27} \qquad (\because \quad V_{f_1} = 0.364\ V_{w_1})$$

or
$$\frac{0.364\ V_{w_1}}{V_{w_1} - 28.27} = \tan \theta = \tan 60° = 1.732$$

$$\therefore \quad 0.364 V_{w_1} = 1.732(V_{w_1} - 28.27) = 1.732 V_{w_1} - 48.96$$

or
$$(1.732 - 0.364)\ V_{w_1} = 48.96$$

$$\therefore \quad V_{w_1} = \frac{48.96}{(1.732 - 0.364)} = 35.789 = 35.79 \text{ m/s.}$$

From equation (*i*), $V_{f_1} = 0.364 \times V_{w_1} = 0.364 \times 35.79 = 13.027$ m/s.

(*a*) Volume flow rate is given by equation (18.21) as $Q = \pi D_1 B_1 \times V_{f_1}$

But
$$\pi D_1 \times B_1 = 0.4 \text{ m}^2 \qquad \text{(given)}$$

$$Q = 0.4 \times 13.027 = \mathbf{5.211\ m^3/s.\ Ans.}$$

(*b*) Work done per sec on the turbine is given by equation (18.18),

$$= \rho Q\ [V_{w_1} u_1] \qquad (\because \quad V_{w_2} = 2)$$

$$= 1000 \times 5.211\ [35.79 \times 28.27] = 5272402 \text{ Nm/s}$$

$$\therefore \quad \text{Power developed in kW} = \frac{\text{Work done per second}}{1000} = \frac{5272402}{1000} = \mathbf{5272.402\ kW.\ Ans.}$$

(*c*) The hydraulic efficiency is given by equation (18.20*B*) as

$$\eta_h = \frac{V_{w_1} u_1}{gH} = \frac{35.79 \times 28.27}{9.81 \times 120} = 0.8595 = \mathbf{85.95\%.\ Ans.}$$

Problem 18.17 *As inward flow reaction turbine has external and internal diameters as 1.0 m and 0.6 m respectively. The hydraulic efficiency of the turbine is 90% when the head on the turbine is 36 m. The velocity of flow at outlet is 2.5 m/s and discharge at outlet is radial. If the vane angle at outlet is 15° and width of the wheel is 100 mm at inlet and outlet, determine : (i) the guide blade angle, (ii) speed of the turbine, (iii) vane angle of the runner at inlet, (iv) volume flow rate of turbine and (v) power developed.*

Solution. Given :

External diameter, $D_1 = 1.0$ m

Internal diameter, $D_2 = 0.6$ m
Hydraulic efficiency, $\eta_h = 90\% = 0.90$
Head, $H = 36$ m
Velocity of flow at outlet, $V_{f_2} = 2.5$ m/s
Discharge is radial, $V_{w_2} = 0$
Vane angle at outlet, $\phi = 15°$
Width of wheel, $B_1 = B_2 = 100$ mm $= 0.1$ m

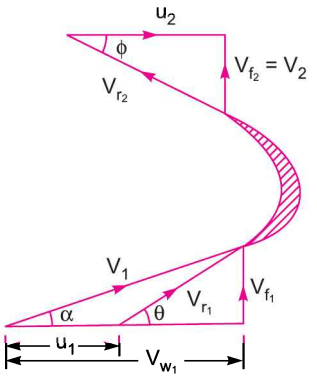


Fig. 18.14

Using equation (18.20 *B*) for hydraulic efficiency as

$$\eta_h = \frac{V_{w_1} u_1}{gH} \text{ or } 0.90 = \frac{V_{w_1} . u_1}{9.81 \times 36}$$

$$\therefore \quad V_{w_1} u_1 = 0.90 \times 9.81 \times 36 = 317.85 \qquad ...(i)$$

From outlet velocity triangle, $\tan \phi = \frac{V_{f_2}}{u_2} = \frac{2.5}{u_2}$

$$\therefore \quad u_2 = \frac{2.5}{\tan \phi} = \frac{2.5}{\tan 15°} = 9.33$$

But $$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.6 \times N}{60}$$

$$\therefore \quad 9.33 = \frac{\pi \times 0.6 \times N}{60} \text{ or } N = \frac{60 \times 9.33}{\pi \times 0.6} = \mathbf{296.98.\ Ans.}$$

$$\therefore \quad u_1 = \frac{\pi \times D_1 \times N}{60} = \frac{\pi \times 1.0 \times 296.98}{60} = 15.55 \text{ m/s.}$$

Substituting this value of 'u_1' in equation (*i*),

$$V_{w_1} \times 15.55 = 317.85$$

$$\therefore \quad V_{w_1} = \frac{317.85}{15.55} = 20.44 \text{ m/s}$$

Using equation (18.21), $\pi D_1 B_1 V_{f_1} = \pi D_2 B_2 V_{f_2}$ or $D_1 V_{f_1} = D_2 V_{f_2}$ $(\because \ B_1 = B_2)$

$$\therefore \quad V_{f_1} = \frac{D_2 \times V_{f_2}}{D_1} = \frac{0.6 \times 2.5}{1.0} = 1.5 \text{ m/s.}$$

(*i*) *Guide blade angle* (α).

From inlet velocity triangle, $\tan \alpha = \frac{V_{f_1}}{V_{w_1}} = \frac{1.5}{20.44} = 0.07338$

$$\therefore \quad \alpha = \tan^{-1} 0.07338 = \mathbf{4.19°} \text{ or } \mathbf{4° 11.8'.\ Ans.}$$

(*ii*) *Speed of the turbine,* $N = \mathbf{296.98\ r.p.m.\ Ans.}$

(*iii*) *Same angle of runner at inlet* (θ)

$$\tan\theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{1.5}{(20.44 - 15.55)} = 0.3067$$

$\therefore$ $\theta = \tan^{-1} .3067 = 17.05°$ or **17° 3′. Ans.**

(*iv*) *Volume flow rate of turbine* is given by equation (18.21) as

$$= \pi D_1 B_1 V_{f_1} = \pi \times 1.0 \times 0.1 \times 1.5 = \mathbf{0.4712\ m^3/s.\ Ans.}$$

(*v*) *Power developed* (in kW)

$$= \frac{\text{Work done per second}}{1000} = \frac{\rho Q \left[V_{w_1} u_1\right]}{1000}$$

[Using equation (18.18) and $V_{w_2} = 0$]

$$= 1000 \times \frac{0.4712 \times 20.44 \times 15.55}{1000} = \mathbf{149.76\ kW.\ Ans.}$$

Problem 18.18 *An inward flow reaction turbine has an exit diameter of 1 metre and its breadth at inlet is 250 mm. If the velocity of flow at inlet is 2 metres/s, find the mass of water passing through the turbine per second. Assume 10% of the area of flow is blocked by blade thickness. If the speed of the runner is 210 r.p.m. and guide blades make an angle of 10° to the wheel tangent, draw the inlet velocity triangle, and find :*

(i) the runner vane angle at inlet, (ii) velocity of wheel at inlet,
(iii) the absolute velocity of water leaving the guide vanes, and
(iv) the relative velocity of water entering the runner blade.

Solution. Given :

Exit or External diameter, $D_1 = 1.0$ m

Breadth at inlet, $B_1 = 250$ mm $= 0.25$ m

Velocity of flow at inlet, $V_{f_1} = 2.0$ m/s

Area blocked by vanes $= 10\%$

Speed, $N = 210$ r.p.m.

Guide blade angle, $\alpha = 10°$

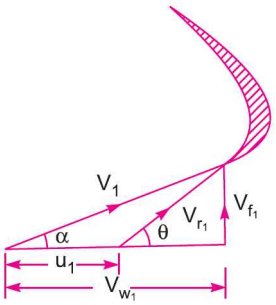


Fig. 18.15

Tangential velocity of wheel at inlet,

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 1.0 \times 210}{60} = 10.99 \text{ m/s}$$

$$\text{Area blocked by vane thickness} = \frac{10}{100} \times \pi D_1 B_1 = 0.1\ \pi D_1 B_1$$

$\therefore$ Actual area through which flow takes place,

$$a = \pi D_1 B_1 - 0.1\ \pi D_1 B_1 = 0.9 \pi D_1 B_1$$
$$= 0.9 \times \pi \times 1.0 \times 0.25 = 0.7068\ \text{m}^2$$

$\therefore$ Mass of water passing per second

$$= \rho \times a \times V_{f_1} = 1000 \times .7068 \times 2.0 = \mathbf{1413.6\ kg.\ Ans.}$$

(*i*) *The runner vane angle at inlet* (θ).

From inlet velocity triangle $\tan \alpha = \dfrac{V_{f_1}}{V_{w_1}} = \dfrac{2.0}{V_{w_1}}$

$$\therefore \quad V_{w_1} = \frac{2.0}{\tan \alpha} = \frac{2.0}{\sin 10} = 11.34 \text{ m/s}$$

$$\therefore \quad \tan \theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{2.0}{11.34 - 10.99} = 5.714$$

$$\therefore \quad \theta = \tan^{-1} 5.714 = \mathbf{80.07°} \text{ or } \mathbf{80°4.2'. Ans.}$$

(*ii*) *Velocity of wheel at inlet,* $u_1 = \mathbf{10.99 \text{ m/s. Ans.}}$

(*iii*) *The absolute velocity of water leaving the guide vanes* (V_1) :

From inlet triangle, $\sin \alpha = \dfrac{V_{f_1}}{V_1}$

$$\therefore \quad V_1 = \frac{V_{f_1}}{\sin \alpha} = \frac{2.0}{\sin 10°} = \mathbf{11.517 \text{ m/s. Ans.}}$$

(*iv*) *The relative velocity of water entering the runner blade* (V_{r_1})

$$\sin \theta = \frac{V_{f_1}}{V_{r_1}}$$

$$\therefore \quad V_{r_1} = \frac{V_{f_1}}{\sin \theta} = \frac{2.0}{\sin 80.07°} = \mathbf{2.03 \text{ m/s. Ans.}}$$

Problem 18.19 *The external and internal diameters of an inward flow reaction turbines are 1.20 m and 0.6 m respectively. The head on the turbine is 22 m and velocity of flow through the runner is constant and equal to 2.5 m/s. The guide blade angle is given as 10° and the runner vanes are radial at inlet. If the discharge at outlet is radial, determine :*

(i) The speed of the turbine, (ii) The vane angle at outlet of the runner, and

(iii) Hydraulic efficiency.

Solution. Given :

External diameter, $D_1 = 1.20$ m

Internal diameter, $D_2 = 0.60$ m

Head, $H = 22.0$ m

Velocity of flow, $V_{f_1} = V_{f_2} = 2.5$ m/s

Guide blade angle, $\alpha = 10°$

Runner vanes radial at inlet means, $\theta = 90°$

$$\therefore \quad V_{w_1} = u_1, V_{r_1} = V_{f_1} = 2.5 \text{ m/s}$$

Discharge is radial

$$\therefore \quad V_{w_2} = 0, V_2 = V_{f_2} = 2.5 \text{ m/s}$$

From inlet velocity triangle,

$$\tan \alpha = \frac{V_{f_1}}{u_1}$$

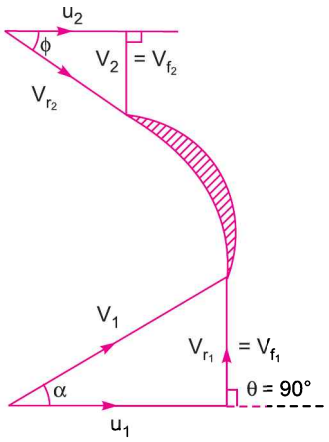


Fig. 18.16

$$\therefore \quad u_1 = \frac{V_{f_1}}{\tan \alpha} = \frac{1.5}{\tan 10^\circ} = 14.178 \text{ m/s}$$

$$\therefore \quad V_{w_1} = u_1 = 14.178 \text{ m/s}$$

(*i*) *The speed of the turbine* (N)

Using $$u_1 = \frac{\pi D_1 \times N}{60}$$

$$\therefore \quad N = \frac{60 \times u_1}{\pi D_1} = \frac{60 \times 14.178}{\pi \times 1.20} = \textbf{225.65 r.p.m. Ans.}$$

(*ii*) *The vane angle at outlet of the runner* (ϕ).

$$u_2 = \frac{\pi D_2 \times N}{60} = \frac{\pi \times 0.6 \times 225.65}{60} = 7.09 \text{ m/s.}$$

From outlet velocity triangle, $\tan \phi = \frac{V_{f_2}}{u_2} = \frac{2.5}{7.09} = 0.3526$

$$\therefore \quad \phi = \tan^{-1} 0.3526 = 19.42^\circ \text{ or } \textbf{19° 25.2′. Ans.}$$

(*iii*) *Hydraulic efficiency* is given by $\eta_h = \frac{V_{w_1} u_1}{gH} = \frac{14.178 \times 14.178}{9.81 \times 22.0} = 0.9314 = \textbf{93.14\%. Ans.}$

Problem 18.20 *233 litres of water per second are supplied to an inward flow reaction turbine. The head available is 11 m. The wheel vanes are radial at inlet and the inlet diameter is twice the outlet diameter. The velocity of flow is constant and equal to 1.83 m/s. The wheel makes 370 r.p.m. Find :*

(a) Guide vane angle, (b) Inlet and outlet diameter of the wheel,

(c) The width of the wheel at inlet and exit. Neglect the thickness of the vanes.

Assume that the discharge is radial and there are no losses in the wheel. Take speed ratio = 0.7.

Solution. Given :

Discharge, $Q = 233 \text{ lit/s} = 0.233 \text{ m}^3/\text{s}$

Head, $H = 11 \text{ m}$

Wheel vanes are radial at inlet. This means angle

$$\theta = 90^\circ \text{ and } V_{r_1} = V_{f_1}$$

Inlet diameter $= 2 \times$ Outlet diameter

$$\therefore \quad D_1 = 2D_2$$

Velocity of flow at inlet and outlet = 1.83 m/s

$$\therefore \quad V_{f_1} = V_{f_2} = 1.83 \text{ m/s}$$

Speed, $N = 370$ r.p.m.

Speed ratio $= 0.7$ or $\frac{u_1}{\sqrt{2gH}} = 0.7$

$$\therefore \quad u_1 = 0.7 \times \sqrt{2gH}$$

$$= 0.7 \times \sqrt{2 \times 9.81 \times 11} = 10.28 \text{ m/s}$$

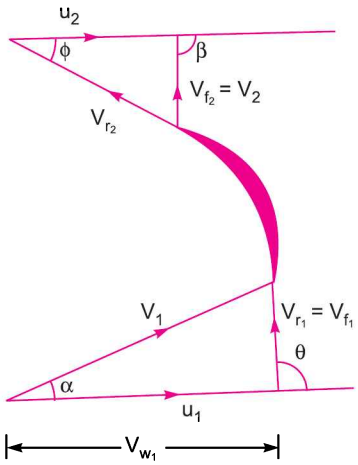


Fig. 18.17

Discharge is radial at outlet. This means angle $\beta = 90^\circ$ and $V_{w_2} = 0$

(*a*) *Guide vane angle* (α)

$$\tan \alpha = \frac{V_{f_1}}{u_1} = \frac{1.83}{10.28} = 0.178$$

$$\therefore \quad \alpha = \tan^{-1} .178 = \mathbf{10° 6'. Ans.}$$

(*b*) *Inlet and outlet diameter of the wheel*

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times D_1 \times 370}{60}$$

$$\therefore \quad D_1 = \frac{60 \times u_1}{\pi \times 370} = \frac{60 \times 10.28}{\pi \times 370} = .532 \text{ m} = \mathbf{53.2 \text{ cm. Ans.}}$$

$$D_2 = \frac{D_1}{2} = \frac{53.2}{2} = \mathbf{26.6 \text{ cm. Ans.}}$$

(*c*) *Width of wheel at inlet and outlet*

$$Q = \pi D_1 \times B_1 \times V_{f_1} = \pi D_2 \times B_2 \times V_{f_2}$$

But $V_{f_1} = V_{f_2} \quad \therefore \quad D_1 \times B_1 = D_2 \times B_2$

As $D_1 = 2D_2, B_2 = 2B_1$

Now $Q = \pi D_1 \times B_1 \times V_{f_1} = \pi \times .532 \times B_1 \times 1.83$

$$\therefore \quad B_1 = \frac{Q}{\pi \times .532 \times 1.83} = \frac{0.233}{\pi \times .532 \times 1.83} = 0.0762 \text{ m} = \mathbf{7.62 \text{ cm. Ans.}}$$

$$B_2 = 2 \times B_1 = 2 \times 7.62 = \mathbf{15.24 \text{ cm. Ans.}}$$

18.7.5 Outward Radial Flow Reaction Turbine. Fig. 18.18 shows outward radial flow reaction turbine in which the water from casing enters the stationary guide wheel. The guide wheel consists of guide

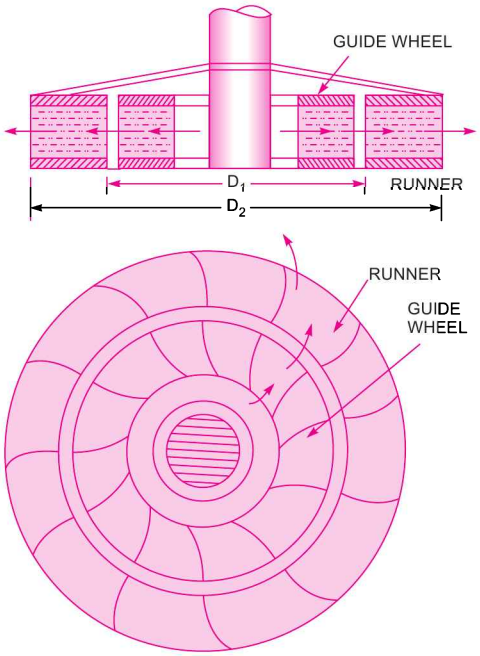


Fig. 18.18 *Outward radial flow turbine.*

vanes which direct water to enter the runner which is around the stationary guide wheel. The water flows through the vanes of the runner in the outward radial direction and is discharged at the outer diameter of the runner. The inner diameter of the runner is inlet and outer diameter is the outlet.

The velocity triangles at inlet and outlet will be drawn by the same procedure as adopted for inward flow turbine. The work done by the water on the runner per second, the horse power developed and hydraulic efficiency will be obtained from the velocity triangles. In this case as inlet of the runner is at the inner diameter of the runner, the tangential velocity at inlet will be less than that of at outlet, *i.e.*,

$$u_1 < u_2 \text{ as } D_1 < D_2.$$

Problem 18.21 *An outward flow reaction turbine has internal and external diameters of the runner as 0.6 m and 1.2 m respectively. The guide blade angle is 15° and velocity of flow through the runner is constant and equal to 4 m/s. If the speed of the turbine is 200 r.p.m., head on the turbine is 10 m and discharge at outlet is radial, determine :*

(i) The runner vane angles at inlet and outlet,

(ii) Work done by the water on the runner per second per unit weight of water striking per second ,

(iii) Hydraulic efficiency, and

(iv) The degree of reaction.

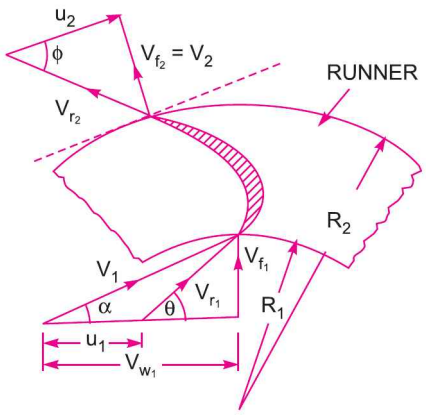


Fig. 18.19

Solution. Given :

Internal diameter,	$D_1 = 0.6$ m
External diameter,	$D_2 = 1.2$ m
Guide blade angle,	$\alpha = 15°$
Velocity of flow,	$V_{f_1} = V_{f_2} = 4$ m/s
Speed,	$N = 200$ r.p.m.
Head,	$H = 10$ m
Discharge at outlet	= Radial
$\therefore$	$V_{w_2} = 0$, $V_{f_2} = V_2$

Tangential velocity of runner at inlet and outlet are :

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 0.6 \times 200}{60} = 6.283 \text{ m/s}$$

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 1.2 \times 200}{60} = 12.566 \text{ m/s.}$$

From the inlet velocity triangle, $\tan \alpha = \dfrac{V_{f_1}}{V_{w_1}}$

$$\therefore \quad V_{w_1} = \frac{V_{f_1}}{\tan \alpha} = \frac{4.0}{\tan 15^\circ} = 14.928 \text{ m/s.}$$

(*i*) *Runner Vane Angles at inlet and outlet* are θ and ϕ

$$\tan \theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{4.0}{(14.928 - 6.283)} = 0.4627$$

$$\theta = \tan^{-1} .4627 = 24.83 \text{ or } \mathbf{24° 49.8'. Ans.}$$

From outlet velocity triangle, $\tan \phi = \frac{V_{f_2}}{u_2} = \frac{4.0}{12.566} = 0.3183$

$\therefore$ $\phi = \tan^{-1} .3183 =$ **17.65° or 17° 39.4′. Ans.**

(*ii*) *Work done by water per second per unit weight of water striking per second*

$$= \frac{1}{g} V_{w_1} u_1 \qquad (\because \ V_{w_2} = 0)$$

$$= \frac{1}{9.81} \times 14.928 \times 6.283 = \textbf{9.561 Nm/N. Ans.}$$

(*iii*) *Hydraulic efficiency* is given by equation (18.20*B*) as

$$\eta_h = \frac{V_{w_1} u_1}{gH} = \frac{14.928 \times 6.283}{9.81 \times 10} = \textbf{0.9561 or 95.61\%. Ans.}$$

(*iv*) Given : In this question, the velocity of flow is constant through the runner (*i.e.*, $V_{f_1} = V_{f_2}$) and the discharge is radial at outlet (*i.e.*, $\beta = 90°$ or $V_{w_2} = 0$), the degree of reaction (R) is given by equation (18.20*I*) as

$$R = 1 - \frac{\cot \alpha}{2 (\cot \alpha - \cot \theta)}$$

Here $\alpha = 13.928°$ and $\theta = 41.09°$ (calculated)

Substituting the value of α and θ, we get

$$R = 1 - \frac{\cot 13.928°}{2 (\cot 13.928° - \cot 41.09°)} = 1 - \frac{4.032}{2 (4.032 - 1.146)}$$

$$= 1 - 0.698 = 0.302 \simeq \textbf{0.3. Ans.}$$

For Francis turbine, the degree of reaction varies from 0 to 1 *i.e.*, $0 \le R \le 1$.

Problem 18.22 *The internal and external diameters of an outward flow reaction turbine are 2 m and 2.75 m respectively. The turbine is running at 250 r.p.m. and rate of flow of water through the turbine is 5 m^3/s. The width of the runner is constant at inlet and outlet and is equal to 250 mm. The head on the turbine is 150 m. Neglecting thickness of the vanes and taking discharge radial at outlet determine :*

(i) Vane angles at inlet and outlet, and

(ii) Velocity of flow at inlet and outlet.

Solution. Given :

Internal diameter,	$D_1 = 2.0$ m
External diameter,	$D_2 = 2.75$ m
Speed of turbine,	$N = 250$ r.p.m.
Discharge,	$Q = 5$ m^3/s
Width at inlet and outlet,	$B_1 = B_2 = 250$ mm $= 0.25$ m
Head,	$H = 150$ m
Discharge at outlet	= radial

$\therefore$ $V_{w_2} = 0$ and $V_{f_2} = V_2$.

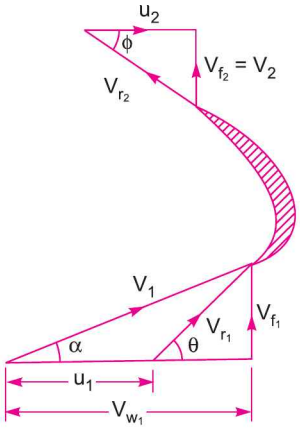


Fig. 18.20

The tangential velocity of the turbine at inlet and outlet,

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 2.0 \times 250}{60} = 26.18 \text{ m/s}$$

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 2.75 \times 250}{60} = 36.0 \text{ m/s.}$$

The discharge through turbine is given by equation (18.21) as

$$Q = \pi D_1 B_1 V_{f_1} = \pi D_2 B_2 V_{f_2}$$

$$\therefore \quad V_{f_1} = \frac{Q}{\pi D_1 B_1} = \frac{5.0}{\pi \times 2.0 \times .25} = 3.183 \text{ m/s.}$$

And $$V_{f_2} = \frac{Q}{\pi D_2 B_2} = \frac{5.0}{\pi \times 2.75 \times .25} = 2.315 \text{ m/s}$$

Using equation (18.24), $H - \frac{V_2^2}{2g} = \frac{V_{w_1} u_1}{g}$ $\left(\because \ V_{w_2} = 0\right)$

But for radial discharge, $V_2 = V_{f_2} = 2.315$ m/s

$$\therefore \quad 150 - \frac{2.315^2}{2 \times 9.81} = \frac{V_{w_2} \times 26.18}{9.81} \text{ or } 149.73 = \frac{V_{w_1} \times 26.18}{9.81}$$

$$\therefore \quad V_{w_1} = \frac{149.73 \times 9.81}{26.18} = 56.1 \text{ m/s.}$$

(*i*) *Vane angles at inlet and outlet*

From inlet velocity triangle, $\tan \theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{3.183}{56.1 - 26.18} = 0.1064$

$$\therefore \quad \theta = \tan^{-1} .1064 = \mathbf{6.072°} \text{ or } \mathbf{6° \ 4.32'. \ Ans.}$$

From outlet velocity triangle, $\tan \phi = \frac{V_{f_2}}{u_2} = \frac{2.315}{36.0} = 0.0643$

$$\therefore \quad \phi = \tan^{-1} .0643 = \mathbf{3.68°} \text{ or } \mathbf{3° \ 40.8'. \ Ans.}$$

(*ii*) *Velocity of flow at inlet and outlet*

$$V_{f_1} = 3.183 \text{ m/s and } V_{f_2} = \mathbf{2.315 \text{ m/s. Ans.}}$$

▶ 18.8 FRANCIS TURBINE

The inward flow reaction turbine having radial discharge at outlet is known as Francis Turbine, after the name of J.B. Francis, an American engineer who in the beginning designed inward radial flow reaction type of turbine. In the modern Francis turbine, the water enters the runner of the turbine in the radial direction at outlet and leaves in the axial direction at the inlet of the runner. Thus the modern Francis Turbine is a mixed flow type turbine.

The velocity triangle at inlet and outlet of the Francis turbine are drawn in the same way as in case of inward flow reaction turbine. As in case of Francis turbine, the discharge is radial at outlet, the velocity of whirl at outlet (*i.e.*, V_{w_2}) will be zero. Hence the work done by water on the runner per second will be

$$= \rho Q[V_{w_1} u_1]$$

And work done per second per unit weight of water striking/s = $\frac{1}{g}\left[V_{w_1} u_1\right]$

Hydraulic efficiency will be given by, $\eta_h = \frac{V_{w_1} u_1}{gH}$.

18.8.1 Important Relations for Francis Turbines. The following are the important relations for Francis Turbines :

1. The ratio of width of the wheel to its diameter is given as $n = \frac{B_1}{D_1}$. The value of n varies from 0.10 to .40.

2. The flow ratio is given as,

Flow ratio = $\frac{V_{f_1}}{\sqrt{2gH}}$ and varies from 0.15 to 0.30.

3. The speed ratio = $\frac{u_1}{\sqrt{2gH}}$ varies from 0.6 to 0.9.

Problem 18.23 *A Francis turbine with an overall efficiency of 75% is required to produce 148.25 kW power. It is working under a head of 7.62 m. The peripheral velocity = 0.26* $\sqrt{2gH}$ *and the radial velocity of flow at inlet is 0.96* $\sqrt{2gH}$. *The wheel runs at 150 r.p.m. and the hydraulic losses in the turbine are 22% of the available energy. Assuming radial discharge, determine :*

(i) The guide blade angle, *(ii) The wheel vane angle at inlet,*
(iii) Diameter of the wheel at inlet, and *(iv) Width of the wheel at inlet.*

Solution. Given :

Overall efficiency $\eta_o = 75\% = 0.75$

Power produced, S.P. = 148.25 kW

Head, $H = 7.62$ m

Peripheral velocity, $u_1 = 0.26\sqrt{2gH} = 0.26 \times \sqrt{2 \times 9.81 \times 7.62} = 3.179$ m/s

Velocity of flow at inlet, $V_{f_1} = 0.96\sqrt{2gH} = 0.96 \times \sqrt{2 \times 9.81 \times 7.62} = 11.738$ m/s.

Speed, $N = 150$ r.p.m.

Hydraulic losses = 22% of available energy

Discharge at outlet = Radial

$V_{w_2} = 0$ and $V_{f_2} = V_2$

Hydraulic efficiency is given as

$$\eta_h = \frac{\text{Total head at inlet} - \text{Hydraulic loss}}{\text{Head at inlet}}$$

$$= \frac{H - .22\,H}{H} = \frac{0.78\,H}{H} = 0.78$$

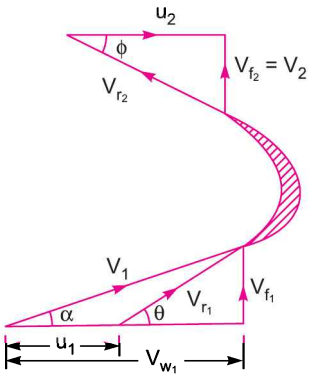


Fig. 18.21

But $$\eta_h = \frac{V_{w_1} u_1}{gH}$$

$$\therefore \quad \frac{V_{w_1} u_1}{gH} = 0.78$$

$$\therefore \quad V_{w_1} = \frac{0.78 \times g \times H}{u_1}$$

$$= \frac{0.78 \times 9.81 \times 7.62}{3.179} = 18.34 \text{ m/s.}$$

(*i*) *The guide blade angle, i.e.*, α. From inlet velocity triangle,

$$\tan \alpha = \frac{V_{f_1}}{V_{w_1}} = \frac{11.738}{18.34} = 0.64$$

$$\therefore \quad \alpha = \tan^{-1} 0.64 = \mathbf{32.619°} \text{ or } \mathbf{32° 37'. Ans.}$$

(*ii*) *The wheel vane angle at inlet, i.e.*, θ

$$\tan \theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{11.738}{18.34 - 3.179} = 0.774$$

$$\therefore \quad \theta = \tan^{-1} .774 = \mathbf{37.74} \text{ or } \mathbf{37° 44.4'. Ans.}$$

(*iii*) *Diameter of wheel at inlet* (D_1).

Using the relation, $$u_1 = \frac{\pi D_1 N}{60}$$

$$D_1 = \frac{60 \times u_1}{\pi \times N} = \frac{60 \times 3.179}{\pi \times 50} = \mathbf{0.4047 \text{ m. Ans.}}$$

(*iv*) *Width of the wheel at inlet* (B_1)

$$\eta_o = \frac{\text{S.P.}}{\text{W.P.}} = \frac{148.25}{\text{W.P.}}$$

But $$\text{W.P.} = \frac{\text{WH}}{1000} = \frac{\rho \times g \times Q \times H}{1000} = \frac{1000 \times 9.81 \times Q \times 7.62}{1000}$$

$$\therefore \quad \eta_o = \frac{148.25}{\frac{1000 \times 9.81 \times Q \times 7.62}{1000}} = \frac{148.25 \times 1000}{1000 \times 9.81 \times Q \times 7.62}$$

or $$Q = \frac{148.25 \times 1000}{1000 \times 9.81 \times 7.62 \times \eta_o} = \frac{148.25 \times 1000}{1000 \times 9.81 \times 7.62 \times 0.75} = 2.644 \text{ m}^3/\text{s}$$

Using equation (18.21), $$Q = \pi D_1 \times B_1 \times V_{f_1}$$

$\therefore \quad 2.644 = \pi \times .4047 \times B_1 \times 11.738$

$\therefore \quad B_1 = \dfrac{2.644}{\pi \times .4047 \times 11.738} = \mathbf{0.177\ m.\ Ans.}$

Problem 18.24 *The following data is given for a Francis Turbine. Net head H = 60 m ; Speed N = 700 r.p.m.; shaft power = 294.3 kW ; η_o = 84% ; η_h = 93%; flow ratio = 0.20 ; breadth ratio n = 0.1; Outer diameter of the runner = 2 × inner diameter of runner. The thickness of vanes occupy 5% of circumferential area of the runner, velocity of flow is constant at inlet and outlet and discharge is radial at outlet. Determine :*

(i) Guide blade angle, *(ii) Runner vane angles at inlet and outlet,*

(iii) Diameters of runner at inlet and outlet, and *(iv) Width of wheel at inlet.*

Solution. Given :

Net head, $H = 60$ m

Speed, $N = 700$ r.p.m.

Shaft power $= 294.3$ kW

Overall efficiency, $\eta_o = 84\% = 0.84$

Hydraulic efficiency, $\eta_h = 93\% = 0.93$

Flow ratio, $\dfrac{V_{f_1}}{\sqrt{2gH}} = 0.20$

$\therefore \quad V_{f_1} = 0.20 \times \sqrt{2gH}$

$= 0.20 \times \sqrt{2 \times 9.81 \times 60} = 6.862$ m/s

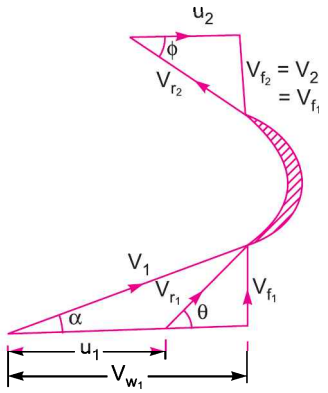


Fig. 18.22

Breadth ratio, $\dfrac{B_1}{D_1} = 0.1$

Outer diameter, $D_1 = 2 \times$ Inner diameter $= 2 \times D_2$

Velocity of flow, $V_{f_1} = V_{f_2} = 6.862$ m/s.

Thickness of vanes $=$ 5% of circumferential area of runner

$\therefore$ Actual area of flow $= 0.95\ \pi D_1 \times B_1$

Discharge at outlet $=$ Radial

$\therefore \quad V_{w_2} = 0$ and $V_{f_2} = V_2$

Using relation, $\eta_o = \dfrac{\text{S.P.}}{\text{W.P.}}$

$0.84 = \dfrac{294.3}{\text{W.P.}}$

$\therefore \quad \text{W.P.} = \dfrac{294.3}{0.84} = 350.357$ kW.

But $\text{W.P.} = \dfrac{WH}{1000} = \dfrac{\rho \times g \times Q \times H}{1000} = \dfrac{1000 \times 9.81 \times Q \times 60}{1000}$

$\therefore \quad \dfrac{1000 \times 9.81 \times Q \times 60}{1000} = 350.357$

$$\therefore \quad Q = \frac{350.357 \times 1000}{60 \times 1000 \times 9.81} = 0.5952 \text{ m}^3/\text{s}.$$

Using equation (18.21), Q = Actual area of flow × Velocity of flow

$$= 0.95\ \pi D_1 \times B_1 \times V_{f_1}$$

$$= 0.95 \times \pi \times D_1 \times 0.1 D_1 \times V_{f_2} \qquad (\because \quad B_1 = 0.1\ D_1)$$

or $$0.5952 = 0.95 \times \pi \times D_1 \times 0.1 \times D_1 \times 6.862 = 2.048\ D_1^2$$

$$\therefore \quad D_1 = \sqrt{\frac{0.5952}{2.048}} = 0.54 \text{ m}$$

But $$\frac{B_1}{D_1} = 0.1$$

$$\therefore \quad B_1 = 0.1 \times D_1 = 0.1 \times .54 = .054 \text{ m} = 54 \text{ mm}$$

Tangential speed of the runner at inlet,

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 0.54 \times 700}{60} = 19.79 \text{ m/s}.$$

Using relation for hydraulic efficiency,

$$\eta_h = \frac{V_{w_1} u_1}{gH} \text{ or } 0.93 = \frac{V_{w_1} \times 19.79}{9.81 \times 60}$$

$$\therefore \quad V_{w_1} = \frac{0.93 \times 9.81 \times 60}{19.79} = 27.66 \text{ m/s}.$$

(*i*) *Guide blade angle* (α)

From inlet velocity triangle, $\tan \alpha = \frac{V_{f_1}}{V_{w_1}} = \frac{6.862}{27.66} = 0.248$

$$\therefore \quad \alpha = \tan^{-1} 0.248 = 13.928° \text{ or } \mathbf{13° 55.7'. Ans.}$$

(*ii*) *Runner vane angles at inlet and outlet (θ and ϕ)*

$$\tan \theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{6.862}{27.66 - 19.79} = 0.872$$

$$\therefore \quad \theta = \tan^{-1} 0.872 = 41.09° \text{ or } \mathbf{41° 5.4'. Ans.}$$

From outlet velocity triangle, $\tan \phi = \frac{V_{f_2}}{u_2} = \frac{V_{f_1}}{u_2} = \frac{6.862}{u_2}$ ...(*i*)

But $$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times D_1}{2} \times \frac{N}{60} \qquad \left(\because \quad D_2 = \frac{D_1}{2} \text{ given}\right)$$

$$= \pi \times \frac{.54}{2} \times \frac{700}{60} = 9.896 \text{ m/s}.$$

Substituting the value of u_2 in equation (*i*),

$$\tan \phi = \frac{6.862}{9.896} = 0.6934$$

$\therefore$ $\phi = \tan^{-1} .6934° = 34.74$ or **34° 44.4′. Ans.**

(*iii*) *Diameters of runner at inlet and outlet*

$D_1 = 0.54$ m, $D_2 = \mathbf{0.27\ m.}$ **Ans.**

(*iv*) *Width of wheel at inlet*

$B_1 = \mathbf{54\ mm.}$ **Ans.**

Problem 18.24 (A) *For the above problem, find the degree of reaction for the given Francis Turbine.*

Solution. Given :

In this question, the velocity of flow is constant through the runner (*i.e.*, $V_{f_1} = V_{f_2}$) and the discharge is radial at outlet (*i.e.*, $\beta = 90°$ or $V_{w_2} = 0$), the degree of reaction (R) is given by equation (18.20 *I*) as

$$R = 1 - \frac{\cot \alpha}{2(\cot \alpha - \cot \theta)}$$

Here $\alpha = 13.928°$ and $\theta = 41.09°$ (calculated)

Substituting the value of α and θ, we get

$$R = 1 - \frac{\cot 13.928°}{2(\cot 13.928° - \cot 41.09°)} = 1 - \frac{4.032}{2(4.032 - 1.146)}$$

$$= 1 - 0.698 = 0.302 \simeq \mathbf{0.3.}$$ **Ans.**

For Francis Turbine, the degree of reaction varies from 0 to 1 *i.e.*, $0 \le R \le 1$.

Problem 18.25 *(a) Show that the hydraulic efficiency for a Francis turbine having velocity of flow through runner as constant, is given by the relation.*

$$\eta_h = \frac{1}{1 + \dfrac{\frac{1}{2}\tan^2 \alpha}{\left(1 - \dfrac{\tan \alpha}{\tan \theta}\right)}}$$

where α *= Guide blade angle and* θ *= Runner vane angle at inlet.*

The turbine is having radial discharge at outlet.

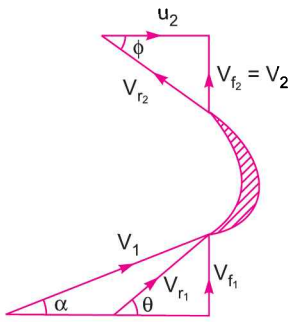


Fig. 18.23

(b) If vanes are radial at inlet, then show $\eta_h = \dfrac{2}{2 + \tan^2 \alpha}$.

Solution. Given :

Velocity of flow = Constant.

$\therefore$ $V_{f_1} = V_{f_2}$

Discharge is radial at outlet.

$\therefore$ $V_{w_2} = 0$ and $V_{f_2} = V_2$

(*a*) From the inlet velocity triangle,

$$\tan\alpha = \frac{V_{f_1}}{V_{w_1}} \quad \therefore \quad V_{f_1} = V_{w_1}\tan\alpha \qquad ...(i)$$

Also $$\tan\theta = \frac{V_{f_1}}{V_{w_1} - u_1}$$

or $$\left(V_{w_1} - u_1\right) = \frac{V_{f_1}}{\tan\theta} = \frac{V_{w_1}\tan\alpha}{\tan\theta} \qquad \left(\because \quad V_{f_1} = V_{w_1}\tan\alpha\right)$$

$$\therefore \qquad u_1 = V_{w_1} - \frac{V_{w_1}\tan\alpha}{\tan\theta} = V_{w_1}\left(1 - \frac{\tan\alpha}{\tan\theta}\right) \qquad ...(ii)$$

Using equation (18.24), we have

$$H - \frac{V_2^2}{2g} = \frac{1}{g}(V_{w_1}u_1) \qquad \left(\because \quad V_{w_2} = 0\right)$$

$$\therefore \qquad H = \frac{1}{g}V_{w_1}u_1 + \frac{V_2^2}{2g} = \frac{1}{g}V_{w_1}u_1 + \frac{V_{f_1}^2}{2g} \qquad \left(\because \quad V_2 = V_{f_2} = V_{f_1}\right)$$

Substituting the values of V_{f_1} and u_1 from equations (*i*) and (*ii*),

$$H = \frac{1}{g}V_{w_1} \times V_{w_1}\left(1 - \frac{\tan\alpha}{\tan\theta}\right) + \frac{\left[V_{w_1}\tan\alpha\right]^2}{2g}$$

$$= \frac{V_{w_1}^2}{g}\left(1 - \frac{\tan\alpha}{\tan\theta}\right) + \frac{V_{w_1}^2}{2g}\tan^2\alpha = \frac{V_{w_1}^2}{g}\left[1 - \frac{\tan\alpha}{\tan\theta} + \frac{\tan^2\alpha}{2}\right].$$

Now, hydraulic efficiency is given by

$$\eta_h = \frac{V_{w_1}u_1}{gH} = \frac{V_{w_1}u_1}{g \times \frac{V_{w_1}^2}{g}\left[1 - \frac{\tan\alpha}{\tan\theta} + \frac{\tan^2\alpha}{2}\right]}$$

$$= \frac{V_{w_1} \times V_{w_1}\left(1 - \frac{\tan\alpha}{\tan\theta}\right)}{V_{w_1}^2\left[1 - \frac{\tan\alpha}{\tan\theta} + \frac{\tan^2\alpha}{2}\right]} \qquad \left[\because \quad u_1 = V_{w_1}\left(1 - \frac{\tan\alpha}{\tan\theta}\right)\right]$$

$$= \frac{\left(1 - \frac{\tan\alpha}{\tan\theta}\right)}{\left[1 - \frac{\tan\alpha}{\tan\theta} + \frac{\tan^2\alpha}{2}\right]} = \frac{1}{1 + \frac{\frac{1}{2}\tan^2\alpha}{\left(1 - \frac{\tan\alpha}{\tan\theta}\right)}}.$$ **Ans.**

(*b*) If vanes are radial at inlet, then $\theta = 90°$

$$\therefore \quad \eta_h = \frac{1}{1+\dfrac{\frac{1}{2}\tan^2\alpha}{\left(1-\dfrac{\tan\alpha}{\tan 90°}\right)}} = \frac{1}{1+\dfrac{\frac{1}{2}\tan^2\alpha}{(1-0)}} = \frac{2}{2+\tan^2\alpha}. \textbf{ Ans.}$$

Problem 18.26 *A Francis turbine working under a head of 30 m has a wheel diameter of 1.2 m at the entrance and 0.6 m at the exit. The vane angle at the entrance is 90° and guide blade angle is 15°. The water at the exit leaves the vanes without any tangential velocity and the velocity of flow in the runner is constant. Neglecting the effect of draft tube and losses in the guide and runner passages, determine the speed of wheel in r.p.m. and vane angle at the exit. State whether the speed calculated is synchronous or not. If not, what speed would you recommend to couple the turbine with an alternator of 50 cycles ?*

Solution. Given :

Head on turbine, $H = 30$ m
Inlet dia., $D_1 = 1.2$ m
Outlet dia., $D_2 = 0.6$ m
Vane angle at inlet, $\theta = 90°$
Guide blade angle, $\alpha = 15°$

The water at exit leaves the vanes without any tangential velocity.

$\therefore \quad V_{w_2} = 0$ and $V_2 = V_{f_2}$

Velocity of flow is constant in runner.

$\therefore \quad V_{f_1} = V_{f_2}$

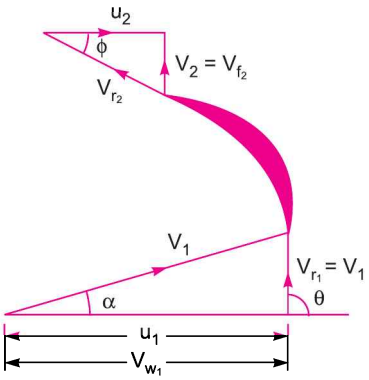


Fig. 18.24

(*i*) *Speed of turbine in r.p.m.*

Using equation (18.24), we have

$$H - \frac{V_2^2}{2g} = \frac{1}{g}\left(V_{w_1}u_1 \pm V_{w_2}u_2\right)$$

$$= \frac{1}{g}\left(V_{w_1} \times u_1\right) \qquad \left(\because \ V_{w_2} = 0\right)$$

$$= \frac{1}{g}u_1 \times u_1 \qquad \left(\because \ V_{w_1} = u_1\right)$$

or $$30 - \frac{V_{f_2}^2}{2g} = \frac{1}{g}u_1^2 \qquad \left(\because \ V_2 = V_{f_2} = V_{f_1}\right) \ ...(i)$$

But from inlet velocity triangle, we have

$$\tan\alpha = \frac{V_{f_1}}{u_1} \text{ or } u_1 = \frac{V_{f_1}}{\tan\alpha} = \frac{V_{f_1}}{\tan 15°} = 3.732\,V_{f_1} \qquad ...(ii)$$

Substituting the values of u_1 in equation (*i*), we get

$$30 - \frac{V_{f_2}^2}{2g} = \frac{1}{g} \times (3.732\ V_{f_1})^2 \text{ or } 30 - \frac{V_{f_1}^2}{2g} = \frac{13.928\,V_{f_2}^2}{g} \qquad \left(\because \ V_{f_2} = V_{f_1}\right)$$

or $$30 = \frac{14.928\, V_{f_1}^2}{g}$$

$$\therefore \quad V_{f_1} = \sqrt{\frac{30 \times 9.81}{14.928}} = 4.44 \text{ m/s.}$$

Substituting the value of V_{f_1} in equation (*ii*), we get

$$u_1 = 3.732 \times 4.44 = 16.57 \text{ m/s}$$

But $$u_1 = \frac{\pi D_1 N}{60} \text{ or } 16.57 = \frac{\pi \times 1.2 \times N}{60}$$

$$\therefore \quad N = \frac{16.57 \times 60}{\pi \times 1.2} = \textbf{263.72 r.p.m. Ans.}$$

(*ii*) *Vane angle at exit* (*i.e.*, ϕ)

$$u_2 = \frac{\pi D_2 \times N}{60} = \frac{\pi \times 0.6 \times 263.72}{60} = 8.285 \text{ m/s}$$

$$V_{f_2} = V_{f_1} = 4.44$$

Now, from velocity triangle at outlet,

$$\tan \phi = \frac{V_{f_2}}{u_2} = \frac{4.44}{8.285} = 0.5359$$

$$\therefore \quad \phi = \textbf{28.187°. Ans.}$$

(*iii*) For a turbine, which is directly coupled to the alternator of 50 cycles, the synchronous speed (N^*) is given by $$f = \frac{p.N^*}{60}$$

where f = Frequency of alternator in cycles/s, p = Number of pair of poles for the alternator.

Assuming the number of pair of poles = 12, we get

$$50 = \frac{12 \times N^*}{60}$$

$$\therefore \quad N^* = \frac{60 \times 50}{12} = 250 \text{ r.p.m.}$$

But the speed of turbine is 263.72. And synchronous speed (N^*) is equal to 250. Hence, the speed of turbine is not synchronous. The speed of turbine should be 250 r.p.m.

▶ 18.9 AXIAL FLOW REACTION TURBINE

If the water flows parallel to the axis of the rotation of the shaft, the turbine is known as axial flow turbine. And if the head at the inlet of the turbine is the sum of pressure energy and kinetic energy and during the flow of water through runner a part of pressure energy is converted into kinetic energy, the turbine is known as reaction turbine.

For the axial flow reaction turbine, the shaft of the turbine is vertical. The lower end of the shaft is made larger which is known as 'hub' or 'boss'. The vanes are fixed on the hub and hence hub acts as a runner for axial flow reaction turbine. The following are the important type of axial flow reaction turbines :

1. Propeller Turbine, and
2. Kaplan Turbine.

When the vanes are fixed to the hub and they are not adjustable, the turbine is known as propeller turbine. But if the vanes on the hub are adjustable, the turbine is known as a *Kaplan Turbine*, after the name of V Kaplan, an Austrian Engineer. This turbine is suitable where a large quantity of water at low head is available. Fig. 18.25 shows the runner of a Kaplan turbine, which consists of a hub fixed to the shaft. On the hub, the adjustable vanes are fixed as shown in Fig. 18.25.

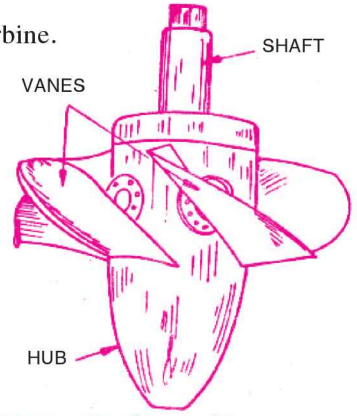


Fig. 18.25 *Kaplan turbine runner.*

The main parts of a Kaplan turbine are :

1. Scroll casing,
2. Guide vanes mechanism,
3. Hub with vanes or runner of the turbine, and
4. Draft tube.

Fig. 18.26 shows all main parts of a Kaplan turbine. The water from penstock enters the scroll casing and then moves to the guide vanes. From the guide vanes, the water turns through 90° and flows axially through the runner as shown in Fig. 18.26. The discharge through the runner is obtained as

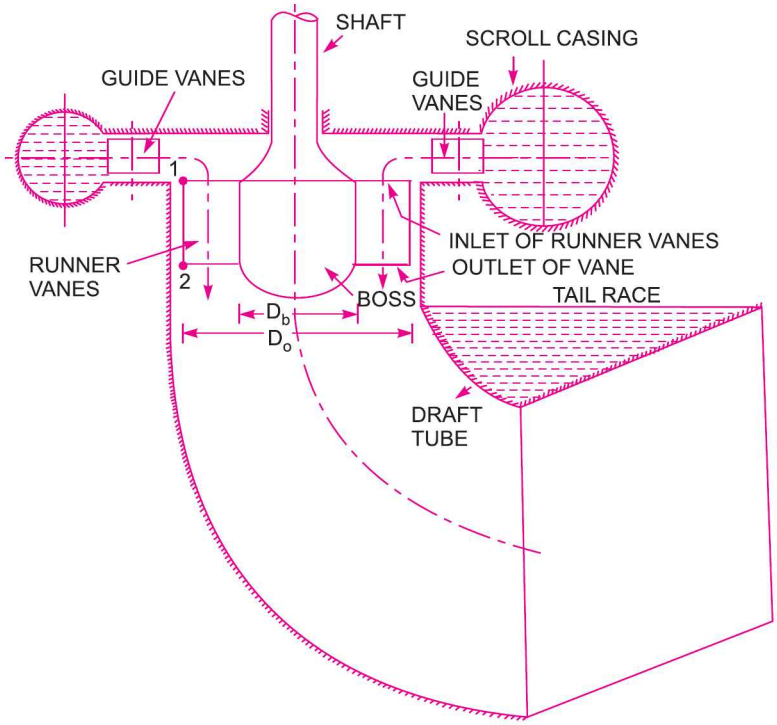


Fig. 18.26 *Main components of Kaplan turbine.*

$$Q = \frac{\pi}{4}\left(D_o^2 - D_b^3\right) \times V_{f_1} \qquad ...(18.25)$$

where D_o = Outer diameter of the runner,

D_b = Diameter of hub, and

V_{f_1} = Velocity of flow at inlet.

The inlet and outlet velocity triangles are drawn at the extreme edge of the runner vane corresponding to the points 1 and 2 as shown in Fig. 18.26.

18.9.1 Some Important Point for Propeller (Kaplan Turbine). The following are the important points for propeller or Kaplan turbine :

1. The peripheral velocity at inlet and outlet are equal

$$\therefore \qquad u_1 = u_2 = \frac{\pi D_o N}{60}, \text{ where } D_o = \text{Outer dia. of runner}$$

2. Velocity of flow at inlet and outlet are equal

$$\therefore \qquad V_{f_1} = V_{f_2}.$$

3. Area of flow at inlet = Area of flow at outlet

$$= \frac{\pi}{4}\left(D_o^2 - D_b^2\right).$$

Problem 18.27 *A Kaplan turbine working under a head of 20 m develops 11772 kW shaft power. The outer diameter of the runner is 3.5 m and hub diameter is 1.75 m. The guide blade angle at the extreme edge of the runner is 35°. The hydraulic and overall efficiencies of the turbines are 88% and 84% respectively. If the velocity of whirl is zero at outlet, determine :*

(i) Runner vane angles at inlet and outlet at the extreme edge of the runner, and

(ii) Speed of the turbine.

Solution. Given :

Head,	$H = 20$ m
Shaft power,	S.P. = 11772 kW
Outer dia. of runner,	$D_o = 3.5$ m
Hub diameter,	$D_b = 1.75$ m
Guide blade angle,	$\alpha = 35°$
Hydraulic efficiency,	$\eta_h = 88\%$
Overall efficiency,	$\eta_o = 84\%$
Velocity of whirl at outlet	$= 0$.

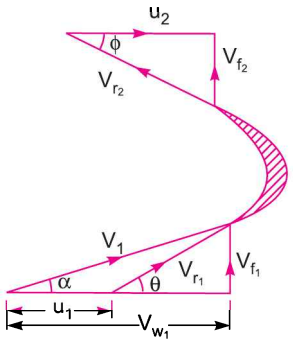


Fig. 18.27

Using the relation, $\eta_o = \dfrac{\text{S.P.}}{\text{W.P.}}$

where $\text{W.P.} = \dfrac{\text{W.P.}}{1000} = \dfrac{\rho \times g \times Q \times H}{1000}$, we get

$$0.84 = \frac{11772}{\dfrac{\rho \times g \times Q \times H}{1000}}$$

$$= \frac{11772 \times 1000}{1000 \times 9.81 \times Q \times 20} \qquad (\because \ \rho = 1000)$$

$$\therefore \quad Q = \frac{11772 \times 1000}{0.84 \times 1000 \times 9.81 \times 20} = 71.428 \text{ m}^3/\text{s}.$$

Using equation (18.25), $Q = \frac{\pi}{4}\left(D_o^2 - D_b^2\right) \times V_{f_1}$

or $$71.428 = \frac{\pi}{4}(3.5^2 - 1.75^2) \times V_{f_1} = \frac{\pi}{4}(12.25 - 3.0625)\ V_{f_1}$$

$$= 7.216\ V_{f_1}$$

$$\therefore \quad V_{f_1} = \frac{71.428}{7.216} = 9.9 \text{ m/s}.$$

From inlet velocity triangle, $\tan \alpha = \frac{V_{f_2}}{V_{w_1}}$

$$\therefore \quad V_{w_1} = \frac{V_{f_1}}{\tan \alpha} = \frac{9.9}{\tan 35^\circ} = \frac{9.9}{.7} = 14.14 \text{ m/s}$$

Using the relation for hydraulic efficiency,

$$\eta_h = \frac{V_{w_1} u_1}{gH} \qquad (\because \ V_{w_2} = 0)$$

$$0.88 = \frac{14.14 \times u_1}{9.81 \times 20}$$

$$\therefore \quad u_1 = \frac{0.88 \times 9.81 \times 20}{14.14} = 12.21 \text{ m/s}.$$

(*i*) Runner vane angles at inlet and outlet at the extreme edge of the runner are given as :

$$\tan \theta = \frac{V_{f_1}}{V_{w_1} - u_1} = \frac{9.9}{(14.14 - 12.21)} = 5.13$$

$$\therefore \quad \theta = \tan^{-1} 5.13 = \mathbf{78.97^\circ} \text{ or } \mathbf{78^\circ\ 58'}. \ \mathbf{Ans.}$$

For Kaplan turbine, $u_1 = u_2 = 12.21$ m/s and $V_{f_1} = V_{f_2} = 9.9$ m/s

$\therefore$ From outlet velocity triangle, $\tan \phi = \frac{V_{f_2}}{u_2} = \frac{9.9}{12.21} = 0.811$

$$\therefore \quad \phi = \tan^{-1}.811 = \mathbf{39.035^\circ} \text{ or } \mathbf{39^\circ\ 2'}. \ \mathbf{Ans.}$$

(*ii*) Speed of turbine is given by $u_1 = u_2 = \frac{\pi D_o N}{60}$

$$12.21 = \frac{\pi \times 3.5 \times N}{60}$$

$$\therefore \quad N = \frac{60 \times 12.21}{\pi \times 3.50} = \mathbf{66.63 \text{ r.p.m.}} \ \mathbf{Ans.}$$

Problem 18.28 *A Kaplan turbine develops 24647.6 kW power at an average head of 39 metres. Assuming a speed ratio of 2, flow ratio of 0.6, diameter of the boss equal to 0.35 times the diameter of the runner and an overall efficiency of 90%, calculate the diameter, speed and specific speed of the turbine.*

Solution. Given :

Shaft power, S.P. = 24647.6 kW

Head, $H = 39$ m

Speed ratio, $u_1\sqrt{2gH} = 2.0$

$\therefore \quad u_1 = 2.0 \times \sqrt{2gH} = 2.0 \times \sqrt{2 \times 9.81 \times 39} = 55.32$ m/s

Flow ratio, $\dfrac{V_{f_1}}{\sqrt{2gH}} = 0.6$

$\therefore \quad V_{f_1} = 0.6 \times \sqrt{2gH} = 0.6 \times \sqrt{2 \times 9.81 \times 39} = 16.59$ m/s

Diameter of boss = 0.35 × Diameter of runner

$\therefore \quad D_b = 0.35 \times D_o$

Overall efficiency, $\eta_o = 90\% = 0.90$

Using the relation, $\eta_o = \dfrac{\text{S.P.}}{\text{W.P.}}$, where W.P. $= \dfrac{\rho \times g \times Q \times H}{1000}$

$\therefore \quad 0.90 = \dfrac{24647.6}{\dfrac{\rho \times g \times Q \times H}{1000}} = \dfrac{24647.6 \times 1000}{1000 \times 9.81 \times Q \times 39}$

$\therefore \quad Q = \dfrac{24647.6 \times 1000}{0.9 \times 1000 \times 9.81 \times 39} = 71.58 \text{ m}^3/\text{s}.$

But from equation (18.25), we have

$$Q = \frac{\pi}{4}\left(D_o^2 - D_b^2\right) \times V_{f_1}$$

$\therefore \quad 71.58 = \dfrac{\pi}{4}\left[D_o^2 - (.35\, D_o)^2\right] \times 16.59 \qquad \left(\because \quad D_b = 0.35\, D_o,\ V_{f_1} = 16.59\right)$

$$= \frac{\pi}{4}\left[D_o^2 - .1225\, D_o^2\right] \times 16.59$$

$$= \frac{\pi}{4} \times .8775\, D_o^2 \times 16.59 = 11.433\, D_o^2$$

(*i*) $\therefore \quad D_o = \sqrt{\dfrac{71.58}{11.433}} =$ **2.5 m. Ans.**

$\therefore \quad D_b = 0.35 \times D_o = 0.35 \times 2.5 =$ **0.875 m. Ans.**

(*ii*) Speed of the turbine is given by $u_1 = \dfrac{\pi D_o N}{60}$

$\therefore \quad 55.32 = \dfrac{\pi \times 2.5 \times N}{60}$

$$\therefore \quad N = \frac{60 \times 55.32}{\pi \times 2.5} = \textbf{422.61 r.p.m. Ans.}$$

(*iii*) Specific speed * is given by $N_s = \frac{N\sqrt{P}}{H^{5/4}}$, where P = Shaft power in kW

$$\therefore \quad N_s = \frac{422.61 \times \sqrt{24647.6}}{(39)^{5/4}} = \frac{422.61 \times 156.99}{97.461} = \textbf{680.76 r.p.m. Ans.}$$

Problem 18.29 *A Kaplan turbine runner is to be designed to develop 9100 kW. The net available head is 5.6 m. If the speed ratio = 2.09, flow ratio = 0.68, overall efficiency = 86% and the diameter of the boss is 1/3 the diameter of the runner. Find the diameter of the runner, its speed and the specific speed of the turbine.*

Solution. Given :

Power, $P = 9100$ kW

Net head, $H = 5.6$ m

Speed ratio $= 2.09$

Flow ratio $= 0.68$

Overall efficiency, $\eta_o = 86\% = 0.86$

Diameter of boss $= \frac{1}{3}$ of diameter of runner

or $$D_b = \frac{1}{3} D_o$$

Now, speed ratio $$= \frac{u_1}{\sqrt{2gH}}$$

$$\therefore \quad u_1 = 2.09 \times \sqrt{2 \times 9.81 \times 5.6} = 21.95 \text{ m/s}$$

Flow ratio $$= \frac{V_{f_1}}{\sqrt{2gH}}$$

$$\therefore \quad V_{f_1} = 0.68 \times \sqrt{2 \times 9.81 \times 5.6} = 7.12 \text{ m/s}$$

The overall efficiency is given by, $\eta_o = \dfrac{P}{\left(\dfrac{\rho \times g.Q.H}{1000}\right)}$

or $$Q = \frac{P \times 1000}{\rho \times g \times H \times \eta_o} = \frac{9100 \times 1000}{1000 \times 9.81 \times 5.6 \times 0.86}$$

$$\left(\because \quad \rho g = 1000 \times 9.81 \text{ N/m}^3\right)$$

$$= 192.5 \text{ m}^3/\text{s}.$$

The discharge through a Kaplan turbine is given by

$$Q = \frac{\pi}{4}[D_o^2 - D_b^2] \times V_{f_1}$$

* For the definition and derivation, please refer to page 920 Arts. 18.11 and 18.11.1.

or $$192.5 = \frac{\pi}{4}\left[D_o^2 - \left(\frac{D_o}{3}\right)^2\right] \times 7.12 \qquad \left(\because\ D_b = \frac{D_o}{3}\right)$$

$$= \frac{\pi}{4}\left[1 - \frac{1}{9}\right] D_o^2 \times 7.12$$

$$\therefore \quad D_o = \sqrt{\frac{4 \times 192.5 \times 9}{\pi \times 8 \times 7.12}} = \mathbf{6.21\ m.\ Ans.}$$

The speed of turbine is given by, $u_1 = \frac{\pi DN}{60}$

$$\therefore \quad N = \frac{60 \times u_1}{\pi \times D} = \frac{60 \times 21.95}{\pi \times 6.21} = \mathbf{67.5\ r.p.m.\ Ans.}$$

The specific speed is given by, $N_s = \frac{N\sqrt{P}}{H^{5/4}} = \frac{67.5 \times \sqrt{9100}}{5.6^{5/4}} = \mathbf{746.\ Ans.}$

Problem 18.30 *The hub diameter of a Kaplan turbine, working under a head of 12 m, is 0.35 times the diameter of the runner. The turbine is running at 100 r.p.m. If the vane angle of the extreme edge of the runner at outlet is 15° and flow ratio is 0.6, find :*

(i) Diameter of the runner, (ii) Diameter of the boss, and

(iii) Discharge through the runner.

The velocity of whirl at outlet is given as zero.

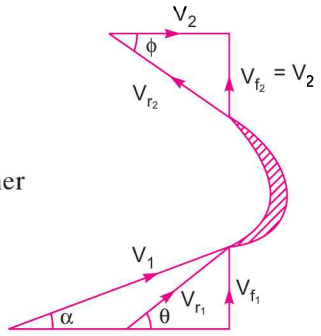


Fig. 18.28

Solution. Given :

Head, $H = 12$ m

Hub diameter, $D_b = 0.35 \times D_o$, where D_o = Dia. of runner

Speed, $N = 100$ r.p.m.

Vane angle at outlet, $\phi = 15°$

Flow ratio $= \frac{V_{f_1}}{\sqrt{2gH}} = 0.6$

$$\therefore \quad V_{f_1} = 0.6 \times \sqrt{2gH} = 0.6 \times \sqrt{2 \times 9.81 \times 12}$$

$$= 9.2 \text{ m/s.}$$

From the outlet velocity triangle, $V_{w_2} = 0$

$$\tan\phi = \frac{V_{f_2}}{u_2} = \frac{V_{f_1}}{u_2} \qquad \left(\because\ V_{f_2} = V_{f_1} = 9.2\right)$$

$$\therefore \quad \tan 15° = \frac{9.2}{u_2}$$

$$\therefore \quad u_2 = \frac{9.2}{\tan 15°} = 34.33 \text{ m/s.}$$

But for Kaplan turbine, $u_1 = u_2 = 34.33$

Now, using the relation, $u_1 = \frac{\pi D_o \times N}{60}$ or $34.33 = \frac{\pi \times D_o \times 100}{60}$

$$D_o = \frac{60 \times 34.33}{\pi \times 100} = \textbf{6.55 m. Ans.}$$

$\therefore$ $D_b = 0.35 \times D_o = 0.35 \times 6.35 = \textbf{2.3 m. Ans.}$

Discharge through turbine is given by equation (18.25) as

$$Q = \frac{\pi}{4}\left[D_o^2 - D_b^2\right] \times V_{f_1} = \frac{\pi}{4}\left[6.55^2 - 2.3^2\right] \times 9.2$$

$$= \frac{\pi}{4}(42.9026 - 5.29) \times 9.2 = \textbf{271.77 m}^3\textbf{/s. Ans.}$$

Problem 18.31 *A Kaplan turbine runner is to be designed to develop 7357.5 kW shaft power. The net available head is 5.50 m. Assume that the speed ratio is 2.09 and flow ratio is 0.68, and the overall efficiency is 60%. The diameter of the boss is* $\frac{1}{3}$*rd of the diameter of the runner. Find the diameter of the runner, its speed and its specific speed.*

Solution. Given :

Shaft power, $P = 7357.5$ kW

Head, $H = 5.50$ m

Speed ratio $= \dfrac{u_1}{\sqrt{2gH}} = 2.09$

$\therefore$ $u_1 = 2.09 \times \sqrt{2 \times 9.81 \times 5.50} = 21.71$ m/s

Flow ratio $= \dfrac{V_{f_1}}{\sqrt{2gH}} = 0.68$

$\therefore$ $V_{f_1} = 2.68 \times \sqrt{2 \times 9.81 \times 5.50} = 7.064$ m/s

Overall efficiency, $\eta_o = 60\% = 0.60$

Diameter of boss, $D_b = \dfrac{1}{3} \times D_o$

Using relation, $\eta_o = \dfrac{\text{Shaft power}}{\text{Water power}} = \dfrac{7357.5}{\dfrac{\rho \times g \times Q \times H}{1000}}$

or $0.60 = \dfrac{7357.5 \times 1000}{\rho \times g \times Q \times H} = \dfrac{7357.5 \times 1000}{1000 \times 9.81 \times Q \times 5.5}$

$\therefore$ $Q = \dfrac{7357.5 \times 1000}{1000 \times 9.81 \times 5.5 \times 0.60} = 227.27$ m^3/s.

Using equation (18.25) for discharge,

$$Q = \frac{\pi}{4}\left(D_o^2 - D_b^2\right) \times V_{f_1}$$

or $$227.27 = \frac{\pi}{4}\left[D_o^2 - \left(\frac{D_o}{3}\right)^2\right] \times 7.064 \qquad \left(\because\ D_b = \frac{D_o}{3}\right)$$

$$= \frac{\pi}{4} \times \frac{8}{9} D_o^2 \times 7.064 = 4.9316\ D_o^2$$

$\therefore$ $$D_o = \sqrt{\frac{227.27}{4.9316}} = \mathbf{6.788\ m.\ Ans.}$$

And $$D_b = \frac{1}{3} \times 6.788 = \mathbf{2.262\ m.\ Ans.}$$

Using the relation, $$u_1 = \frac{\pi D_o \times N}{60}$$

$\therefore$ $$N = \frac{60 \times u_1}{\pi D_o} = \frac{60 \times 21.71}{\pi \times 6.788} = \mathbf{61.08\ r.p.m.\ Ans.}$$

The specific speed (N_s) is given by,

$$N_s = \frac{N\sqrt{P}}{H^{5/4}} = \frac{61.08 \times \sqrt{7357.5}}{5.50^{5/4}} = \mathbf{622\ r.p.m.\ Ans.}$$

Problem 18.32 *In a tidal power plant, a bulb turbine (which is basically an axial flow turbine) operates a 5 MW generator at 150 r.p.m. under a head of 5.5 m. The generator efficiency is 93% and the overall efficiency of the turbine is 88%. The tip diameter of the runner is 4.5 m and hub diameter is 2 m. Assuming hydraulic efficiency of 94% and no exit whirl, determine the runner vane angles at inlet and exit at the mean diameter of the vanes.*

Solution. Given :

Output of generator $= 5$ M* W $= 5 \times 10^6$ W
Speed of turbine, $N = 150$ r.p.m.
Head on turbine, $H = 5.5$ m
Generator efficiency, $\eta_g = 93\%$ or 0.93
Overall efficiency of the turbine, $\eta_o = 88\%$ or 0.88
Tip dia. of runner, $D_o = 4.5$ m
Hub dia. of runner, $D_b = 2$ m
Hydraulic efficiency, $\eta_h = 94\%$ or 0.94

No exit whirl means the velocity of whirl at outlet is zero *i.e.*, $V_{w_2} = 0$. And hence angle $\beta = 90°$ as shown in Fig. 18.29.

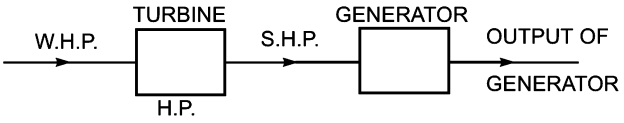


Fig. 18.29

Now, generator efficiency is given by,

$$\eta_g = \frac{\text{Output of generator}}{\text{Input of generator}}$$

$$= \frac{\text{Output of generator}}{\text{Output of turbine}} \quad (\because \text{Output of turbine} = \text{Input of generator})$$

or $$0.93 = \frac{5 \times 10^6}{\text{S.P.}}$$

$\therefore$ $$\text{S.P.} = \frac{5 \times 10^6}{0.93} \text{W} \qquad ...(i)$$

* MW stands for Mega Watt which is equal to 10^6 Watt or 10^6 W.

Now, overall efficiency of turbine is given by, $\eta_o = \dfrac{\text{S.P.}}{\text{W.P.}}$

But W.P. in Watt $= \rho \times Q \times H \times 9.81 = 1000 \times Q \times 5.5 \times 9.81$ W

$\therefore \quad \eta_o = \dfrac{\text{S.P.}}{1000 \times Q \times 5.5 \times 9.81}$

$\therefore \quad \text{S.P.} = \eta_o \times 1000 \times Q \times 5.5 \times 9.81$ ...(*ii*)

Equating the two values of S.P. given by equations (*i*) and (*ii*), we get

$$\frac{5 \times 10^6}{0.93} = 0.88 \times 1000 \times Q \times 5.5 \times 9.81 \qquad (\because \eta_o = 0.88)$$

$$\therefore \quad Q = \frac{5 \times 10^6}{0.93 \times 0.88 \times 1000 \times 5.5 \times 9.81} = 113.23 \text{ m}^3/\text{s}$$

The vane angles are to be calculated at the mean diameter of the runner.

$\therefore$ Mean diameter, $D_m = \dfrac{D_o + D_b}{2} = \dfrac{4.5 + 2.0}{2} = 3.25$ m

Inlet vane velocity corresponding to mean dia. is given by,

$$u_1 = \frac{\pi D_m \times N}{60} = \frac{\pi \times 3.25 \times 150}{60} = 25.52 \text{ m/s}$$

For axial flow turbine, $u_1 = u_2 = 25.52$ m/s and $V_{f_1} = V_{f_2}$

For no whirl velocity at outlet, the hydraulic efficiency is given by,

$$\eta_h = \frac{V_{w_2} \times u_1}{g \times H} \text{ or } 0.94 = \frac{V_{w_1} \times 25.52}{9.81 \times 5.5}$$

$$\therefore \quad V_{w_1} = \frac{0.94 \times 9.81 \times 5.5}{25.52} = 1.987 \text{ m/s}$$

This value of V_{w_1} is less than u_1. Hence, the velocity triangle at inlet will be as shown in Fig. 18.30.

Now, using equation (18.25), we get

$$Q = \frac{\pi}{4}\left(D_o^2 - D_b^2\right) \times V_{f_1}$$

or $$113.23 = \frac{\pi}{4}(4.5^2 - 2^2) \times V_{f_1} = \frac{\pi}{4} \times 16.25 \times V_{f_1}$$

$$\therefore \quad V_{f_1} = \frac{113.23 \times 4}{\pi \times 16.25} = 8.87 \text{ m/s}$$

$$\therefore \quad V_{f_2} = V_{f_1} = 8.87 \text{ m/s.}$$

Let θ = Runner vane angle at inlet and

ϕ = Runner vane angle at outlet.

From inlet velocity triangle,

$$\tan\theta = \frac{V_{f_1}}{(u_1 - V_{w_1})} = \frac{8.87}{25.52 - 1.987}$$

$$= 0.3769$$

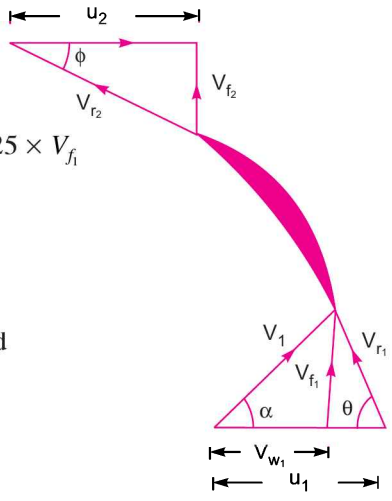


Fig. 18.30

$\therefore \quad \theta = \tan^{-1} 0.3769 = \mathbf{20.65°}$. **Ans.**

Now, from outlet velocity triangle,

$$\tan \phi = \frac{V_{f_2}}{u_2} = \frac{8.87}{25.52} \qquad \left(\because \ V_{f_2} = V_{f_1} \text{ and } u_1 = u_2\right)$$

$$= 0.347$$

$\therefore \quad \phi = \tan^{-1} 0.347 = \mathbf{19.16°}$. **Ans.**

Problem 18.33 *A propeller reaction turbine of runner diameter 4.5 m is running at 40 r.p.m. The guide blade angle at inlet is 145° and runner blade angle at outlet is 25° to the direction of vane. The axial flow area of water through runner is 25 m^2. If the runner blade angle at inlet is radial determine :*

(i) Hydraulic efficiency of the turbine, (ii) Discharge through turbine,

(iii) Power developed by the turbine, and (iv) Specific speed of the turbine.

Solution. Given :

Runner diameter, $D_o = 4.5$ m

Speed, $N = 40$ r.p.m.

Guide blade angle, $\alpha = 145°$

Runner blade angle at outlet, $\phi = 25°$

Flow area, $a = 25$ m^2

Runner blade angle at inlet is radial

$\therefore \quad \theta = 90°$, $V_{r_1} = V_{f_1}$ and $u_1 = V_{w_1}$

For Kaplan turbine, the discharge is given by the product of area of flow and velocity of flow.

As area of flow is constant and hence $V_{f_1} = V_{f_2}$ ($\because$ Q = Area of flow $\times$ V_{f_1} = Area of flow $\times$ V_{f_2})

The tangential speed of turbine at inlet,

$$u_1 = \frac{\pi D_o N}{60} = \frac{\pi \times 4.5 \times 40}{60}$$

$$= 9.42 \text{ m/s}$$

Also $u_2 = u_1 = 9.42$ m/s.

From inlet velocity triangle,

$$\tan (180° - \alpha) = \frac{V_{f_1}}{u_1}$$

or $$\tan (180° - 145°) = \tan 35° = \frac{V_{f_1}}{u_1}$$

$\therefore \quad V_{f_1} = u_1 \tan 35° = 9.42 \tan 35° = 6.59$

Also $V_{w_1} = u_1 = 9.42$ m/s.

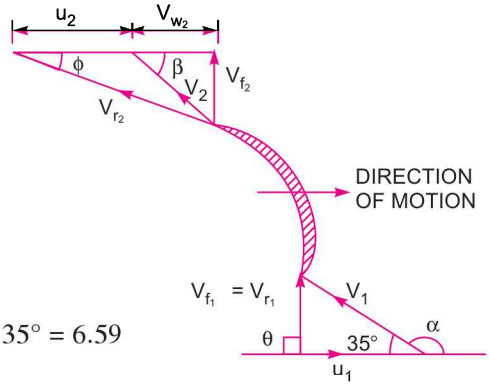


Fig. 18.31

From outlet velocity triangle,

$$\tan \phi = \frac{V_{f_2}}{u_2 + V_{w_2}} \qquad \text{(where } V_{f_2} = V_{f_1} = 6.59 \text{ and } u_2 = u_1 = 9.42\text{)}$$

$\therefore \quad \tan 25° = \dfrac{6.59}{9.42 + V_{w_2}}$

$$\therefore \quad V_{w_2} + 9.42 = \frac{6.59}{\tan 25^\circ} = 14.13$$

$$\therefore \quad V_{w_2} = 14.13 - 9.42 = 4.71 \text{ m/s}$$

$$\therefore \quad V_2 = \sqrt{V_{f_2}^2 + V_{w_2}^2} = \sqrt{6.59^2 + 4.71^2} = \sqrt{43.43 + 22.18} = 8.1 \text{ m/s.}$$

Using equation (18.24),

$$H - \frac{V_2^2}{2g} = \frac{1}{g}\left[V_{w_1}u_1 - V_{w_2}u_2\right].$$

Here –ve sign is taken as the absolute velocity at inlet and outlet (*i.e.*, V_1 and V_2) are in the same direction and hence change of velocity will be with a –ve sign

$$\therefore \quad H - \frac{8.1^2}{2 \times 9.81} = \frac{1}{9.81}\,[9.42 \times 9.42 - 4.71 \times 9.42]$$

$$H - 3.344 = \frac{1}{9.81}\,[88.736 - 44.368] = 4.522 \text{ m}$$

$$\therefore \quad H = 4.522 + 3.344 = 7.866 \text{ m.}$$

(*i*) Hydraulic efficiency is given by equation (18.20*A*) as

$$\eta_h = \frac{V_{w_1}u_1 - V_{w_2}u_2}{g \times H}$$

$$= \frac{(9.42 \times 9.42 - 4.71 \times 9.42)}{9.81 \times 7.866} = 0.575 = \mathbf{57.5\%.\ Ans.}$$

(*ii*) Discharge through turbine is given by,

$$Q = \text{Area of flow} \times \text{Velocity of flow}$$

$$= 25 \times V_{f_1} = 25 \times 6.59 = \mathbf{164.75\ m^3/s.\ Ans.}$$

(*iii*) Power developed by turbine $= \dfrac{\text{Work done per second}}{1000}$

$$= \frac{1}{g}\frac{\left[V_{w_1}u_1 - V_{w_2}u_2\right]}{1000} \times \text{Weight of water}$$

$$= \frac{1}{9.81}\left[\frac{9.42 \times 9.42 - 4.71 \times 9.42}{1000}\right] \times \rho \times g \times Q$$

$$= \frac{1}{9.81}\left[\frac{9.42 \times 9.42 - 4.71 \times 9.42}{1000}\right] \times 1000 \times 9.81 \times 164.75$$

$$= \mathbf{6867\ kW.\ Ans.}$$

(*iv*) Specific speed is given by the relation,

$$N_s = \frac{N\sqrt{P}}{H^{5/4}} = \frac{N\sqrt{6867}}{7.866^{5/4}} = \frac{40 \times \sqrt{6867}}{7.866^{5/4}}$$

$$= \frac{40 \times 82.867}{13.173} = \mathbf{251.62\ r.p.m.\ Ans.}$$

▶ 18.10 DRAFT-TUBE

The draft-tube is a pipe of gradually increasing area which connects the outlet of the runner to the tail race. It is used for discharging water from the exit of the turbine to the tail race. This pipe of gradually increasing area is called a draft-tube. One end of the draft-tube is connected to the outlet of the runnner while the other end is sub-merged below the level of water in the tail race. The draft-tube, in addition to serve a passage for water discharge, has the following two purposes also :

1. It permits a negative head to be established at the outlet of the runner and thereby increase the net head on the turbine. The turbine may be placed above the tail race without any loss of net head and hence turbine may be inspected properly.

2. It converts a large proportion of the kinetic energy $(V_2^2/2g)$ rejected at the outlet of the turbine into useful pressure energy. Without the draft tube, the kinetic energy rejected at the outlet of the turbine will go waste to the tail race.

Hence by using draft-tube, the net head on the turbine increases. The turbine develops more power and also the efficiency of the turbine increases.

If a reaction turbine is not fitted with a draft-tube, the pressure at the outlet of the runner will be equal to atmospheric pressure. The water from the outlet of the runner will discharge freely into the tail race. The net head on the turbine will be less than that of a reaction turbine fitted with a draft-tube. Also without a draft-tube, the kinetic energy $\left(\frac{V_2^2}{2g}\right)$ rejected at the outlet of the runner will go waste to the tail race.

18.10.1 Types of Draft-Tubes. The following are the important types of draft-tubes which are commonly used :

1. Conical draft-tubes,
2. Simple elbow tubes,
3. Moody spreading tubes, and
4. Elbow draft-tubes with circular inlet and rectangular outlet.

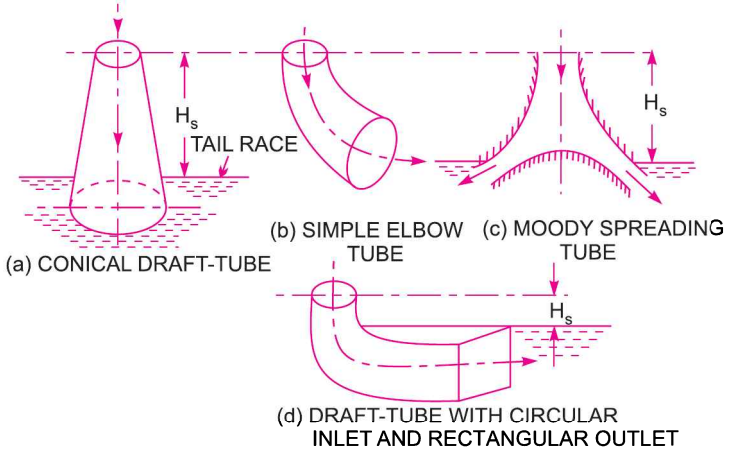


Fig. 18.32 *Types of draft-tubes.*

These different types of draft-tubes are shown in Fig. 18.32. The conical draft-tubes and Moody spreading draft-tubes are most efficient while simple elbow tubes and elbow draft-tubes with circular inlet and rectangular outlet require less space as compared to other draft-tubes.

18.10.2 Draft-Tube Theory. Consider a capital draft-tube as shown in Fig. 18.33.

Let H_s = Vertical height of draft-tube above the tail race,

y = Distance of bottom of draft-tube from tail race.

Applying Bernoulli's equation to inlet (section 1-1) and outlet (section 2-2) of the draft-tube and taking section 2-2 as the datum line,we get

$$\frac{p_1}{\rho g}+\frac{V_1^2}{2g}+(H_s+y)=\frac{p_2}{\rho g}+\frac{V_2^2}{2g}+0+h_f \qquad ...(i)$$

where h_f = loss of energy between sections 1-1 and 2-2.

But $\frac{p_2}{\rho g}$ = Atmospheric pressure head + y

$$=\frac{p_a}{\rho g}+y.$$

Substituting this value of $\frac{p_2}{\rho g}$ in equation (i), we get

$$\frac{p_1}{\rho g}+\frac{V_1^2}{2g}+(H_s+y)=\frac{p_a}{\rho g}+y+\frac{V_2^2}{2g}+h_f$$

or $$\frac{p_1}{\rho g}+\frac{V_1^2}{2g}+H_s=\frac{p_a}{\rho g}+\frac{V_2^2}{2g}+h_f$$

$$\therefore \quad \frac{p_1}{\rho g}=\frac{p_a}{\rho g}+\frac{V_2^2}{2g}+h_f-\frac{V_1^2}{2g}-H_s$$

$$=\frac{p_a}{\rho g}-H_s-\left(\frac{V_1^2}{2g}-\frac{V_2^2}{2g}-h_f\right) \qquad ...(18.26)$$

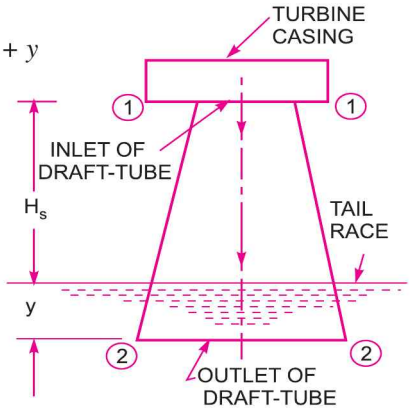


Fig. 18.33 *Draft-tube theory.*

In equation (18.26), $\frac{p_1}{\rho g}$ is less than atmospheric pressure.

18.10.3 Efficiency of Draft-Tube. The efficiency of a draft-tube is defined as the ratio of actual conversion of kinetic head into pressure head in the draft-tube to the kinetic head at the inlet of the draft-tube. Mathematically, it is written as

$$\eta_d=\frac{\text{Actual conversion of kinetic head into pressure head}}{\text{Kinetic head at the inlet of draft-tube}}$$

Let V_1 = Velocity of water at inlet of draft-tube,

V_2 = Velocity of water at outlet of draft-tube, and

h_f = Loss of head in the draft-tube.

Theoretical conversion of kinetic head into pressure head in draft-tube = $\left(\frac{V_1^2}{2g}-\frac{V_2^2}{2g}\right)$.

Actual conversion of kinetic head into pressure head = $\left(\frac{V_1^2}{2g}-\frac{V_2^2}{2g}\right)-h_f$

$$\therefore \qquad \eta_d = \frac{\left(\frac{V_1^2}{2g} - \frac{V_2^2}{2g}\right) - h_f}{\left(\frac{V_1^2}{2g}\right)} \qquad ...(18.27)$$

Problem 18.33 (A) *A water turbine has a velocity of 6 m/s at the entrance to the draft-tube and a velocity of 1.2 m/s at the exit. For friction losses of 0.1 m and a tail water 5 m below the entrance to the draft-tube, find the pressure head at the entrance.*

Solution. Given :

Velocity at inlet, $V_1 = 6$ m/s

Velocity at outlet, $V_2 = 1.2$ m/s

Friction loss, $h_f = 0.1$ m

Vertical height between tail race and inlet of draft-tube = 5 m

Let y = Vertical height between tail race and outlet of draft-tube.

Applying Bernoulli's equation at the inlet and outlet of draft-tube and taking reference line passing through section (2-2), we get

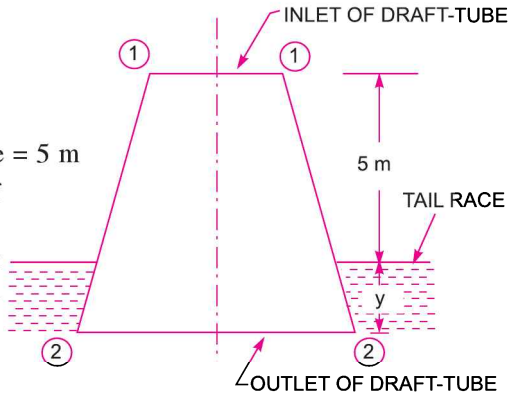


Fig. 18.33 (*a*)

$$\frac{p_1}{\rho g} + \frac{V_1^2}{2g} + Z_1 = \frac{p_2}{\rho g} + \frac{V_2^2}{2g} + Z_2 + h_f$$

where $Z_1 = (5 + y)$; $V_1 = 6$ m/s ; $V_2 = 1.2$ m/s,

$h_f = 0.1$

$$\frac{p_2}{\rho g} = \text{Atmospheric pressure head} + y = \frac{p_a}{\rho g} + y$$

$$Z_2 = 0$$

Substituting the values, we get

$$\frac{p_1}{\rho g} + \frac{6^2}{2 \times 9.81} + (5 + y) = \left(\frac{p_a}{\rho g} + y\right) + \frac{1.2^2}{2 \times 9.81} + 0 + 0.1$$

or $$\frac{p_1}{\rho g} + 1.835 + 5 + y = \frac{p_a}{\rho g} + y + 0.0734 + 0.1$$

or $$\frac{p_1}{\rho g} + 6.835 = \frac{p_a}{\rho g} + 0.1734 \qquad ...(i)$$

If $\frac{p_a}{\rho g}$ (*i.e.*, atmospheric pressure head) is taken zero, then we will get $\frac{p_1}{\rho g}$ as vacuum pressure head at inlet of draft-tube.

But if $\frac{p_a}{\rho g} = 10.3$ m of water, then we will get $\frac{p_1}{\rho g}$ as absolute pressure head at inlet of draft-tube.

Taking $\frac{p_a}{\rho g} = 0$ and substituting this value in equation (*i*), we get

$$\frac{p_1}{\rho g} + 6.835 = 0 + 0.1734$$

$$\therefore \quad \frac{p_1}{\rho g} = -6.835 + 0.1734 = \mathbf{-6.6616\ m.\ Ans.}$$

Negative sign means vacuum pressure head.

Problem 18.34 *A conical draft-tube having diameter at the top as 2.0 m and pressure head at 7 m of water (vacuum), discharges water at the outlet with a velocity of 1.2 m/s at the rate of 25 m^3/s. If atmospheric pressure head is 10.3 m of water and losses between the inlet and outlet of the draft-tubes are negligible, find the length of draft-tube immersed in water. Total length of tube is 5 m.*

Solution. Given :

Diameter at top, $D_1 = 2.0$ m

Pressure head, $\frac{p_1}{\rho g} = 7$ m (Vacuum)

$= 10.3 - 7.0 = 3.3$ m (abs.)

Velocity at outlet, $V_2 = 1.2$ m/s

Discharge, $Q = 25\ m^3/s$

Loss of energy, h_f = Negligible

Let the length of the tube immersed in water = y m.

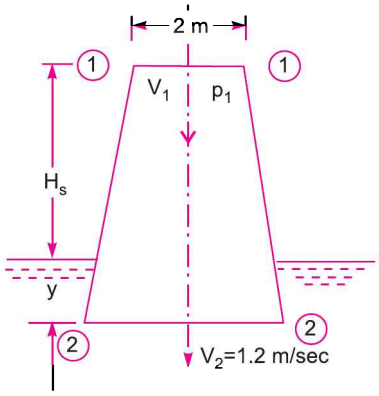


Fig. 18.34

Total length of the tube = 5 m

The velocity at inlet, $V_1 = \frac{\text{Discharge}}{\text{Area at inlet}}$

$$= \frac{Q}{\frac{\pi}{4}D_1^2} = \frac{25}{\frac{\pi}{4}(2.0)^2} = 7.957 \text{ m/s.}$$

Using equation (18.26), we have

$$\frac{p_1}{\rho g} = \frac{p_a}{\rho g} - H_s - \left(\frac{V_1^2}{2g} - \frac{V_2^2}{2g} - h_f\right)$$

$$3.30 = 10.3 - H_s - \left(\frac{7.957^2}{2\times 9.81} - \frac{1.2^2}{2\times 9.81} - 0\right)$$

$$\left(\because \ h_f = 0 \text{ and } \frac{p_a}{\rho g} = 10.3\right)$$

$$= 10.3 - H_s - (3.227 - .0734)$$

or $3.3 = 10.3 - H_s - 3.1536$

$\therefore \quad H_s = 10.3 - 3.1536 - 3.3 = 3.8464$ m

$\therefore \quad y$ = Total length $- H_s = 5 - 3.8464 =$ **1.1536 m. Ans.**

Problem 18.35 *A conical draft-tube having inlet and outlet diameters 1 m and 1.5 m discharges water at outlet with a velocity of 2.5 m/s. The total length of the draft-tube is 6 m and 1.20 m of the length of draft-tube is immersed in water . If the atmospheric pressure head is 10.3 m of water and loss of head due to friction in the draft-tube is equal to 0.2 × velocity head at outlet of the tube, find :*

(i) Pressure head at inlet, and (ii) Efficiency of the draft-tube.

Solution. Given :

Diameter at inlet, $D_1 = 1.0$ m

Diameter at outlet $D_2 = 1.5$ m

Velocity at outlet, $V_2 = 2.5$ m/s

Total length of tube, $H_s + y = 6.0$ m

Length of tube in water, $y = 1.20$ m

$\therefore$ $H_s = 6.0 - 1.20 = 4.80$ m

Atmospheric pressure head, $\frac{p_a}{\rho g} = 10.3$ m

Loss of head due to friction, $h_f = 0.2 \times$ Velocity head at outlet

$$= 0.2 \times \frac{V_2^2}{2g}$$

Discharge through tube, $Q = A_2 V_2 = \frac{\pi}{4} D_2^2 \times 2.5 = \frac{\pi}{4} (1.5)^2 \times 2.5 = 4.4178 \text{ m}^3/\text{s}$

Velocity at inlet, $V_1 = \frac{Q}{A_1} = \frac{4.4178}{\frac{\pi}{4} \times 1^2} = 5.625$ m/s

(*i*) *Pressure head at inlet* $\left(\frac{p_1}{\rho g}\right)$.

Using equation (18.26), $\frac{p_1}{\rho g} = \frac{p_a}{\rho g} - H_s - \left(\frac{V_1^2}{2g} - \frac{V_2^2}{2g} - h_f\right)$

$$= 10.3 - 4.8 - \left(\frac{5.625^2}{2 \times 9.81} - \frac{2.5^2}{2 \times 9.81} - 0.2 \times \frac{V_2^2}{2g}\right)$$

$$= 10.3 - 4.8 - \left(1.6126 - .3185 - \frac{0.2 \times 2.5^2}{2 \times 9.81}\right)$$

$= 10.3 - 4.8 - (1.6126 - .3185 - .0637) = 5.5 - (1.2304) = 4.269$

$\simeq$ **4.27 m (abs.) Ans.**

(*ii*) *Efficiency of Draft-tube* (η_d)

Using equation (18.27), $\eta_d = \frac{\left(\frac{V_1^2}{2g} - \frac{V_2^2}{2g}\right) - h_f}{\frac{V_1^2}{2g}} = \frac{\frac{V_1^2}{2g} - \frac{V_2^2}{2g} - \frac{0.2\, V_2^2}{2g}}{\frac{V_1^2}{2g}}$

$$= \frac{V_1^2 - 1.2\, V_2^2}{V_1^2} = 1 - 1.2 \left(\frac{V_2}{V_1}\right)^2 = 1 - 1.2 \left(\frac{2.5}{5.625}\right)^2 = 1 - 0.237$$

$=$ **0.763** or **76.3%. Ans.**

▶ 18.11 SPECIFIC SPEED

It is defined as the speed of a turbine which is identical in shape, geometrical dimensions, blade angles, gate opening etc., with the actual turbine but of such a size that it will develop unit power when working under unit head. It is denoted by the symbol N_s. The specific speed is used in comparing the different types of turbines as every type of turbine has different specific speed.

In M.K.S. units, unit power is taken as one horse power and unit head as one metre. But in S.I. units, unit power is taken as one kilowatt and unit head as one metre.

18.11.1 Derivation of the Specific Speed. The overall efficiency (η_o) of any turbine is given by,

$$\eta_o = \frac{\text{Shaft power}}{\text{Water power}} = \frac{\text{Power developed}}{\frac{\rho \times g \times Q \times H}{1000}} = \frac{P}{\frac{\rho \times g \times Q \times H}{1000}} \quad ...(i)$$

where H = Head under which the turbine is working,

Q = Discharge through turbine,

P = Power developed or shaft power.

From equation (i), $$P = \eta_o \times \frac{\rho \times g \times Q \times H}{1000}$$

$$\propto Q \times H \text{ (as } \eta_o \text{ and } \rho \text{ are constant)} \quad ...(ii)$$

Now let D = Diameter of actual turbine,

N = Speed of actual turbine,

u = Tangential velocity of the turbine,

N_s = Specific speed of the turbine,

V = Absolute velocity of water.

The absolute velocity, tangential velocity and head on the turbine are related as,

$$u \propto V, \text{ where } V \propto \sqrt{H}$$

$$\propto \sqrt{H} \quad ...(iii)$$

But the tangential velocity u is given by

$$u = \frac{\pi DN}{60}$$

$$\propto DN \quad ...(iv)$$

$\therefore$ From equations (iii) and (iv), we have

$$\sqrt{H} \propto DN \text{ or } D \propto \frac{\sqrt{H}}{N} \quad ...(v)$$

The discharge through turbine is given by

$$Q = \text{Area} \times \text{Velocity}$$

But $$\text{Area} \propto B \times D \quad \text{(where } B = \text{Width)}$$

$$\propto D^2 \quad (\because \ B \propto D)$$

And $$\text{Velocity} \propto \sqrt{H}$$

$\therefore$ $$Q \propto D^2 \times \sqrt{H}$$

$$\propto \left(\frac{\sqrt{H}}{N}\right)^2 \times \sqrt{H} \qquad \left(\because \text{ From equation } (v),\ D \propto \frac{\sqrt{H}}{N}\right)$$

$$\propto \frac{H}{N^2} \times \sqrt{H} \propto \frac{H^{3/2}}{N^2} \qquad ...(vi)$$

Substituting the value of Q in equation (ii), we get

$$P \propto \frac{H^{3/2}}{N^2} \times H \propto \frac{H^{5/2}}{N^2}$$

$$\therefore \qquad P = K\,\frac{H^{5/2}}{N^2}, \text{ where } K = \text{Constant of proportionality.}$$

If $P = 1$, $H = 1$, the speed N = Specific speed N_s. Substituting these values in the above equation, we get

$$1 = \frac{K \times 1^{5/2}}{N_s^2} \qquad \text{or} \qquad N_s^2 = \text{K}$$

$$\therefore \qquad P = N_s^2\,\frac{H^{5/2}}{N^2} \qquad \text{or} \qquad N_s^2 = \frac{N^2 P}{H^{5/2}}$$

$$\therefore \qquad N_s = \sqrt{\frac{N^2 P}{H^{5/2}}} = \frac{N\sqrt{P}}{H^{5/4}} \qquad ...(18.28)$$

In equation (18.28), if P is taken in metric horse power the specific speed is obtained in M.K.S. units. But if P is taken in kilowatts, the specific speed is obtained in S.I. units.

18.11.2 Significance of Specific Speed. Specific speed plays an important role for selecting the type of the turbine. Also the performance of a turbine can be predicted by knowing the specific speed of the turbine. The type of turbine for different specific speed is given in Table 18.1 as :

Table 18.1

S. No.	*Specific speed*		*Types of turbine*
	(M.K.S.)	*(S.I.)*	
1.	10 to 35	8.5 to 30	Pelton wheel with single jet
2.	35 to 60	30 to 51	Pelton wheel with two or more jets
3.	60 to 300	51 to 225	Francis turbine
4.	300 to 1000	255 to 860	Kaplan or Propeller turbine

Problem 18.36 *A turbine develops 7225 kW power under a head of 25 metres at 135 r.p.m. Calculate the specific speed of the turbine and state the type of the turbine.*

Solution. Given :

Power developed, $P = 7225$ kW

Head, $H = 25$ m

Speed, $N = 135$ r.p.m.

Specific speed of the turbine (N_s)

Using equation (18.28),

$$N_s = \frac{N\sqrt{P}}{H^{5/4}} = \frac{135 \times \sqrt{7225}}{25^{5/4}} = \mathbf{205.28.\ Ans.}$$

From Table 18.1, for specific speeds (S.I.) between 51 and 255 the type of turbine is Francis. As the specific speed 205.28 lies in this range and hence type of turbine is Francis. **Ans.**

Problem 18.37 *A turbine is to operate under a head of 25 m at 200 r.p.m. The discharge is 9 cumec. If the efficiency is 90%, determine :*

(i) Specific speed of the machine, (ii) Power generated, and

(iii) Type of turbine.

Solution. Given :

Head, $H = 25$ m

Speed, $N = 200$ r.p.m.

Discharge, $Q = 9$ cumec $= 9\ m^3/s$

Efficiency, $\eta_o = 90\% = 0.90$ (Take the efficiency as overall η)

Now using relation, $$\eta_o = \frac{\text{Power developed}}{\text{Water power}} = \frac{P}{\frac{\rho \times g \times Q \times H}{1000}}$$

$\therefore$ $$P = \eta_o \times \frac{\rho \times g \times Q \times H}{1000}$$

$$= \frac{0.90 \times 9.81 \times 1000 \times 9 \times 25}{1000} = 1986.5 \text{ kW}$$

(*i*) *Specific speed of the machine* (N_s)

Using equation (18.28), $$N_s = \frac{N\sqrt{P}}{H^{5/4}} = \frac{200 \times \sqrt{1986.5}}{25^{5/4}} = \mathbf{159.46\ r.p.m.\ Ans.}$$

(*ii*) *Power generated*

$P =$ **1986.5 kW. Ans.**

(*iii*) As the specific speed lies between 51 and 255, the turbine is a Francis turbine. **Ans.**

Problem 18.38 *A turbine develops 9000 kW when running at a speed of 140 r.p.m. and under a head of 30 m. Determine the specific speed of the turbine.*

Solution. Given :

Power developed, $P = 9000$ kW

Head, $H = 30$ m

Speed , $N = 140$ r.p.m.

The specific speed is given by equation (18.28) as

$$N_s = \frac{N\sqrt{P}}{H^{5/4}} = \frac{140 \times \sqrt{9000}}{30^{5/4}} = \frac{13281.56}{70.21}$$

$$= \mathbf{189.167\ (S.I.\ units).\ Ans.}$$

Problem 18.39 *A Pelton wheel develops 8000 kW under a net head of 130 m at a speed of 200 r.p.m. Assuming the co-efficient of velocity for the nozzle 0.98, hydraulic efficiency 87%, speed ratio 0.46 and jet diameter to wheel diameter ratio $\frac{1}{9}$, determine :*

(i) the discharge required, (ii) the diameter of the wheel,

(iii) the diameter and number of jets required, and (iv) the specific speed.

Mechanical efficiency is 75%.

Solution. Given :

Power developed, $P = 8000$ kW

Net head, $H = 130$ m

Speed, $N = 200$ r.p.m.

Co-efficient of velocity, $C_v = 0.98$

Hydraulic efficiency, $\eta_h = 87\% = 0.87$

Speed ratio, $\dfrac{u_1}{\sqrt{2gH}} = 0.46$

$\therefore \quad u_1 = 0.46 \times \sqrt{2gH} = 0.46 \times \sqrt{2 \times 9.81 \times 130} = 23.23$ m/s.

Jet diameter to wheel diameter $= \dfrac{d}{D} = \dfrac{1}{9}$

Mechanical efficiency, $\eta_m = 75\% = 0.75$

Overall efficiency is given by equation (18.6) as

$$\eta_o = \eta_h \times \eta_m = 0.87 \times 0.75 = 0.6525$$

Also $\eta_o = \dfrac{\text{Power developed}}{\text{Water power}} = \dfrac{8000}{\text{W.P. in kW}}$ or $0.6525 = \dfrac{8000}{\text{W. P. in kW}}$

$\therefore$ W.P. in kW $= \dfrac{8000}{0.6525} = 12260.536$ kW

But W.P. in kW $= \dfrac{\rho \times g \times Q \times H}{1000}$

$= \dfrac{1000 \times 9.81 \times Q \times H}{1000}$ ($\because$ ρg in S.I. $= 1000 \times 9.81$)

$= Q \times H \times 9.81 = Q \times 130 \times 9.81$

$\therefore \quad 12260.536 = Q \times 130 \times 9.81$

$\therefore \quad Q = \dfrac{12260.536}{130 \times 9.81} =$ **9.614 m^3/s. Ans.**

(*i*) *Discharge required*

$Q =$ **9.614 m^3/s. Ans.**

(*ii*) *Diameter of wheel (D)*

Using the relation, $u_1 = \dfrac{\pi DN}{60}$

$\therefore \quad D = \dfrac{60 \times u_1}{\pi \times N} = \dfrac{60 \times 23.23}{\pi \times 200} =$ **2.218 m. Ans.**

(*iii*) *Diameter of jet (d) and number of jets required*

$$\frac{d}{D} = \frac{1}{9}$$

$\therefore \quad d = \dfrac{D}{9} = \dfrac{2.218}{9} = 0.2464$ m = **246.4 mm. Ans.**

$\therefore$ Area of jet, $a = \dfrac{\pi}{4} d^2 = \dfrac{\pi}{4}(.2464)^2 = .04768$ m^2.

Velocity of jet is given by, $V_1 = C_v\sqrt{2gH} = 0.98\sqrt{2\times 9.81\times 130} = 49.49$ m/s

$\therefore$ Discharge through one jet = Area of jet × Velocity of jet = $a \times V_1$

$= .04768 \times 49.49 = 2.359$ m^3/s

$\therefore$ Number of jets $= \dfrac{\text{Total discharge}}{\text{Discharge through one jet}}$

$= \dfrac{Q}{2.359} = \dfrac{9.614}{2.359} = 4.07$ say **4.0. Ans.**

(*iv*) *Specific speed* is given by equation (18.28) as

N_s (S.I. units) $= \dfrac{N\sqrt{P}}{H^{5/4}} = \dfrac{200\times\sqrt{8000}}{130^{5/4}} = \dfrac{17888.54}{438.96} =$ **40.75. Ans.**

Problem 18.40 *A Pelton turbine develops 3000 kW under a head of 300 m. The overall efficiency of the turbine is 83%. If speed ratio = 0.46, C_v = 0.98 and specific speed is 16.5, then find :*

(i) Diameter of the turbine, and (ii) Diameter of the jet.

Solution. Given :

Power, $P = 3000$ kW

Net head, $H = 300$ m

Overall efficiency, $\eta_o = 83\%$ or 0.83

Speed ratio $= 0.46$

Value of C_v, $= 0.98$

Specific speed*, $N_s = 16.5$

Using equation, $N_s = \dfrac{N\sqrt{P}}{H^{5/4}}$ or $N = \dfrac{N_s\, H^{5/4}}{\sqrt{P}} = \dfrac{16.5\times 300^{5/4}}{\sqrt{3000}} = 375$ r.p.m.

The velocity (V) at the outlet of nozzle is given by,

$V = C_v\sqrt{2\times g\times H} = 0.98\sqrt{2\times 9.81\times 300} = 75.1$ m/s

Now speed ratio $= \dfrac{u}{\sqrt{2gH}}$ or u = Speed ratio × $\sqrt{2gH}$

$= 0.46\times\sqrt{2\times 9.81\times 300} = 34.95$ m/s.

(*i*) *Diameter of the turbine (D)*

Using, $u = \dfrac{\pi DN}{60}$ or $D = \dfrac{60\times u}{\pi\times N} = \dfrac{60\times 34.95}{\pi\times 375} =$ **1.78 m. Ans.**

(*ii*) *Diameter of the jet (d)*

Let Q = Discharge through turbine in m^3/s

Using the relation, $\eta_o = \dfrac{P}{\left(\dfrac{\rho\times g\times Q\times H}{1000}\right)}$, where $\rho\times g = 1000\times 9.81$ N/m^3 for water

$\therefore$ $0.83 = \dfrac{3000}{\left(\dfrac{1000\times 9.81\times Q\times 300}{1000}\right)}$

* Specific speed is the speed of the turbine working under a unit head and develops one kilowatt power.

$$\therefore \quad Q = \frac{3000}{9.81 \times 300 \times 0.83} = 1.23 \text{ m}^3/\text{s}$$

But discharge through a Pelton turbine is given by,

$$Q = \text{Area of jet} \times \text{Velocity}$$

or $$1.23 = \frac{\pi}{4} d^2 \times 75.1$$

$$\therefore \quad d = \sqrt{\frac{4 \times 1.23}{\pi \times 75.1}} = 0.142 \text{ m} = \textbf{142 mm. Ans.}$$

Problem 18.41 *Water under a head of 300 m is available for a hydel-plant situated at a distance of 2.35 km from the source. The frictional losses of energy for transporting water is equivalent to 26 (J/N). A number of Pelton wheels are to be installed generating a total output of 18 MW. Determine the number of units to be installed, diameter of Pelton wheel and the jet diameter when the following are available : Wheel speed 650 r.p.m.; ratio of bucket to jet speed 0.46 ; specific speed not to exceed 30 (m, kW, r.p.m.) ; C_v and C_d for the nozzle 0.97 and 0.94 respectively and the overall efficiency of the wheel 87%.*

Solution. Given :

Total head = 300 m

Length = 2.35 km = 2350 m

Frictional losses = 26 (J/N) = 26 (Nm/N) (as J = Nm) = 26 m

$\therefore$ Net head, $H = 300 - 26 = 274$ m

Total output = 18 MW = 18×10^3 kW

$N = 650$ r.p.m.

Ratio of bucket to jet speed = 0.46

$C_v = 0.97$, $C_d = 0.94$

$\eta_o = 87\% = 0.87$

and $N_s = 30$, where H is in m, P in kW and N in r.p.m.

Find : (*i*) Number of units to be installed

(*ii*) Dia. of Pelton wheel (D)

(*iii*) Dia. of jet of water (d)

(*i*) *Number of units to be installed*

Let P = Power output of each unit in kW

Using equation (18.28) as $N_s = \frac{N\sqrt{P}}{H^{5/4}}$ or $30 = \frac{650 \times \sqrt{P}}{274^{5/4}}$ or $\sqrt{P} = \frac{30 \times 274^{5/4}}{650}$

Squaring both sides, we get $P = \frac{30^2 \times 274^{5/2}}{650^2} = 2647.2$ kW

$\therefore$ No. of units $= \frac{\text{Total output in kW}}{\text{Output of one unit in kW}}$

$$= \frac{18 \times 10^3}{2647.2} = 6.799 \simeq \textbf{7 units. Ans.}$$

(*ii*) *Dia. of Pelton wheel (D)*

Velocity of jet is given by, $V_1 = C_v \times \sqrt{2gH}$

$= 0.97 \times \sqrt{2 \times 9.81 \times 274} = 71.12$ (m/s)

But ratio of bucket to jet speed $= 0.46$*

or $$\frac{\text{Speed of bucket}}{\text{Speed of jet}} = 0.46 \text{ or } \frac{u_1}{V_1} = 0.46$$

$$\therefore \quad u_1 = 0.46 \times V_1 = 0.46 \times 71.12 = 32.715 \text{ m/s.}$$

But $$u_1 = \frac{\pi DN}{60}$$

$$\therefore \quad 32.715 = \frac{\pi \times D \times 650}{60} \text{ or } \frac{32.715 \times 60}{\pi \times 650} = D \text{ or } 0.945 \text{ m} = D$$

$\therefore$ Dia. of Pelton wheel $=$ **0.945. Ans.**

(*iii*) *Dia. of jet* (*d*)

We know $$\eta_o = \frac{\text{Total power output}}{\text{Total water power in kW}}$$

or $$0.87 = \frac{18 \times 10^3}{\text{Total water power in kW}}$$

$\therefore$ Total water power in kW $= \dfrac{18 \times 10^3}{0.87} = 20.689 \times 10^3$ kW

$\therefore$ Water power in kW per unit $= \dfrac{\text{Total water power}}{\text{No. of units}} = \dfrac{20.689 \times 10^3}{7} = 2.955 \times 10^3$ kW

But water power in kW per unit is given by equation (18.3 *A*) as,

Water power $$= \frac{\rho \times g \times Q \times H}{1000} \text{ kW}$$

$$\therefore \quad 2.955 \times 10^3 = \frac{\rho \times g \times Q \times H}{1000} = \frac{(1000 \times 9.81) \times Q \times H}{1000}$$

$(\because \rho \times g = 1000 \times 9.81)$

$$= 9.81 \times Q \times 274$$

$$\therefore \quad Q = \frac{2.955 \times 10^3}{9.81 \times 274} = 1.099 \text{ m}^3\text{/s}$$

But discharge (Q) through one unit is also given by

$$Q = C_d \times \frac{\pi}{4} d^2 \times \sqrt{2gH}$$

---

* It is not speed ratio. It is the ratio of bucket speed to jet speed *i.e.*, ratio of u_1 and V_1. Speed ratio is $u_1\sqrt{2gH}$.

or $$1.099 = 0.94 \times \frac{\pi}{4} d^2 \times \sqrt{2 \times 9.81 \times 274}$$

or $$d^2 = \frac{1.099 \times 4}{0.94 \times \pi \times \sqrt{2 \times 9.81 \times 274}} = 0.0203 \text{ m}$$

$\therefore$ $$d = \sqrt{0.0203} = 0.1424 \text{ m} = \mathbf{142.4 \text{ mm. Ans.}}$$

▶ 18.12 UNIT QUANTITIES

In order to predict the behaviour of a turbine working under varying conditions of head, speed, output and gate opening, the results are expressed in terms of quantities which may be obtained when the head on the turbine is reduced to unity. The conditions of the turbine under unit head are such that the efficiency of the turbine remains unaffected. The following are the three important unit quantities which must be studied under unit head :

1. Unit speed, 2. Unit discharge, and 3. Unit power.

18.12.1 Unit Speed. It is defined as the speed of a turbine working under a unit head (*i.e.*, under a head of 1 m). It is denoted by 'N_u'. The expression for unit speed (N_u) is obtained as :

Let N = Speed of a turbine under a head H,

H = Head under which a turbine is working,

u = Tangential velocity.

The tangential velocity, absolute velocity of water and head on the turbine are related as

$$u \propto V, \qquad \text{where } V \propto \sqrt{H}$$

$$\propto \sqrt{H} \qquad ...(i)$$

Also tangential velocity (u) is given by

$$u = \frac{\pi DN}{60}, \qquad \text{where } D = \text{Diameter of turbine.}$$

For a given turbine, the diameter (D) is constant.

$\therefore$ $$u \propto N \text{ or } N \propto u \text{ or } N \propto \sqrt{H} \qquad (\because \text{ From } (i),\ u \propto \sqrt{H})$$

$\therefore$ $$N = K_1 \sqrt{H} \qquad ...(ii)$$

where K_1 is a constant of proportionality.

If head on the turbine becomes unity, the speed becomes unit speed or

when $$H = 1, N = N_u$$

Substituting these values in equation (*ii*), we get

$$N_u = K_1 \sqrt{1.0} = K_1$$

Substituting the value of K_1 in equation (*ii*),

$$N = N_u \sqrt{H} \text{ or } N_u = \frac{N}{\sqrt{H}}. \qquad ...(18.29)$$

18.12.2 Unit Discharge. It is defined as the discharge passing through a turbine, which is working under a unit head (*i.e.*, 1 m). It is denoted by the symbol 'Q_u'. The expression for unit discharge is given as :

Let H = Head of water on the turbine,

Q = Discharge passing through turbine when head is H on the turbine,

a = Area of flow of water.

The discharge passing through a given turbine under a head 'H' is given by,

$$Q = \text{Area of flow} \times \text{Velocity}$$

But for a turbine, area of flow is constant and velocity is proportional to $\sqrt{H}$.

$$\therefore \quad Q \propto \text{Velocity} \propto \sqrt{H}$$

or $$Q = K_2 \sqrt{H} \qquad ...(iii)$$

where K_2 is constant of proportionality.

If $H = 1$, $Q = Q_u$. (By definition)

Substituting these values in equation (*iii*), we get

$$Q_u = K_2 \sqrt{1.0} = K_2.$$

Substituting the value of K_2 in equation (*iii*), we get

$$Q = Q_u \sqrt{H}$$

$$\therefore \quad Q_u = \frac{Q}{\sqrt{H}} \qquad ...(18.30)$$

18.12.3 Unit Power. It is defined as the power developed by a turbine, working under a unit head (*i.e.*, under a head or 1 m). It is denoted by the symbol 'P_u'. The expression for unit power is obtained as :

Let H = Head of water on the turbine,

P = Power developed by the turbine under a head of H,

Q = Discharge through turbine under a head H.

The overall efficiency (η_o) is given as

$$\eta_o = \frac{\text{Power developed}}{\text{Water power}} = \frac{P}{\frac{\rho \times g \times Q \times H}{1000}}$$

$$\therefore \quad P = \eta_o \times \frac{\rho \times g \times Q \times H}{1000}$$

$$\propto Q \times H$$

$$\propto \sqrt{H} \times H \qquad (\because \quad Q \propto \sqrt{H})$$

$$\propto H^{3/2}$$

$$\therefore \quad P = K_3 H^{3/2} \qquad ...(iv)$$

where K_3 is a constant of proportionality.

When $H = 1$ m $\quad P = P_u$

$$\therefore \quad P_u = K_3 (1)^{3/2} = K_3.$$

Substituting the value of K_3 in equation (*iv*), we get

$$P = P_u H^{3/2}$$

$$\therefore \quad P_u = \frac{P}{H^{3/2}}. \qquad ...(18.31)$$

18.12.4 Use of Unit Quantities (N_u, Q_u, P_u). If a turbine is working under different heads, the behaviour of the turbine can be easily known from the values of the unit quantities, *i.e.*, from the values of unit speed, unit discharge and unit power.

Let H_1, H_2, ...are the heads under which a turbine works,
N_1, N_2, ...are the corresponding speeds,
Q_1, Q_2, ...are the discharge, and
P_1, P_2, ...are the power developed by the turbine.

Using equations (18.29), (18.30) and (18.31) respectively,

$$\left.\begin{aligned} N_u &= \frac{N_1}{\sqrt{H_1}} = \frac{N_2}{\sqrt{H_2}} \\ Q_u &= \frac{Q_1}{\sqrt{H_1}} = \frac{Q_2}{\sqrt{H_2}} \\ P_u &= \frac{P_1}{H_1^{3/2}} = \frac{P_2}{H_2^{3/2}} \end{aligned}\right\} \quad ...(18.32)$$

Hence, if the speed, discharge and power developed by a turbine under a head are known, then by using equation (18.32) the speed, discharge and power developed by the same turbine under a different head can be obtained easily.

Problem 18.41 (A) *A turbine develops 9000 kW when running at 10 r.p.m. The head on the turbine is 30 m. If the head on the turbine is reduced to 18 m, determine the speed and power developed by the turbine.*

Solution. Given :

Power developed, $P_1 = 9000$ kW
Speed, $N_1 = 100$ r.p.m.
Head, $H_1 = 30$ m
Let for a head, $H_2 = 18$ m
Speed $= N_2$
Power $= P_2$

Using equation (18.32), $\frac{N_1}{\sqrt{H_1}} = \frac{N_2}{\sqrt{H_2}}$

$$N_2 = \frac{N_1\sqrt{H_2}}{\sqrt{H_1}} = \frac{100\sqrt{18}}{\sqrt{30}} = \frac{100 \times 4.2426}{5.4772} = \mathbf{77.46\ r.p.m.\ Ans.}$$

Also we have $\frac{P_1}{H_1^{3/2}} = \frac{P_2}{H_2^{3/2}}$

$\therefore$ $$P_2 = \frac{P_1 H_2^{3/2}}{H_1^{3/2}} = \frac{9000 \times 18^{3/2}}{30^{3/2}} = \frac{687307.78}{164.316} = \mathbf{4182.84\ kW.\ Ans.}$$

Problem 18.42 *A turbine develops 500 kW power under a head of 100 metres at 200 r.p.m. What would be its normal speed and output under a head of 81 metres ?*

Solution. Given :

Power, $P_1 = 500$ kW
Head, $H_1 = 100$ m

Speed, $N_1 = 200$ r.p.m.
For a head, $H_2 = 81$ m
Let, N_2 = Speed
P_2 = Power
Using equation (18.32) for speed, we have

$$\frac{N_1}{\sqrt{H_1}} = \frac{N_2}{\sqrt{H_2}}$$

$$\therefore \quad N_2 = \sqrt{H_2} \times \frac{N_1}{\sqrt{H_1}} = \sqrt{\frac{H_2}{H_1}} \times N_1 = \sqrt{\frac{81}{100}} \times 200$$

$$= \frac{9}{10} \times 20 = \textbf{180 r.p.m. Ans.}$$

Using equation (18.32) for power, we have

$$\frac{P_1}{H_1^{3/2}} = \frac{P_2}{H_2^{3/2}}$$

$$\therefore \quad P_2 = H_2^{3/2} \times \frac{P_1}{H_1^{3/2}} = \frac{81^{3/2}}{100^{3/2}} \times 500$$

$$= \frac{729}{1000} \times 500 = \textbf{364.5 kW. Ans.}$$

Problem 18.43 *A turbine is to operate under a head of 25 m at 200 r.p.m. The discharge is 9 cumec. If the efficiency is 90%, determine the performance of the turbine under a head of 20 metres.*

Solution. Given :
Head on turbine, $H_1 = 25$ m
Speed, $N_1 = 200$ r.p.m.
Discharge, $Q_1 = 9$ m^3/s
Overall efficiency, $\eta_o = 90\%$ or 0.90.
Performance of the turbine under a head, $H_2 = 20$ m, means to find the speed, discharge and power developed by the turbine when working under the head of 20 m.
Let for the head, $H_2 = 20$ m, Speed = N_2, discharge = Q_2 and power = P_2
Using the relation,

$$\eta_o = \frac{P}{\text{W.P.}} = \frac{p_1}{\frac{\rho \times g \times Q_1 \times H_1}{1000}}$$

$$\therefore \quad P_1 = \frac{\eta_o \times \rho \times g \times Q_1 \times H_1}{1000} = \frac{0.90 \times 1000 \times 9.81 \times 9 \times 25}{1000} = 1986.5 \text{ kW}$$

Using equation (18.32), $\dfrac{N_1}{\sqrt{H_1}} = \dfrac{N_2}{\sqrt{H_2}}$

$$\therefore \quad N_2 = \frac{N_1\sqrt{H_2}}{\sqrt{H_1}} = 200 \times \frac{\sqrt{20}}{\sqrt{25}} = \textbf{178.88 r.p.m. Ans.}$$

Also $$\frac{Q_1}{\sqrt{H_1}} = \frac{Q_2}{\sqrt{H_2}}$$

$\therefore$ $$Q_2 = Q_1 \times \frac{\sqrt{H_2}}{\sqrt{H_1}} = 9.0 \times \sqrt{\frac{20}{25}} = \mathbf{8.05\ m^3/s.\ Ans.}$$

And $$\frac{P_1}{H_1^{3/2}} = \frac{P_2}{H_2^{3/2}}$$

$\therefore$ $$P_2 = \frac{P_1 H_2^{3/2}}{H_1^{3/2}} = P_1 \left(\frac{H_2}{H_1}\right)^{3/2} = 1986.5 \left(\frac{20}{25}\right)^{3/2} = \mathbf{1421.42\ kW.\ Ans.}$$

Problem 18.44 *A Pelton wheel is revolving at a speed of 190 r.p.m. and develops 5150.25 kW when working under a head of 220 m with an overall efficiency of 80%. Determine unit speed, unit discharge and unit power. The speed ratio for the turbine is given as 0.47. Find the speed, discharge and power when this turbine is working under a head of 140 m.*

Solution. Given :

Speed, $N_1 = 190$ r.p.m.
Power, $P_1 = 5150.25$ kW
Head, $H_1 = 220$ m
Overall efficiency, $\eta_o = 80\% = 0.80$
Speed ratio $= 0.47$
New head of water, $H_2 = 140$ m

Overall efficiency is given by $\eta_o = \dfrac{P_1}{\dfrac{\rho \times g \times Q_1 \times H_1}{1000}} = \dfrac{1000 \times P_1}{\rho \times g \times Q_1 \times H_1}$

$\therefore$ $$Q_1 = \frac{1000 \times P_1}{\eta_o \times \rho \times g \times H_1} = \frac{1000 \times 5150.25}{0.80 \times 1000 \times 9.81 \times 220} = 2.983\ \text{m}^3/\text{s}$$

Unit speed is given by equation (18.29),

$$N_u = \frac{N_1}{\sqrt{H_1}} = \frac{190}{\sqrt{220}} = \mathbf{12.81\ r.p.m.\ Ans.}$$

Unit discharge is given by equation (18.30),

$$Q_u = \frac{Q_1}{\sqrt{H_1}} = \frac{2.983}{\sqrt{220}} = \mathbf{0.201\ m^3/s.\ Ans.}$$

Unit power is given by equation (18.31),

$$P_u = \frac{P_1}{H_1^{3/2}} = \frac{5150.25}{220^{3/2}} = \mathbf{1.578\ kW.\ Ans.}$$

When the turbine is working under a new head of 140 m, the speed, discharge and power are given by equation (18.32) as

For speed, $$\frac{N_1}{\sqrt{H_1}} = \frac{N_2}{\sqrt{H_2}}$$

$\therefore$ $$N_2 = \frac{N_1 \sqrt{H_2}}{\sqrt{H_1}} = N_1 \sqrt{\frac{H_2}{H_1}} = 190 \sqrt{\frac{140}{220}} = \mathbf{151.56\ r.p.m.\ Ans.}$$

For discharge, $\frac{Q_1}{\sqrt{H_1}} = \frac{Q_2}{\sqrt{H_2}}$

$\therefore \quad Q_2 = \frac{Q_1\sqrt{H_2}}{\sqrt{H_1}} = Q_1\sqrt{\frac{H_2}{H_1}} = 2.983\sqrt{\frac{140}{220}} = \mathbf{2.379\ m^3/s.\ Ans.}$

For power, $\frac{P_1}{H_1^{3/2}} = \frac{P_2}{H_2^{3/2}}$

$\therefore \quad P_2 = P_1\frac{H_2^{3/2}}{H_1^{3/2}} = P_1\left(\frac{H_2}{H_1}\right)^{3/2} = 5150.25\left(\frac{140}{220}\right)^{3/2} = \mathbf{2614.48\ kW.\ Ans.}$

Problem 18.45 *A Pelton wheel is supplied with water under a head of 35 m at the rate of 40.5 kilo litre/min. The bucket deflects the jet through an angle of 160° and the mean bucket speed is 13 m/s. Calculate the power and hydraulic efficiency of the turbine.*

Solution. Given :

Net head, $H = 35$ m

Discharge, $Q = 40.5$ kilo litre/min

$= 40.5 \times 1000$ litre/min

$= \frac{40.5\times1000}{1000}$ m^3/min $\qquad \left(1\text{ litre} = \frac{1}{1000}\text{m}^3\right)$

$= 40.5$ m^3 / min $= \frac{40.5}{60}$ m^3/s $= 0.675$ m^3/s

Angle of deflection $= 160°$

$\therefore$ Angle, $\phi = 180° - 160° = 20°$

Mean bucket speed, $u = u_1 = u_2 = 13$ m/s

Calculate : (*i*) Power at runner and (*ii*) Hydraulic efficiency.

Taking the value of $C_v = 1.0$

The velocity of jet, $V_1 = C_v\sqrt{2gH} = 1 \times \sqrt{2\times9.81\times35} = 26.2$ m/s

$\therefore \quad V_{r_1} = V_1 - u_1 = 26.2 - 13 = 13.2$ m/s

Also $V_{w_1} = V_1 = 26.2$ m/s

$V_{r_2} = V_{r_1} = 13.2$ m/s

and $V_{w_2} = V_{r_2}\cos\phi - u_2$

$= 13.2 \times \cos 20° - 13 = 12.554 - 13 = -0.446$ m/s

(*i*) *Power at runner*

Using equation (18.9), we get the work done by the jet on the runner per second.

$\therefore$ Work done/s $= \rho \times a \times V_1\left[V_{w_1} + V_{w_2}\right] \times u$

$= \rho \times Q \times \left[V_{w_1} + V_{w_2}\right] \times u \qquad (\because \quad a \times V_1 = Q)$

$$= 1000 \times 0.675\ [26.2 + (-0.446)] \times 13\ \frac{\text{Nm}}{\text{s}} = 225991\ \text{W}$$

($\because$ Nm/s = W)

$$= 225.991\ \text{kW}$$

$\therefore$ Power at runner = **225.991 kW. Ans.**

(*ii*) *Hydraulic efficiency*

Input power in kW is given by equation (18.3*A*).

$\therefore$ Input power $= \dfrac{\rho \times g \times Q \times H}{1000}$, where $\rho = 1000\ \text{kg/m}^3$

$$= \frac{1000 \times 9.81 \times 0.675 \times 35}{1000} = 231.761\ \text{kW}$$

$\therefore$ Hydraulic efficiency $= \dfrac{\text{Power at runner}}{\text{Input power}}$

$$= \frac{225.991}{231.761} = 0.975 = 0.975 \times 100 = \mathbf{97.5\%.\ Ans.}$$

▶ 18.13 CHARACTERISTIC CURVES OF HYDRAULIC TURBINES

Characteristic curves of a hydraulic turbine are the curves, with the help of which the exact behaviour and performance of the turbine under different working conditions, can be known. These curves are plotted from the results of the tests performed on the turbine under different working conditions.

The important parameters which are varied during a test on a turbine are :

1. Speed (N)
2. Head (H)
3. Discharge (Q)
4. Power (P)
5. Overall efficiency (η_o) and
6. Gate opening.

Out of the above six parameters, three parameters namely speed (N), head (H) and discharge (Q) are independent parameters.

Out of the three independent parameters, (N, H, Q) one of the parameter is kept constant (say H) and the variation of the other four parameters with respect to any one of the remaining two independent variables (say N and Q) are plotted and various curves are obtained. These curves are called characteristic curves. The following are the important characteristic curves of a turbine.

1. Main Characteristic Curves or Constant Head Curves.
2. Operating Characteristic Curves or Constant Speed Curves.
3. Muschel Curves or Constant Efficiency Curves.

18.13.1 Main Characteristic Curves or Constant Head Curves. Main characteristic curves are obtained by maintaining a constant head and a constant gate opening (G.O.) on the turbine. The speed of the turbine is varied by changing load on the turbine. For each value of the speed, the corresponding values of the power (P) and discharge (Q) are obtained. Then the overall efficiency (η_o) for each value of the speed is calculated. From these readings the values of unit speed (N_u), unit power (P_u) and unit discharge (O_u) are determined. Taking N_u as abscissa, the values of Q_u, P_u, P and η_o are plotted as shown in Figs. 18.35 and 18.36. By changing the gate opening, the values of Q_u, P_u and η_o and N_u are determined and taking N_u as abscissa, the values of Q_u, P_u and η_o are plotted. Fig. 18.35

shows the main characteristic curves for Pelton wheel and Fig. 18.36 shows the main characteristic curves for reaction (Francis and Kaplan) turbines.

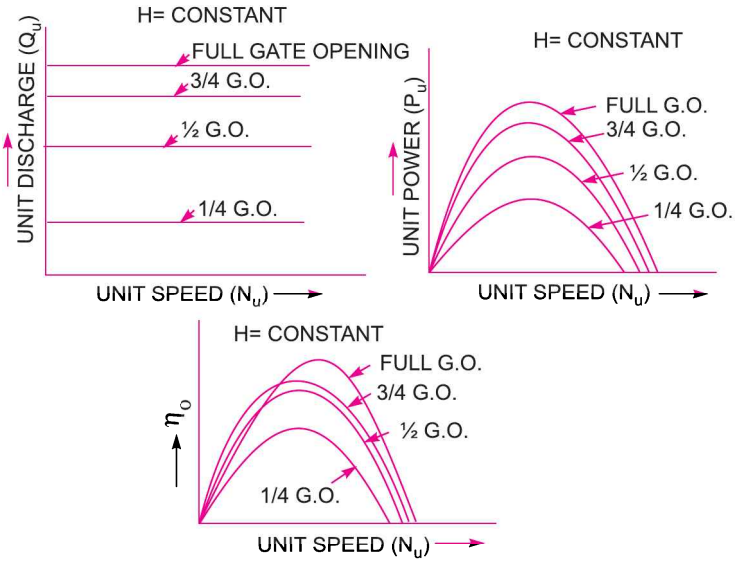


Fig. 18.35 *Main characteristic curves for a Pelton wheel.*

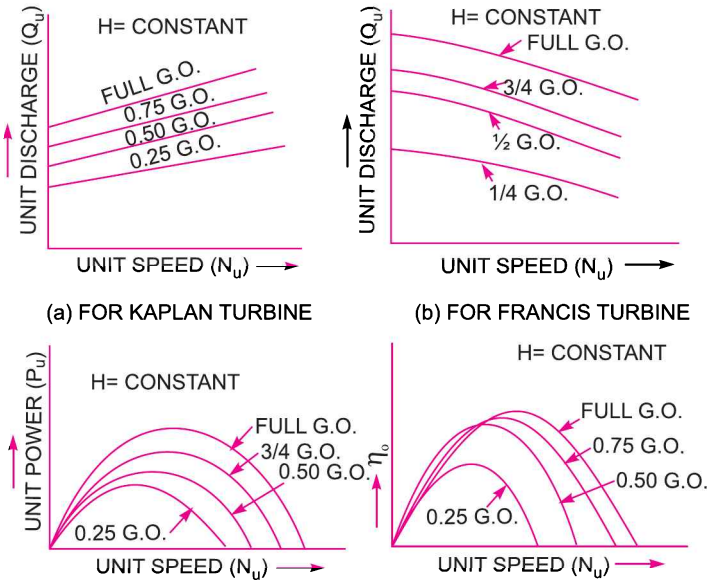


Fig. 18.36 *Main characteristic curves for reaction turbine.*

18.13.2 Operating Characteristic Curves or Constant Speed Curves. Operating characteristic curves are plotted when the speed on the turbine is constant. In case of turbines, the head is generally constant. As mentioned in Art. 18.13, there are three independent parameters namely N, H and Q. For operating characteristics N and H are constant and hence the variation of power and efficiency with respect to discharge Q are plotted. The power curve for turbines shall not pass through

the origin because certain amount of discharge is needed to produce power to overcome initial friction. Hence the power and efficiency curves will be slightly away from the origin on the x-axis, as to overcome initial friction certain amount of discharge will be required. Fig. 18.37 shows the variation of power and efficiency with respect to discharge.

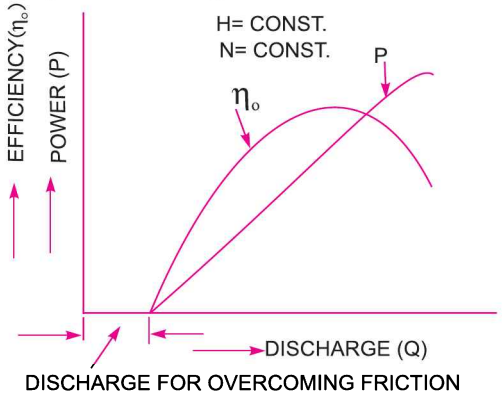


Fig. 18.37 *Operating characteristic curves.*

18.13.3 Constant Efficiency Curves or Muschel Curves or Iso-Efficiency Curves. These curves are obtained from the speed *vs*. efficiency and speed *vs*. discharge curves for different gate openings. For a given efficiency from the N_u *vs*. η_o curves, there are two speeds. From the N_u *vs*. Q_u curves, corresponding to two values of speeds there are two values of discharge. Hence for a given efficiency there are two values of discharge for a particular gate opening. This means for a given efficiency there are two values of speeds and two values of the discharge for a given gate opening. If the efficiency is maximum there is only one value. These two values of speed and two values of discharge corresponding to a particular gate opening are plotted as shown in Fig. 18.38 (*b*). The procedure is repeated for different gate openings and the curves Q *vs*. N are plotted. The points having the same efficiencies are joined. The curves having same efficiency are called iso-efficiency curves. These curves are helpful for determining the zone of constant efficiency and for predicating the performance of the turbine at various efficiencies.

For plotting the iso-efficiency curves, horizontal lines representing the same efficiency are drawn on the η_o ~ speed curves. The points at which these lines cut the efficiency curves at various gate openings are transferred to the corresponding Q ~ speed curves. The points having the same efficiency are then joined by a smooth curves. These smooth curves represents the iso-efficiency curve.

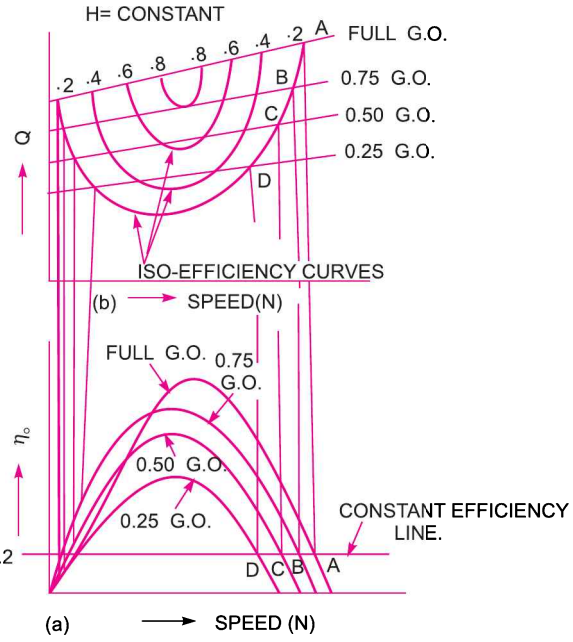


Fig. 18.38 *Constant efficiency curve.*

▶ 18.14 GOVERNING OF TURBINES

The governing of a turbine is defined as the operation by which the speed of the turbine is kept constant under all conditions of working. It is done automatically by means of a governor, which regulates the rate of flow through the turbines according to the changing load conditions on the turbine.

Governing of a turbine is necessary as a turbine is directly coupled to an electric generator, which is required to run at constant speed under all fluctuating load conditions. The frequency of power generation by a generator of constant number of pair of poles under all varying conditions should be constant. This is only possible when the speed of the generator, under all changing load condition, is constant. The speed of the generator will be constant, when the speed of the turbine (which is coupled to the generator) is constant.

When the load on the generator decreases, the speed of the generator increases beyond the normal speed (constant speed). Then the speed of the turbine also increases beyond the normal speed. If the turbine or the generator is to run at constant (normal) speed, the rate of flow of water to the turbine should be decreased till the speed becomes normal. This process by which the speed of the turbine (and hence of generator) is kept constant under varying condition of load is called governing.

Governing of Pelton Turbine (Impulse Turbine)

Governing of Pelton turbine is done by means of oil pressure governor, which consists of the following parts :

1. Oil sump.
2. Gear pump also called oil pump, which is driven by the power obtained from turbine shaft.
3. The Servomotor also called the relay cylinder.
4. The control valve or the distribution valve or relay valve.
5. The centrifugal governor or pendulum which is driven by belt or gear from the turbine shaft.
6. Pipes connecting the oil sump with the control valve and control valve with servomotor and
7. The spear rod or needle.

Fig. 18.39 shows the position of the piston in the relay cylinder, position of control or relay valve and fly-balls of the centrifugal governor, when the turbine is running at the normal speed.

When the load on the generator decreases, the speed of the generator increases. This increases the speed of the turbine beyond the normal speed. The centrifugal governor, which is connected to the turbine main shaft, will be rotating at an increased speed . Due to increase in the speed of the centrifugal governor, the fly-balls move upward due to the increased centrifugal force on them. Due to the upward movement of the fly-balls, the sleeve will also move upward. A horizontal lever, supported over a fulcrum, connects the sleeve and the piston rod of the control valve. As the sleeve moves up, the lever turns about the fulcrum and the piston rod of the control valve moves downward. This closes the valve V_1 and opens the valve V_2 as shown in Fig. 18.39.

The oil, pumped from the oil pump to the control valve or relay valve, under pressure will flow through the valve V_2 to the servomotor (or relay cylinder) and will exert force on the face A of the piston of the relay cylinder. The piston along with piston rod and spear will move towards right. This will decrease the area of flow of water at the outlet of the nozzle. This decrease of area of flow will reduce the rate of flow of water to the turbine which consequently reduces the speed of the turbine. When the speed of the turbine becomes normal, the fly-balls, sleeve, lever and piston rod of control valve come to its normal position as shown in Fig. 18.39.

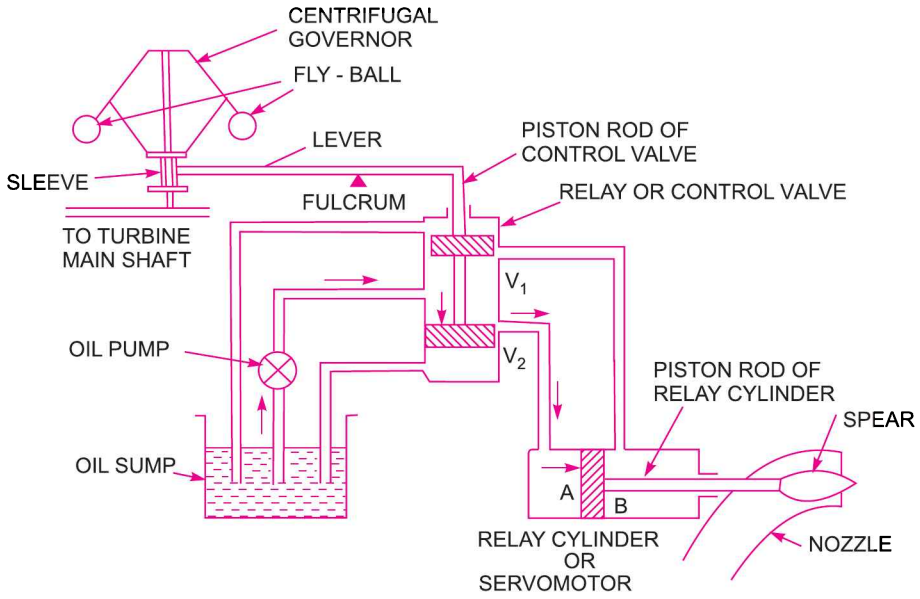


Fig. 18.39. *Governing of Pelton turbine.*

When the load on the generator increases, the speed of the generator and hence of the turbine decreases. The speed of the centrifugal governor also decreases and hence centrifugal force acting on the fly-balls also reduces. This brings the fly-balls in the downward direction. Due to this, the sleeve moves downward and the lever turns about the fulcrum, moving the piston rod of the control valve in the upward direction. This closes the valve V_2 and opens the valve V_1. The oil under pressure from the control valve, will move through valve V_1 to the servomotor and will exert a force on the face B of the piston. This will move the piston along with the piston rod and spear towards left, increasing the area of flow of water at the outlet of the nozzle. This will increase the rate of flow of water to the turbine and consequently, the speed of the turbine will also increase, till the speed of the turbine becomes normal.

HIGHLIGHTS

1. The hydraulic machines, which convert the hydraulic energy into mechanical energy, are called turbines.
2. Gross head is the vertical difference between the head race and tail race levels. Net head or effective head is the head, available at the inlet of the turbine. It is given by

$$H = H_g - h_f$$

where H_g = Gross head, and

h_f = Loss of head due to friction in penstocks

$$= \frac{4f \times L \times V^2}{D \times 2g}$$

where D = Dia. of penstock.

3. The efficiencies of a turbine are : (*i*) Hydraulic efficiency, η_h, (*ii*) Mechanical efficiency, η_m, and (*iii*) Overall efficiency, η_o.
4. Hydraulic efficiency, η_h is given by

$$\eta_h = \frac{\text{Power delivered to runner}}{\text{Power supplied at inlet}} = \frac{\text{R.P.}}{\text{W.P.}}$$

$$= \frac{W}{g}\frac{(V_{w_1}u_1 \pm V_{w_2}u_2)}{1000} \Big/ \frac{(W \times H)}{1000} = \frac{(V_{w_1}u_1 \pm V_{w_2}u_2)}{gH}$$

where W.P. = Water power,
R.P. = Runner power *i.e.*, power available at the runner of the turbine,
S.P. = Shaft power *i.e.*, power at the shaft of the turbine.

5. Mechanical efficiency, η_m is given by $\eta_m = \frac{\text{S.P.}}{\text{R.P.}}$

6. Overall efficiency, η_o is given by $\eta_o = \frac{\text{S.P.}}{\text{W.P.}} = \eta_m \times \eta_h$.

7. If at the inlet of a turbine, the energy available is only kinetic energy, the turbine is known as impulse turbine. But if at the inlet of the turbine, the energy available is kinetic energy as well as pressure energy, the turbine is called reaction turbine.

8. Pelton wheel (or turbine) is a tangential flow impulse turbine and is used for high head. In this turbine,

$$V_1 = C_v\sqrt{2gH}\ , u_1 = u_2 = u.$$

9. For the maximum efficiency of Pelton wheel the condition is $u = \frac{V_1}{2}$

Max. efficiency is given by $\eta_{\max} = \frac{(1+\cos\phi)}{2}$, where ϕ = Vane angle at outlet.

10. The jet ratio (m) is defined as ratio of the pitch diameter (D) of the Pelton wheel to the diameter of the jet (d) or $m = \frac{D}{d}$.

11. Francis turbine is an inward radial flow reaction turbine having discharge radial at outlet, which means the angle made by absolute velocity at outlet is 90°, *i.e.*, $\beta = 90°$. Then $V_{w_2} = 0$ and work done by water on the runner per second per unit weight of water becomes as $= \frac{1}{g}V_{w_1} \times u_1$.

12. Speed ratio is the ratio of the velocity of wheel at inlet to the velocity given by $\sqrt{2gH}$ whereas the flow ratio is the ratio of velocity of flow at inlet to the velocity given by $\sqrt{2gH}$.

13. Kaplan turbine is an axial flow reaction turbine in which the vanes on the hub are adjustable. The peripheral velocity at inlet and outlet are equal, *i.e.*, $u_1 = u_2$.

14. The discharge through a turbine is given by

$$Q = \frac{\pi}{4}d^2 \times \sqrt{2gH} \qquad \text{...For a Pelton wheel}$$

$$= \pi D_1 B_1 \times V_{f_1} \qquad \text{...For a Francis turbine}$$

$$= \frac{\pi}{4}(D_o^2 - D_b^2) \times V_{f_1} \qquad \text{...For a Kaplan turbine.}$$

15. Draft-tube is a pipe of gradually increasing area used for discharging water from the exit of a reaction turbine. They may be conical or simple elbow type. The efficiency of the draft-tube is given by

$$\eta_d = \frac{\left(\frac{V_1}{2g} - \frac{V_2^2}{2g}\right) - h_f}{\left(\frac{V_1^2}{2g}\right)}$$

where V_1 = Velocity of water at the inlet of the draft-tube,
V_2 = Velocity of water at the outlet of the draft-tube,
h_f = Loss of head in draft-tube.

16. Specific speed of a turbine is defined as the speed at which a turbine runs when it is working under a unit head and develops unit power. The expression for specific speed (N_s) is given as

$$N_s = \frac{N\sqrt{P}}{H^{5/4}}$$

where P = shaft power in kW, H = Net head on the turbine.

17. Unit quantities are the quantities (like speed, discharge, power, etc.) which are obtained when the head on the turbine is unity. They are unit speed (N_u), unit power (P_u) and unit discharge (Q_u). They are given as

$$N_u = \frac{N}{\sqrt{H}}, \; Q_u = \frac{Q}{\sqrt{H}}, \; P_u = \frac{P}{H^{3/2}} \; .$$

18. The important characteristic curves of a turbine are :
(*a*) Main characteristic curves or Constant head curves.
(*b*) Operating characteristic curves or Constant speed curves, and
(*c*) Muschel curves or Constant efficiency curves.

19. Governing of a turbine is defined as the operation by which the speed of the turbine is kept constant under all conditions of working. It is done by oil pressure governor.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define the terms : Hydraulic machines, Turbines and Pumps.

2. Differentiate between the turbines and pumps.

3. (*a*) What do you mean by gross head, net head and efficiency of turbine ? Explain the different types of the efficiency of a turbine.
(*b*) Explain clearly the following terms as they are applied to a Pelton wheel :
(*i*) Gross head ; (*ii*) Net head.

4. How will you classify the turbines ?

5. Differentiate between : (*a*) The impulse and reaction turbines, (*b*) Radial and axial flow turbines, (*c*) Inward and outward radial flow turbine, and (*d*) Kaplan and propeller turbines.

6. Obtain an expression for the work done per second by water on the runner of a Pelton wheel. Hence derive an expression for maximum efficiency of the Pelton wheel giving the relationship between the jet speed and bucket speed.
Draw inlet and outlet velocity triangles for a Pelton turbine and indicate the direction of various velocities.

7. Prove that the work done per second per unit weight of water in a reaction turbine is given as

$$= \frac{1}{g}(V_{w_1} u_1 \pm V_{w_2} u_2)$$

where V_{w_1} and V_{w_2} = Velocities of whirl at inlet and outlet,

u_1 and u_2 = Peripheral velocities at inlet and outlet.

8. Define the terms : speed ratio, flow ratio and jet ratio.
9. (*a*) What is a draft-tube ? Why is it used in a reaction turbine ? Describe with sketch two different types of draft-tubes.
   (*b*) What are the uses of a draft-tube ? Describe with neat sketches different types of draft-tubes. (*J.N.T.U., Hyderabad S 2002*).
10. What is the basis of selection of a turbine at a particular place ?
11. Define the specific speed of a turbine ? Derive an expression for the specific speed. What is the significance of the specific speed?
12. What are unit quantities ? Define the unit quantities for a turbine. Why are they important ?
13. Obtain an expression for unit speed, unit discharge and unit power for a turbine.
14. What do you understand by the characteristic curves of a turbine ? Name the important types of characteristic curves.
15. Define the term 'Governing of a turbine'. Describe with a neat sketch the working of an oil pressure governor.
16. Give the range of specific speed values of the Kaplan, Francis turbines and Pelton wheels.
    What factors decide whether Kaplan, Francis, or a Pelton type turbine would be used in a hydroelectric project ?
17. (*a*) Draw neat sketches of the Pelton turbine and Francis Turbine.
    (*b*) Describe briefly the function of various main components of Pelton turbine with neat sketches.
18. What is cavitation ? How can it be avoided in reaction turbine ?
19. Define the terms 'unit power', 'unit speed ' and 'unit discharge' with reference to a hydraulic turbine. Also derive expressions for these terms.
20. (*a*) Define specific speed of a turbine and derive an expression for the same. Show that Pelton turbine is a low specific speed turbine.
    (*b*) What is specific speed ? State its significance in the study of hydraulic machines.
21. (*a*) By means of a neat sketch explain the governing mechanism of Francis Turbine.
    (*b*) Explain the difference between Kaplan turbine and propeller turbine.
22. Define and explain hydraulic efficiency, mechanical efficiency and overall efficiency of a turbine.
23. Define the terms : specific speed of a turbine, unit speed, unit power and unit rate of flow of a turbine. Derive the expressions for specific speed and unit speed.
24. (*a*) What is meant by the speed ratio of a Pelton wheel ?
    (*b*) What is a draft-tube ? What are its functions ?
    (*c*) Differentiate between an inward and an outward flow reaction turbine.

(B) NUMERICAL PROBLEMS

1. A Pelton wheel has a mean bucket speed of 35 m/s with a jet of water flowing at the rate of 1 m^3/s under a head of 270 m. The buckets deflect the jet through an angle of 170° . Calculate the power delivered to the runner and the hydraulic efficiency of the turbine. Assume co-efficient of velocity as 0.98.
   [**Ans.** 2523.8 kW, 95.3%]
2. A Pelton wheel is to be designed for the following specifications. Power = 735.75 kW, S.P. Head = 200 m, Speed = 800 r.p.m. , η_o = 0.86 and jet diameter is not to exceed one-tenth the wheel diameter.
   Determine : (*i*) Wheel diameter, (*ii*) The number of jets required, and (*iii*) Diameter of the jet. Take C_v = 0.98 and speed ratio = 0.45. [**Ans.** (*i*) 0.673 m, (*ii*) 2, (*iii*) 67.3 mm]

3. A Pelton wheel is having a mean bucket diameter of 0.8 m and is running at 1000 r.p.m. The net head on the Pelton wheel is 400 m. If the side clearance angle is 15° and discharge through nozzle is 150 litres/s, find : (*i*) Power available at the nozzle, and (*ii*) Hydraulic efficiency of the turbine.

[**Ans.** (*i*) 588.6 kW, (*ii*) 98%]

4. Two jets strike at buckets of a Pelton wheel, which is having shaft power as 14,715 kW. The diameter of each jet is given as 150 mm. If the net head on the turbine is 500 m, find the overall efficiency of the turbine. Take $C_v = 1.0$. [**Ans.** 85.7%]

5. The following data is related to the Pelton wheel :

Head at the base of the nozzle = 110 m,
Diameter of the jet = 7.5 cm,
Discharge of the nozzle = 200 litres/s,
Shaft power = 191.295 kW
Power absorbed in mechanical resistance = 3.675 kW.

Determine : (*i*) Power lost in nozzle and, (*ii*) Power lost due to hydraulic resistance in the runner.

[**Ans.** (*i*) 10.874 kW, (*ii*) 9.97 kW]

6. Design a Pelton wheel for a head of 80 m and speed 300 r.p.m. The Pelton wheel develops 103 kW S.P. Take $C_v = 0.98$, speed ratio = 0.45 and overall efficiency = 0.80.

[**Ans.** $D = 1.135$ m, $d = 72.6$ mm, size = 36.3×8.7, $Z = 23$]

7. An inward flow reaction turbine has external and internal diameters as 1.2 m and 0.6 m respectively. The velocity of flow through the runner is constant and is equal to 1.8 m/s. Determine : (*i*) Discharge through the runner, and (*ii*) Width at outlet if the width at inlet = 200 mm. [**Ans.** (*i*) 1.357 m^3/s, (*ii*) 400 mm]

8. A reaction turbine works at 500 r.p.m. under a head of 100 m. The diameter of turbine at inlet is 100 cm and flow area is 0.35 m^2. The angles made by absolute and relative velocities at inlet are 15° and 60° respectively with the tangential velocity. Determine :
(*i*) The volume flow rate, (*ii*) The power developed, and (*iii*) Efficiency. Assume whirl at outlet to be zero.

[**Ans.** (*i*) 2.905 m^3/s, (*ii*) 2355.35 kW, (*iii*) 82.6%]

9. An inward flow reaction turbine has an external diameter of 1 m and its breadth at inlet is 200 mm. If the velocity of flow at inlet is 1.5 m/s, find the mass of water passing through the turbine per second. Assume 15% of the area of flow is blocked by blade thickness. If the speed of the runner is 200 r.p.m. and guide blades make an angle of 15° to the wheel tangent, draw the inlet velocity triangle and find : (*i*) The runner vane angle at inlet (*ii*) Velocity of wheel at inlet, (*iii*) The absolute velocity of water leaving the guide vanes, and (*iv*) The relative velocity of water entering the runner blade.

[**Ans.** 1602.2 kg/s, (*i*) 76.19, (*ii*) 10.47 m/s, (*iii*) 11.59 m/s, (*iv*) 3.087 m/s]

10. An outward flow reaction turbine has internal and external diameters of the runner as 0.5 m and 1.0 m respectively. The guide blade angle is 15° and velocity of flow through the runner is constant and equal to 3 m/s. If the speed of the turbine is 250 r.p.m., head on turbine is 10 m and discharge at outlet is radial, determine : (*i*) The runner vane angles at inlet and outlet, (*ii*) Work done by the water on the runner per second per unit weight of water striking per second and (*iii*) Hydraulic efficiency.

[**Ans.** (*i*) 32° 48′, 12° 55′, (*ii*) 7.47 m, (*iii*) 7.47%]

11. A Francis turbine with an overall efficiency of 70% is required to produce 147.15 kW. It is working under a head of 8 m. The peripheral velocity = $0.30\sqrt{2gH}$ and the radial velocity of flow at inlet is $0.96\sqrt{2gH}$. The wheel runs at 200 r.p.m. and the hydraulic losses in the turbine are 20% of the available energy. Assume radial discharge, determine : (*i*) The guide blade angle, (*ii*) The wheel vane angle at inlet, (*iii*) Diameter of the wheel at inlet, and (*iv*) Width of wheel at inlet.

[**Ans.** (*i*) 35° 45′,(*ii*) 42° 54′, (*iii*) 35.9 cm, (*iv*) 19.75 cm]

12. The following data is given for a Francis turbine : Net head = 70 m, speed = 600 r.p.m., shaft power = 367.875 kW, $\eta_o = 85\%$, $\eta_h = 95\%$, flow ratio = 0.25, breadth ratio = 0.1, outer diameter of the runner = 2 × inner diameter of runner. The thickness of vanes occupy 10% of the circumferential area of the

runner. Velocity of flow is constant at inlet and outlet and discharge is radial at outlet. Determine : (*i*) Guide blade angle, (*ii*) Runner vane angles at inlet and outlet, (*iii*) Diameters of runner at inlet and outlet, and (*iv*) Width of wheel at inlet.

[**Ans.** (*i*) 12° 20′, (*ii*) 18° 57′, 50° 17′, (*iii*) .49,.245 m, (*iv*) 49 mm]

13. A Kaplan turbine working under a head of 15 m develops 7357.5 kW shaft power. The outer diameter of the runner is 4 m and hub diameter is 2 m. The guide blade angle at the extreme edge of the runner is 30°. The hydraulic and overall efficiencies of the turbine are 90% and 85% respectively. If the velocity of whirl is zero at outlet, detemine : (*i*) runner vane angles at inlet and outlet at the extreme edge of the runner and (*ii*) speed of the turbine. [**Ans.** (*i*) 103° 10′, 26° 58.5′, (*ii*) 58.5]

14. A Kaplan turbine runner is to be designed to develop 7357.5 kW S.P. The net available head is 10 m. Assume that the speed ratio is 1.8 and flow ratio is 0.6. If the overall efficiency is 70% and diameter of the boss is 0.4 times the diameter of the runner, find the diameter of the runner, its speed and specific speed.

[**Ans.** 4.39 m, 109.63 r.p.m., 528.82]

15. A conical draft-tube having inlet and outlet diameters 0.8 m and 1.2 m discharges water at outlet with a velocity of 3 m/s. The total length of the draft-tube is 8 m and 2 m of the length of draft-tube is immersed in water. If the atmospheric pressure head is 10.3 m of water and loss of head due to friction in the draft-tube is equal to 0.25 times the velocity head at outlet of the tube, find : (*i*) Pressure head at inlet, and (*ii*) Efficiency of the draft-tube. [**Ans.** (*i*) 2.551 m (abs.), (*ii*) 75.3%]

16. A turbine is to operate under a head of 30 m at 300 r.p.m. The discharge is 10 m^3/s. If the efficiency is 90%, determine : (*i*) specific speed of the machine, (*ii*) power generated, and (*iii*) types of the turbine.

[**Ans.** (*i*) 219.9, (*ii*) 2648.7 kW (*iii*) Francis]

17. A turbine develops 7357.5 kW S.P. when running at 200 r.p.m. The head on the turbine is 40 m. If the head on the turbine is reduced to 25 m, determine the speed and power developed by the turbine.

[**Ans.** 158.11, 3635.34 kW]

18. A Pelton wheel is revolving at a speed of 200 r.p.m. and develops 5886 kW S.P. when working under a head of 200 m with an overall efficiency of 80%. Determine unit speed, unit discharge and unit power . The speed ratio for the turbine is given as 0.48. Find the speed, discharge and power when this turbine is working under a head of 150 m.

[**Ans.** 14.14, 0.265 m^3/s, 2.08 kW and 173.2 r.p.m., 3.247 m^3/s, 3823 kW]

19. A Kaplan turbine working under a head of 29 m develops 1287.5 kW S.P. If the speed ratio is equal to 2.1, flow ratio = 0.62, diameter of boss = 0.34 times the diameter of the runner and overall efficiency of the turbine = 89%, find the diameter of the runner and the speed of turbine. [**Ans.** 0.705 m, 1162.3]

20. A Kaplan turbine working under a head of 25 m develops 16000 kW shaft power. The outer diameter of the runner is 4 m and hub diameter is 2 m . The guide blade angle is 35°. The hydraulic and overall efficiency are 90% and 85% respectively. If the velocity of whirl is zero at outlet, determine runner vane angles at inlet and outlet, and speed of turbine. [**Ans.** 35° 27′, 88° 36′, 9.236]

21. A Kaplan turbine develops 9000 kW under a net head of 7.5 m. Mechanical efficiency of the wheel is 86%. The speed ratio based on the outer diameter is 2.2 and the flow ratio is 0.66. Diameter of the boss is 0.35 times the external diameter of the wheel. Determine the diameter of the runner and the specific speed of the runner. (*J.N.T.U., Hyderabad, S 2002*)

[**Hint.** Given : S.P. = 9000 kW; $H = 7.5$ m ; $\eta_m = 86\% = 0.86$; Speed ratio = 2.2 ;
flow ratio = 0.66 ; Dia. of boss = 0.35 × External dia. of wheel *i.e.*, $D_b = 0.35\, D_o$.

$$\frac{u_1}{\sqrt{2gH}} = 2.2 \quad \therefore \quad u_1 = 2.2 \times \sqrt{2gH} = 2.2 \times \sqrt{2 \times 9.81 \times 7.5} = 26.68 \text{ m/s}$$

$$\frac{V_{f_1}}{\sqrt{2gH}} = 0.66 \therefore V_{f_1} = 0.66 \times \sqrt{2gH} = 0.66 \times \sqrt{2 \times 9.81 \times 7.5} = 8 \text{ m/s}$$

Note. In this question either data is incomplete or instead of mechanical efficiency it should be overall efficiency. The question is solved taking the given efficiency as overall efficiency.

Now $\eta_o = 0.86$ But $\eta_o = \dfrac{\text{S.P.}}{\text{W.P.}}$ $\therefore$ $\text{W.P.} = \dfrac{\text{S.P.}}{\eta_o} = \dfrac{9000}{0.86}$ kW

But $\text{W.P.} = \dfrac{\rho \times Q \times g \times H}{1000}$ kW $= \dfrac{1000 \times Q \times 9.81 \times 7.5}{1000} = Q \times 9.81 \times 7.5$ kW

$\therefore Q \times 9.81 \times 7.5 = \dfrac{9000}{0.86}$ or $Q = \dfrac{9000}{0.86} \times \dfrac{1}{9.81 \times 7.5} = 142.237$ m^3/s

But $\text{Q} = \dfrac{\pi}{4}\left[D_o^2 - D_b^2\right] \times V_{f_1}$ $\therefore$ $142.237 = \dfrac{\pi}{4}\left[D_o^2 - D_b^2\right] \times V_{f_1} = \dfrac{\pi}{4}[D_o^2 - (0.35D_o)^2] \times 8$

or $142.237 = \dfrac{\pi}{4} \times 0.8775\, D_o^2 \times 8$ or $D_o = \sqrt{\dfrac{142.237 \times 4}{\pi \times 0.8775 \times 8}} = 5.079$ m $\simeq$ **5 m.**

Specific speed of turbine, $N_s = \dfrac{N\sqrt{\text{S.P.}}}{H^{5/4}}$, where N is obtained from u_1

$\therefore$ $u_1 = \dfrac{\pi D_o N}{60}$ or $26.68 = \dfrac{\pi \times 5 \times N}{60}$ or $N = \dfrac{26.68 \times 60}{\pi \times 5} = 101.91$ r.p.m.

$\therefore$ $N_s = \dfrac{101.91 \times \sqrt{9000}}{7.5^{1.25}} =$ **778.95]**

19 CHAPTER

CENTRIFUGAL PUMPS

▶ 19.1 INTRODUCTION

The hydraulic machines which convert the mechanical energy into hydraulic energy are called pumps. The hydraulic energy is in the form of pressure energy. If the mechanical energy is converted into pressure energy by means of centrifugal force acting on the fluid, the hydraulic machine is called centrifugal pump.

The centrifugal pump acts as a reverse of an inward radial flow reaction turbine. This means that the flow in centrifugal pumps is in the radial outward directions. The centrifugal pump works on the principle of forced vortex flow which means that when a certain mass of liquid is rotated by an external torque, the rise in pressure head of the rotating liquid takes place. The rise in pressure head at any point of the rotating liquid is proportional to the square of tangential velocity of the liquid at that point $\left(i.e.,\text{ rise in pressure head } = \frac{V^2}{2g} \text{ or } \frac{\omega^2 r^2}{2g}\right)$. Thus at the outlet of the impeller, where radius is more, the rise in pressure head will be more and the liquid will be discharged at the outlet with a high pressure head. Due to this high pressure head, the liquid can be lifted to a high level.

▶ 19.2 MAIN PARTS OF A CENTRIFUGAL PUMP

The following are the main parts of a centrifugal pump :

1. Impeller.
2. Casing.
3. Suction pipe with a foot valve and a strainer.
4. Delivery pipe.

All the main parts of the centrifugal pump are shown in Fig. 19.1.

1. Impeller. The rotating part of a centrifugal pump is called 'impeller'. It consists of a series of backward curved vanes. The impeller is mounted on a shaft which is connected to the shaft of an electric motor.

2. Casing. The casing of a centrifugal pump is similar to the casing of a reaction turbine. It is an air-tight passage surrounding the impeller and is designed in such a way that the kinetic energy of the water discharged at the outlet of the impeller is converted into pressure energy before the water leaves the casing and enters the delivery pipe. The following three types of the casings are commonly adopted :

(*a*) Volute casing as shown in Fig. 19.1.
(*b*) Vortex casing as shown in Fig. 19.2 (*a*).
(*c*) Casing with guide blades as shown in Fig. 19.2 (*b*).

(*a*) **Volute Casing.** Fig 19.1 shows the volute casing, which surrounds the impeller. It is of spiral type in which area of flow increases gradually. The increase in area of flow decreases the velocity of flow. The decrease in velocity increases the pressure of the water flowing through the casing. It has been observed that in case of volute casing, the efficiency of the pump increases slightly as a large amount of energy is lost due to the formation of eddies in this type of casing.

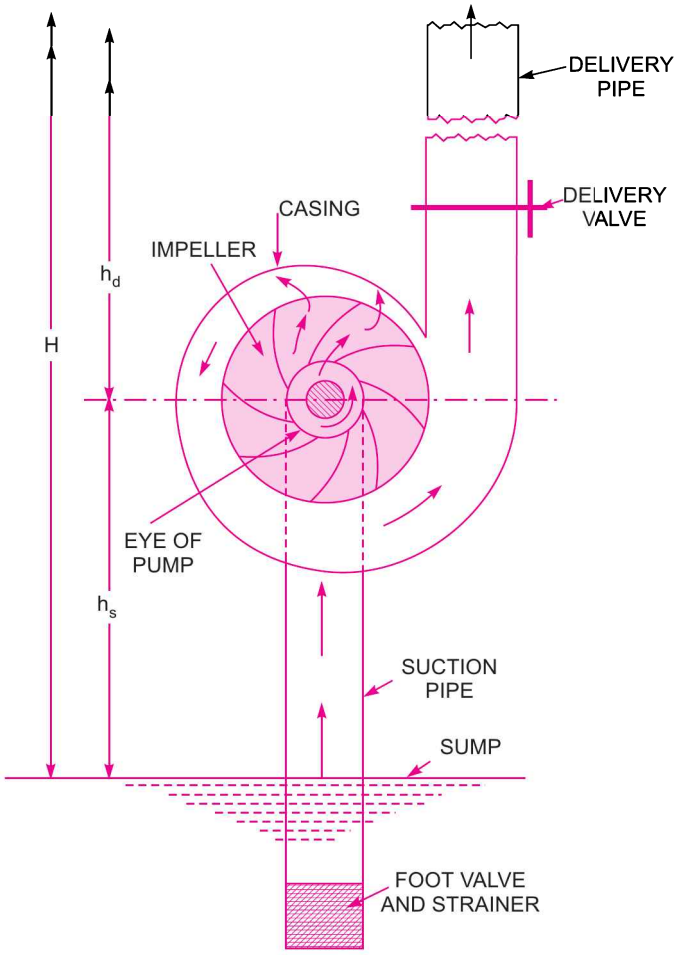


Fig. 19.1 *Main parts of a centrifugal pump.*

(*b*) **Vortex Casing.** If a circular chamber is introduced between the casing and the impeller as shown in Fig. 19.2 (*a*), the casing is known as Vortex Casing. By introducing the circular chamber, the loss of energy due to the formation of eddies is reduced to a considerable extent. Thus the efficiency of the pump is more than the efficiency when only volute casing is provided.

(*c*) **Casing with Guide Blades.** This casing is shown in Fig. 19.2 (*b*) in which the impeller is surrounded by a series of guide blades mounted on a ring which is known as diffuser. The guide vanes are designed in such a way that the water from the impeller enters the guide vanes without stock.

Also the area of the guide vanes increases, thus reducing the velocity of flow through guide vanes and consequently increasing the pressure of water. The water from the guide vanes then passes through the surrounding casing which is in most of the cases concentric with the impeller as shown in Fig. 19.2 (*b*).

3. Suction Pipe with a Foot valve and a Strainer. A pipe whose one end is connected to the inlet of the pump and other end dips into water in a sump is known as suction pipe. A foot valve which is a non-return valve or one-way type of valve is fitted at the lower end of the suction pipe. The foot valve opens only in the upward direction. A strainer is also fitted at the lower end of the suction pipe.

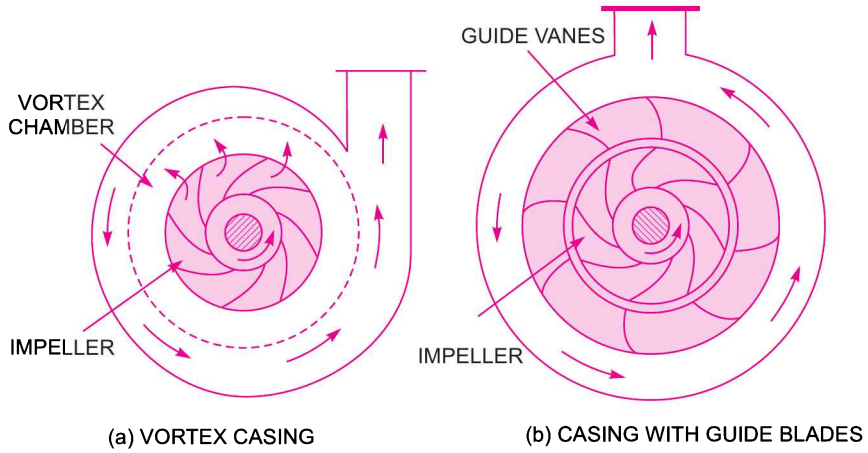


Fig. 19.2 *Different types of casing.*

4. Delivery Pipe. A pipe whose one end is connected to the outlet of the pump and other end delivers the water at a required height is known as delivery pipe.

▶ 19.3 WORK DONE BY THE CENTRIFUGAL PUMP (OR BY IMPELLER) ON WATER

In case of the centrifugal pump, work is done by the impeller on the water. The expression for the work done by the impeller on the water is obtained by drawing velocity triangles at inlet and outlet of the impeller in the same way as for a turbine. The water enters the impeller radially at inlet for best efficiency of the pump, which means the absolute velocity of water at inlet makes an angle of 90° with the direction of motion of the impeller at inlet. Hence angle $\alpha = 90°$ and $V_{w_1} = 0$. For drawing the velocity triangles, the same notations are used as that for turbines. Fig. 19.3 shows the velocity triangles at the inlet and outlet tips of the vanes fixed to an impeller.

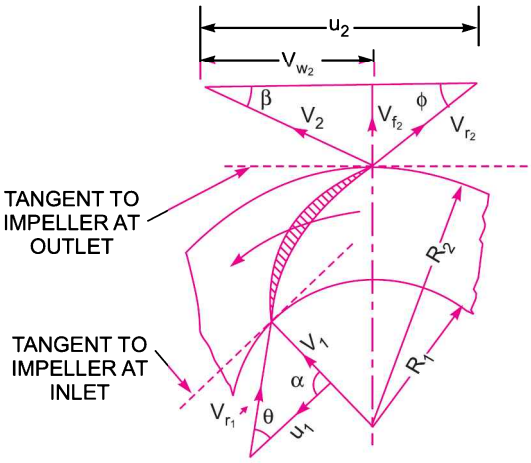


Fig. 19.3 *Velocity triangles at inlet and outlet.*

Let N = Speed of the impeller in r.p.m.,

D_1 = Diameter of impeller at inlet,

u_1 = Tangential velocity of impeller at inlet,

$$= \frac{\pi D_1 N}{60}$$

D_2 = Diameter of impeller at outlet,

u_2 = Tangential velocity of impeller at outlet

$$= \frac{\pi D_2 N}{60}$$

V_1 = Absolute velocity of water at inlet,

V_{r_1} = Relative velocity of water at inlet,

α = Angle made by absolute velocity (V_1) at inlet with the direction of motion of vane,

θ = Angle made by relative velocity (V_{r_1}) at inlet with the direction of motion of vane, and V_2, V_{r_2}, β and ϕ are the corresponding values at outlet.

As the water enters the impeller radially which means the absolute velocity of water at inlet is in the radial direction and hence angle $\alpha = 90°$ and $V_{w_1} = 0$.

A centrifugal pump is the reverse of a radially inward flow reaction turbine. But in case of a radially inward flow reaction turbine, the work done by the water on the runner per second per unit weight of the water striking per second is given by equation (18.19) as

$$= \frac{1}{g}[V_{w_1}u_1 - V_{w_2}u_2]$$

$\therefore$ Work done by the impeller on the water per second per unit weight of water striking per second

$$= - [\text{Work done in case of turbine}]$$

$$= -\left[\frac{1}{g}\left(V_{w_1}u_1 - V_{w_2}u_2\right)\right] = \frac{1}{g}\left[V_{w_2}u_2 - V_{w_1}u_1\right]$$

$$= \frac{1}{g}V_{w_2}u_2 \qquad (\because \quad V_{w_1} = 0 \text{ here}) \quad ...(19.1)$$

Work done by impeller on water per second

$$= \frac{W}{g}.V_{w_2}u_2 \qquad ...(19.2)$$

where W = Weight of water = $\rho \times g \times Q$

where Q = Volume of water

and Q = Area × Velocity of flow = $\pi D_1 B_1 \times V_{f_1}$

$$= \pi D_2 B_2 \times V_{f_2} \qquad ...(19.2A)$$

where B_1 and B_2 are width of impeller at inlet and outlet and V_{f_1} and V_{f_2} are velocities of flow at inlet and outlet.

Equation (19.1) gives the head imparted to the water by the impeller or energy given by impeller to water per unit weight per second.

▶ 19.4 DEFINITIONS OF HEADS AND EFFICIENCIES OF A CENTRIFUGAL PUMP

1. Suction Head (h_s). It is the vertical height of the centre line of the centrifugal pump above the water surface in the tank or pump from which water is to be lifted as shown in Fig. 19.1. This height is also called suction lift and is denoted by 'h_s'.

2. Delivery Head (h_d). The vertical distance between the centre line of the pump and the water surface in the tank to which water is delivered is known as delivery head. This is denoted by 'h_d'.

3. Static Head (H_s). The sum of suction head and delivery head is known as static head. This is represented by 'H_s' and is written as

$$H_s = h_s + h_d. \qquad ...(19.3)$$

4. Manometric Head (H_m). The manometric head is defined as the head against which a centrifugal pump has to work. It is denoted by 'H_m'. It is given by the following expressions :

(*a*) H_m = Head imparted by the impeller to the water – Loss of head in the pump

$$= \frac{V_{w_2} u_2}{g} - \text{Loss of head in impeller and casing} \qquad ...(19.4)$$

$$= \frac{V_{w_2} u_2}{g} \quad ...\text{if loss of pump is zero} \qquad ...(19.5)$$

(*b*) H_m = Total head at outlet of the pump – Total head at the inlet of the pump

$$= \left(\frac{P_o}{\rho g} + \frac{V_o^2}{2g} + Z_o\right) - \left(\frac{p_i}{\rho g} + \frac{V_i^2}{2g} + Z_i\right) \qquad ...(19.6)$$

where $\frac{p_o}{\rho g}$ = Pressure head at outlet of the pump = h_d

$\frac{V_o^2}{2g}$ = Velocity head at outlet of the pump

= Velocity head in delivery pipe = $\frac{V_d^2}{2g}$

Z_o = Vertical height of the outlet of the pump from datum line, and

$\frac{p_i}{\rho g}, \frac{V_i^2}{2g}, Z_i$ = Corresponding values of pressure head, velocity head and datum head at the inlet of the pump,

i.e., $h_s, \frac{V_s^2}{2g}$ and Z_s respectively.

(*c*) $$H_m = h_s + h_d + h_{f_s} + h_{f_d} + \frac{V_d^2}{2g} \qquad ...(19.7)$$

where h_s = Suction head, h_d = Delivery head,

h_{f_s} = Frictional head loss in suction pipe, h_{f_d} = Frictional head loss in delivery pipe, and

V_d = Velocity of water in delivery pipe.

5. Efficiencies of a Centrifugal Pump. In case of a centrifugal pump, the power is transmitted from the shaft of the electric motor to the shaft of the pump and then to the impeller. From the impeller, the power is given to the water. Thus power is decreasing from the shaft of the pump to the impeller and then to the water. The following are the important efficiencies of a centrifugal pump :

(*a*) Manometric efficiency, η_{man} (*b*) Mechanical efficiency, η_m and

(*c*) Overall efficiency, η_o.

(*a*) **Manometric Efficiency (η_{man}).** The ratio of the manometric head to the head imparted by the impeller to the water is known as manometric efficiency. Mathematically, it is written as

$$\eta_{man} = \frac{\text{Manometric head}}{\text{Head imparted by impeller to water}}$$

$$= \frac{H_m}{\left(\frac{V_{w_2} u_2}{g}\right)} = \frac{gH_m}{V_{w_2} u_2} \qquad ...(19.8)$$

The power at the impeller of the pump is more than the power given to the water at outlet of the pump. The ratio of the power given to water at outlet of the pump to the power available at the impeller, is known as manometric efficiency.

The power given to water at outlet of the pump = $\frac{WH_m}{1000}$ kW

The power at the impeller $= \frac{\text{Work done by impeller per second}}{1000}$ kW

$$= \frac{W}{g} \times \frac{V_{w_2} \times u_2}{1000} \text{kW}$$

$$\eta_{man} = \frac{\frac{W \times H_m}{1000}}{\frac{W}{g} \times \frac{V_{w_2} \times u_2}{1000}} = \frac{g \times H_m}{V_{w_2} \times u_2}.$$

(*b*) **Mechanical Efficiency (η_m).** The power at the shaft of the centrifugal pump is more than the power available at the impeller of the pump. The ratio of the power available at the impeller to the power at the shaft of the centrifugal pump is known as mechanical efficiency. It is written as

$$\eta_m = \frac{\text{Power at the impeller}}{\text{Power at the shaft}}$$

The power at the impeller in kW $= \frac{\text{Work done by impeller per second}}{1000}$

$$= \frac{W}{g} \times \frac{V_{w_2} u_2}{1000} \qquad \text{[Using equation (19.2)]}$$

$$\eta_m = \frac{\frac{W}{g}\left(\frac{V_{w_2} u_2}{1000}\right)}{\text{S.P.}} \qquad ...(19.9)$$

where S.P. = Shaft power.

(*c*) **Overall Efficiency (η_o).** It is defined as ratio of power output of the pump to the power input to the pump. The power output of the pump in kW

$$= \frac{\text{Weight of water lifted} \times H_m}{1000} = \frac{WH_m}{1000}$$

Power input to the pump = Power supplied by the electric motor

= S.P. of the pump.

$\therefore$ $$\eta_o = \frac{\left(\frac{WH_m}{1000}\right)}{\text{S.P.}} \qquad ...(19.10)$$

Also $$\eta_o = \eta_{man} \times \eta_m. \qquad ...(19.11)$$

Problem 19.1 *The internal and external diameters of the impeller of a centrifugal pump are 200 mm and 400 mm respectively. The pump is running at 1200 r.p.m. The vane angles of the impeller at inlet and outlet are 20° and 30° respectively. The water enters the impeller radially and velocity of flow is constant. Determine the work done by the impeller per unit weight of water.*

Solution. Given :

Internal diameter of impeller, $D_1 = 200$ mm $= 0.20$ m

External diameter of impeller, $D_2 = 400$ mm $= 0.40$ m

Speed, $N = 1200$ r.p.m.

Vane angle at inlet, $\theta = 20°$

Vane angle at outlet, $\phi = 30°$

Water enters radially* means, $\alpha = 90°$ and $V_{w_1} = 0$

Velocity of flow, $V_{f_1} = V_{f_2}$

Tangential velocity of impeller at inlet and outlet are,

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 0.20 \times 1200}{60} = 12.56 \text{ m/s}$$

and $$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.4 \times 1200}{60} = 25.13 \text{ m/s}.$$

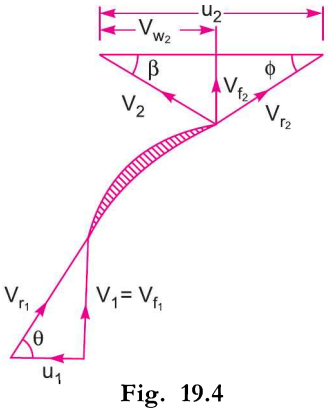


Fig. 19.4

From inlet velocity triangle, $\tan\theta = \frac{V_{f_1}}{u_1} = \frac{V_{f_1}}{12.56}$

$\therefore \quad V_{f_1} = 12.56 \tan\theta = 12.56 \times \tan 20° = 4.57$ m/s

$\therefore \quad V_{f_2} = V_{f_1} = 4.57$ m/s.

From outlet velocity triangle, $\tan\phi = \frac{V_{f_2}}{u_2 - V_{w_2}} = \frac{4.57}{25.13 - V_{w_2}}$

or $$25.13 - V_{w_2} = \frac{4.57}{\tan\phi} = \frac{4.57}{\tan 30°} = 7.915$$

$\therefore \quad V_{w_2} = 25.13 - 7.915 = 17.215$ m/s.

The work done by impeller per kg of water per second is given by equation (19.1) as

$$= \frac{1}{g} V_{w_2} u_2 = \frac{17.215 \times 25.13}{9.81} = \mathbf{44.1 \text{ Nm/N. Ans.}}$$

Problem 19.2 *A centrifugal pump is to discharge 0.118 m^3/s at a speed of 1450 r.p.m. against a head of 25 m. The impeller diameter is 250 mm, its width at outlet is 50 mm and manometric efficiency is 75%. Determine the vane angle at the outer periphery of the impeller.*

Solution. Given :

Discharge, $Q = 0.118$ m^3/s

Speed, $N = 1450$ r.p.m.

Head, $H_m = 25$ m

Diameter at outlet, $D_2 = 250$ mm $= 0.25$ m

Width at outlet, $B_2 = 50$ mm $= 0.05$ m

Manometric efficiency, $\eta_{man} = 75\% = 0.75$.

Let vane angle at outlet $= \phi$

Tangential velocity of impeller at outlet,

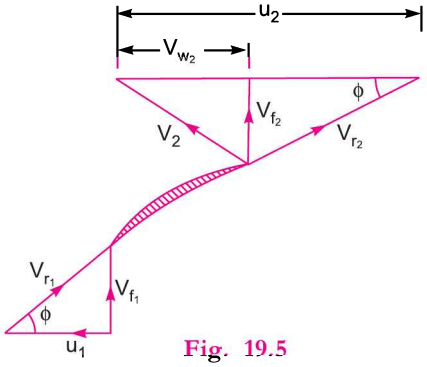


Fig. 19.5

* If in the problem, this condition is not given even then the water is assumed to be entering radially unless stated otherwise

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.25 \times 1450}{60} = 18.98 \text{ m/s}$$

Discharge is given by $Q = \pi D_2 B_2 \times V_{f_2}$

$$\therefore \quad V_{f_2} = \frac{Q}{\pi D_2 B_2} = \frac{0.118}{\pi \times 0.25 \times .05} = 3.0 \text{ m/s.}$$

Using equation (19.8), $\eta_{man} = \dfrac{gH_m}{V_{w_2} u_2} = \dfrac{9.81 \times 25}{V_{w_2} \times 18.98}$

$$\therefore \quad V_{w_2} = \frac{9.81 \times 25}{\eta_{man} \times 18.98} = \frac{9.81 \times 25}{0.75 \times 18.98} = 17.23.$$

From outlet velocity triangle, we have

$$\tan \phi = \frac{V_{f_2}}{(u_2 - V_{w_2})} = \frac{3.0}{(18.98 - 17.23)} = 1.7143$$

$$\therefore \quad \phi = \tan^{-1} 1.7143 = 59.74° \text{ or } \mathbf{59° 44'. Ans.}$$

Problem 19.3 *A centrifugal pump delivers water against a net head of 14.5 metres and a design speed of 1000 r.p.m. The vanes are curved back to an angle of 30° with the periphery. The impeller diameter is 300 mm and outlet width is 50 mm. Determine the discharge of the pump if manometric efficiency is 95%.*

Solution. Given :

Net head, $H_m = 14.5$ m

Speed, $N = 1000$ r.p.m.

Vane angle at outlet, $\phi = 30°$

Impeller diameter means the diameter of the impeller at outlet

$\therefore$ Diameter, $D_2 = 300$ mm $= 0.30$ m

Outlet width, $B_2 = 50$ mm $= 0.05$ m

Manometric efficiency, $\eta_{man} = 95\% = 0.95$

Tangential velocity of impeller at outlet,

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.30 \times 1000}{60} = 15.70 \text{ m/s.}$$

Now using equation (19.8), $\eta_{man} = \dfrac{gH_m}{V_{w_2} \times u_2}$

$$\therefore \quad 0.95 = \frac{9.81 \times 14.5}{V_{w_2} \times 15.70}$$

$$\therefore \quad V_{w_2} = \frac{0.95 \times 14.5}{0.95 \times 15.70} = 9.54 \text{ m/s.}$$

Refer to Fig. 19.5. From outlet velocity triangle, we have

$$\tan \phi = \frac{V_{f_2}}{(u_2 - V_{w_2})} \text{ or } \tan 30° = \frac{V_{f_2}}{(15.70 - 9.54)} = \frac{V_{f_2}}{6.16}$$

$$\therefore \quad V_{f_2} = 6.16 \times \tan 30° = 3.556 \text{ m/s.}$$

$\therefore$ Discharge, $Q = \pi D_2 B_2 \times V_{f_2}$

$= \pi \times 0.30 \times 0.05 \times 3.556 \text{ m}^3/\text{s} = \mathbf{0.1675 \text{ m}^3/\text{s}}$. **Ans.**

Problem 19.4 *A centrifugal pump having outer diameter equal to two times the inner diameter and running at 1000 r.p.m. works against a total head of 40 m. The velocity of flow through the impeller is constant and equal to 2.5 m/s. The vanes are set back at an angle of 40° at outlet. If the outer diameter of the impeller is 500 mm and width at outlet is 50 mm, determine* :

(*i*) *Vane angle at inlet,* (*ii*) *Work done by impeller on water per second, and*

(*iii*) *Manometric efficiency.*

Solution. Given :

Speed, $N = 1000$ r.p.m.

Head, $H_m = 40$ m

Velocity of flow, $V_{f_1} = V_{f_2} = 2.5$ m/s

Vane angle at outlet, $\phi = 40°$

Outer dia. of impeller, $D_2 = 500$ mm $= 0.50$ m

Inner dia. of impeller, $D_1 = \frac{D_2}{2} = \frac{0.50}{2} = 0.25$ m

Width at outlet, $B_2 = 50$ mm $= 0.05$ m

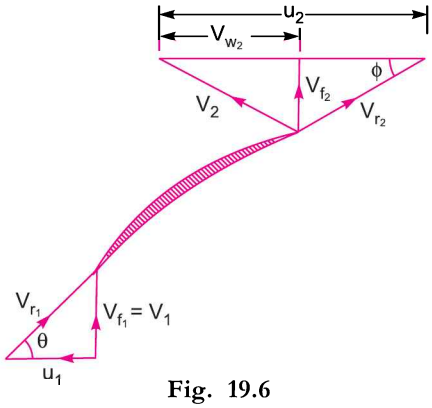


Fig. 19.6

Tangential velocity of impeller at inlet and outlet are

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 0.25 \times 1000}{60} = 13.09 \text{ m/s}$$

and $$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.50 \times 1000}{60} = 26.18 \text{ m/s.}$$

Discharge is given by, $Q = \pi D_2 B_2 \times V_{f_2} = \pi \times 0.50 \times .05 \times 2.5 = 0.1963 \text{ m}^3/\text{s}$.

(*i*) **Vane angle at inlet** (θ).

From inlet velocity triangle $\tan \theta = \frac{V_{f_1}}{u_1} = \frac{2.5}{13.09} = 0.191$

$\therefore$ $\theta = \tan^{-1} .191 = 10.81°$ or **10° 48′. Ans.**

(*ii*) **Work done by impeller on water per second** is given by equation (19.2) as

$$= \frac{W}{g} \times V_{w_2} u_2 = \frac{\rho \times g \times Q}{g} \times V_{w_2} \times u_2$$

$$= \frac{1000 \times 9.81 \times 0.1963}{9.81} \times V_{w_2} \times 26.18 \qquad \ldots(i)$$

But from outlet velocity triangle, we have

$$\tan \phi = \frac{V_{f_2}}{u_2 - V_{w_2}} = \frac{2.5}{(26.18 - V_{w_2})}$$

$\therefore$ $$26.18 - V_{w_2} = \frac{2.5}{\tan \phi} = \frac{2.5}{\tan 40°} = 2.979$$

$\therefore$ $$V_{w_2} = 26.18 - 2.979 = 23.2 \text{ m/s.}$$

Substituting this value of V_{w_2} in equation (*i*), we get the work done by impeller as

$$= \frac{1000 \times 9.81 \times 0.1963}{9.81} \times 23.2 \times 26.18$$

$$= \mathbf{119227.9 \text{ Nm/s. Ans.}}$$

(*iii*) **Manometric efficiency (η_{man}).** Using equation (19.8), we have

$$\eta_{man} = \frac{gH_m}{V_{w_2} u_2} = \frac{9.81 \times 40}{23.2 \times 26.18} = 0.646 = \mathbf{64.4\%}. \textbf{ Ans.}$$

Problem 19.5 *A centrifugal pump discharges 0.15 m^3/s of water against a head of 12.5 m, the speed of the impeller being 600 r.p.m. The outer and inner diameters of impeller are 500 mm and 250 mm respectively and the vanes are bent back at 35° to the tangent at exit. If the area of flow remains 0.07 m^2 from inlet to outlet, calculate :*

(i) Manometric efficiency of pump, (ii) Vane angle at inlet, and

(iii) Loss of head at inlet to impeller when the discharge is reduced by 40% without changing the speed.

Solution. Given :

Discharge, $Q = 0.15$ m^3/s

Head, $H_m = 12.5$ m

Speed, $N = 600$ r.p.m.

Outer dia., $D_2 = 500$ mm $= 0.50$ m

Inner dia., $D_1 = 250$ mm $= 0.25$ m

Vane angle at outlet, $\phi = 35°$

Area of flow, $= 0.07$ m^2

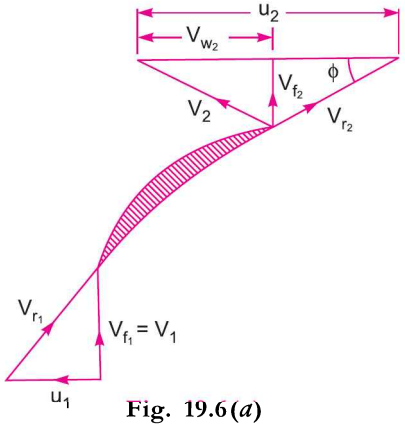


Fig. 19.6(*a*)

As area of flow is constant from inlet to outlet, then velocity of flow will be constant from inlet to outlet.

Discharge = Area of flow × Velocity of flow

or $0.15 = 0.07 \times$ Velocity of flow

$$\therefore \quad \text{Velocity of flow} = \frac{0.15}{0.07} = 2.14 \text{ m/s.}$$

$$\therefore \quad V_{f_1} = V_{f_2} = 2.14 \text{ m/s.}$$

Tangential velocity of impeller at inlet and outlet are

$$u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 0.25 \times 600}{60} = 7.85 \text{ m/s}$$

and

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.50 \times 600}{60} = 15.70 \text{ m/s}$$

From outlet velocity triangle, $V_{w_2} = u_2 - \frac{V_{f_2}}{\tan \phi} = 15.70 - \frac{2.14}{\tan 35°} = 12.64$ m/s

(*i*) *Manometric efficiency of the pump*

Using equation (19.8), we have $\eta_{man} = \frac{g \times H_m}{V_{w_2} \times u_2} = \frac{9.81 \times 12.5}{12.64 \times 15.7} = 0.618$ or **61.8%. Ans.**

(*ii*) *Vane angle at inlet* (θ)

From inlet velocity triangle, $\tan \theta = \frac{V_{f_1}}{u_1} = \frac{2.14}{7.85} = 0.272$

$$\therefore \quad \theta = \tan^{-1} 0.272 = \mathbf{15° 12'}. \textbf{ Ans.}$$

(*iii*) *Loss of head at inlet to impeller when discharge is reduced by 40% without changing the speed.*

When there is an increase or decrease in the discharge from the normal discharge, a loss of head occurs at entry due to shock. In this case, discharge is reduced by 40%. Hence the new discharge is given by,

$$Q^* = 0.6 \times Q$$

where $Q = 0.15$ m^3/s

As area of flow is constant, hence new velocity of flow ($V_{f_1}{}^*$) will be given by,

$$V_{f_1}{}^* = \frac{Q^*}{\text{Area of flow}}$$

$$= \frac{0.6 \times Q}{0.07} = \frac{0.6 \times 0.15}{0.07} = 1.284 \text{ m/s.}$$

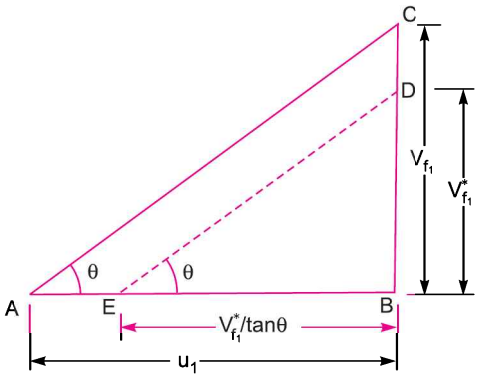


Fig. 19.6 (*b*)

Fig. 19.6 (*b*) shows the velocity triangle at inlet corresponding to normal discharge and reduced discharge. *ABC* is the velocity triangle due to normal discharge. Triangle *BDE* is corresponding to reduced discharge *BD* = 1.284 m/s and *DE* is parallel to *AC*.

The blade angle θ at inlet cannot change and hence *DE* will be parallel to *AC*.

There will be a sudden change in the tangential velocity from *AB* to *BE*. Hence due to this shock, there will be a loss of head at inlet.

$$\therefore \quad \text{Head lost at inlet} = \frac{(\text{change in tangential velocity at inlet})^2}{2g}$$

$$= \frac{(AB - BE)^2}{2g} = \frac{\left(u_1 - \dfrac{V_{f_1}^*}{\tan\theta}\right)^2}{2g} = \frac{\left(7.85 - \dfrac{1.284}{\tan 15.2°}\right)^2}{2 \times 9.81} = \mathbf{0.5 \text{ m. Ans.}}$$

Problem 19.6 *The outer diameter of an impeller of a centrifugal pump is 400 mm and outlet width is 50 mm. The pump is running at 800 r.p.m. and is working against a total head of 15 m. The vanes angle at outlet is 40° and manometric efficiency is 75%. Determine :*

(i) velocity of flow at outlet, (ii) velocity of water leaving the vane,

(iii) angle made by the absolute velocity at outlet with the direction of motion at outlet, and

(iv) discharge.

Solution. Given :

Outer diameter, $D_2 = 400$ mm$= 0.4$ m

Width at outlet, $B_2 = 50$ mm $= 0.05$ m

Speed, $N = 800$ r.p.m.

Head, $H_m = 15$ m

Vane angle at outlet, $\phi = 40°$

Manometric efficiency, $\eta_{man} = 75\% = 0.75$

Tangential velocity of impeller at outlet,

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.4 \times 800}{60} = 16.75 \text{ m/s.}$$

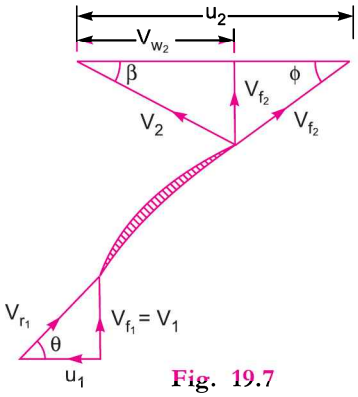


Fig. 19.7

Using equation (19.8), $\eta_{man} = \dfrac{gH_m}{V_{w_2}\, u_2}$

$$0.75 = \frac{9.81 \times 15}{V_{w_2} \times 16.75}$$

$$\therefore \quad V_{w_2} = \frac{9.81 \times 15}{0.75 \times 16.75} = 11.71 \text{ m/s.}$$

From the outlet velocity triangle, we have

$$\tan \phi = \frac{V_{f_2}}{u_2 - V_{w_2}} = \frac{V_{f_2}}{(16.75 - 11.71)} = \frac{V_{f_2}}{5.04}$$

(*i*) $\therefore \quad V_{f_2} = 5.04 \tan \phi = 5.04 \times \tan 40° =$ **4.23 m/s. Ans.**

(*ii*) **Velocity of water leaving the vane** (V_2).

$$V_2 = \sqrt{V_{f_2}^2 + V_{w_2}^2} = \sqrt{4.23^2 + 11.71^2}$$

$$= \sqrt{17.89 + 137.12} = \textbf{12.45 m/s. Ans.}$$

(*iii*) **Angle made by absolute velocity at outlet** (β),

$$\tan \beta = \frac{V_{f_2}}{V_{w_2}} = \frac{4.23}{11.71} = 0.36$$

$$\therefore \quad \beta = \tan^{-1} 0.36 = 19.80° \text{ or } \textbf{19° 48′. Ans.}$$

(*iv*) **Discharge** through pump is given by,

$$Q = \pi D_2 B_2 \times V_{f_2} = \pi \times 0.4 \times 0.05 \times 4.23 = \textbf{0.265 m}^3\textbf{/s. Ans.}$$

Problem 19.7 *A centrifugal pump is running at 1000 r.p.m. The outlet vane angle of the impeller is 45° and velocity of flow at outlet is 2.5 m/s. The discharge through the pump is 200 litres/s when the pump is working against a total head of 20 m. If the manometric efficiency of the pump is 80%, determine :*

(*i*) *the diameter* of the impeller, and* (*ii*) *the width of the impeller at outlet.*

Solution. Given :

Speed, $N = 1000$ r.p.m.

Outlet vane angle, $\phi = 45°$

Velocity of flow at outlet, $V_{f_2} = 2.5$ m/s

Discharge, $Q = 200$ litres/s $= 0.2$ m^2/s

Head, $H_m = 20$ m

Manometric efficiency, $\eta_{man} = 80\% = 0.80$

From outlet velocity triangle, we have

$$\tan \phi = \frac{V_{f_2}}{u_2 - V_{w_2}}$$

or $$u_2 - V_{w_2} = \frac{V_{f_2}}{\tan \phi} = \frac{2.5}{\tan 45} = 2.5$$

$$\therefore \quad V_{w_2} = (u_2 - 2.5) \qquad \ldots(i)$$

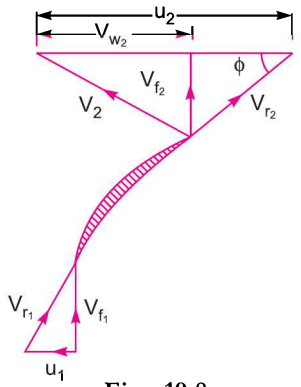


Fig. 19.8

* Diameter of impeller means the outside diameter.

Using equation (19.8), $\eta_{man} = \dfrac{gH_m}{V_{w_2} u_2}$

$$0.80 = \frac{9.81 \times 20}{V_{w_2} u_2}$$

$$\therefore \quad V_{w_2} u_2 = \frac{9.81 \times 20}{0.80} = 245.25 \qquad ...(ii)$$

Substituting the value of V_{w_2} from equation (i) in (ii), we get

$$(u_2 - 2.5)u_2 = 245.25$$

$$u_2^2 - 2.5u_2 - 245.25 = 0$$

which is a quadratic equation in u_2 and its solution is

$$u_2 = \frac{2.5 \pm \sqrt{(2.5)^2 + 4 \times 245.25}}{2} = \frac{2.5 + \sqrt{6.25 + 981}}{2}$$

$$= \frac{2.5 \pm 31.42}{2} = 16.96 \text{ or } -14.46$$

$$\therefore \quad u_2 = 16.96 \qquad (\because \text{ –ve value is not possible})$$

(i) *Diameter of impeller* (D_2).

Using, $u_2 = \dfrac{\pi D_2 N}{60}$

$$\therefore \quad 16.96 = \frac{\pi D_2 N}{60} = \frac{\pi \times D_2 \times 1000}{60}$$

$$\therefore \quad D_2 = \frac{16.96 \times 60}{\pi \times 1000} = 0.324 \text{ m} = \textbf{324 mm. Ans.}$$

(ii) *Width of impeller at outlet* (B_2).

Discharge, $Q = \pi D_2 B_2 V_{f_2}$

$$0.2 = \pi \times .324 \times B_2 \times 2.5$$

$$\therefore \quad B_2 = \frac{0.2}{\pi \times .324 \times 2.5} = 0.0786 \text{ m} = \textbf{78.6 mm. Ans.}$$

Problem 19.7 (A) *A centrifugal pump has the following dimensions : inlet radius = 80 mm ; outlet radius = 160 mm ; width of impeller at the inlet = 50 mm ;* $\beta_1 = 0.45$ *radians ;* $\beta_2 = 0.25$ *radians ; width of impeller at outlet = 50 mm.*

Assuming shockless entry determine the discharge and the head developed by the pump when the impeller rotates at 90 radians/second.

Solution. Given :

Inlet radius, $R_1 = 80 \text{ mm} = 0.08 \text{ m}$

Outlet radius $R_2 = 160 \text{ mm} = 0.16 \text{ m}$

Width at inlet, $B_1 = 50 \text{ mm} = 0.05 \text{ m}$

Width at outlet, $B_2 = 50 \text{ mm} = 0.05 \text{ m}$

Angles, $\beta_1 = 0.45$ radians and $\beta_2 = 0.25$ radians.

Here β_1 is the vane angle at inlet and β_2 is the vane angle at outlet.

∴ Vane angle at inlet, $\theta = \beta_1 = 0.45$ radians

Vane angle at outlet, $\phi = \beta_2 = 0.25$ radians.

Angular velocity, $\omega = 90$ rad/s

Find :

(*i*) discharge, and (*ii*) head developed.

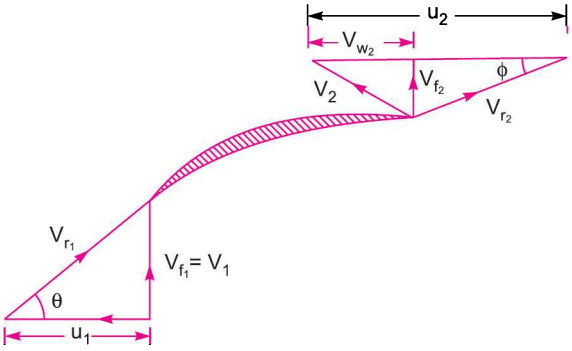


Fig. 19.8 (*a*)

Now tangential velocity of impeller at inlet and outlet are,

$$u_1 = \frac{\pi D_1 \times N}{60} = \frac{2\pi N}{60} \times \frac{D_1}{2} = \omega \times R_1 = 90 \times 0.08 = 7.2 \text{ m/s}$$

and $$u_2 = \omega \times R_2 = 90 \times 0.16 = 14.4 \text{ m/s}$$

From inlet velocity triangle, $\tan\theta = \dfrac{V_{f_1}}{u_1}$

∴ $$V_{f_1} = u_1 \times \tan\theta = 7.2 \times \tan(0.45 \text{ rad}) = 7.2 \times 0.483 = 3.478 \text{ m/s}$$

(*i*) *Discharge* (Q)

Discharge is given by, $Q = \pi D_1 \times B_1 \times V_{f_1} = \pi \times (2R_1) \times B_1 \times V_{f_1}$

$= \pi \times 2 \times 0.08 \times 0.05 \times 3.478 \text{ m}^3/\text{s} =$ **0.0874 m^3/s. Ans.**

(*ii*) *Head developed* (H_m)

For the shockless entry, the losses of the pump will be zero. Hence, the head developed (H_m) will be given by equation (19.5).

∴ $$H_m = \frac{V_{w_2} \times u_2}{g} \quad \ldots(i)$$

where from outlet velocity triangle, $V_{w_2} = u_2 - V_{f_2} \times \cot\phi$

The value of V_{f_2} is obtained from $Q = \pi D_2 \times B_2 \times V_{f_2}$

or $$0.0874 = \pi \times (2R_2) \times B_2 \times V_{f_2}$$

$$= \pi \times (2 \times 0.16) \times 0.05 \times V_{f_2}$$

∴ $$V_{f_2} = \frac{0.0874}{\pi \times 2 \times 0.16 \times 0.05} = 1.7387 \text{ m/s}$$

∴ $$V_{w_2} = u_2 - V_{f_2} \times \cot\phi$$

$= 14.4 - 1.7387 \times \cot (0.25 \text{ radians})$

$= 14.4 - 1.7387 \times 3.9163 = 14.4 - 6.809 = 7.591$ m/s

Substituting this value in equation (*i*) above, we get

$$H_m = \frac{V_{w_2} \times u_2}{g} = \frac{7.591 \times 14.4}{9.81} = \textbf{11.142 m. Ans.}$$

Problem 19.8 *The internal and external diameter of an impeller of a centrifugal pump which is running at 1000 r.p.m., are 200 mm and 400 mm respectively. The discharge through pump is 0.04 m^3/s and velocity of flow is constant and equal to 2.0 m/s. The diameters of the suction and delivery pipes are 150 mm and 100 mm respectively and suction and delivery heads are 6 m (abs.) and 30 m (abs.) of water respectively. If the outlet vane angle is 45° and power required to drive the pump is 16.186 kW, determine :*

(i) Vane angle of the impeller at inlet, (ii) The overall efficiency of the pump, and

(iii) Manometric efficiency of the pump.

Solution. Given :

Speed,	$N = 1000$ r.p.m.
Internal dia.,	$D_1 = 200$ mm $= 0.2$ m
External dia.,	$D_2 = 400$ mm $= 0.4$ m
Discharge,	$Q = 0.04$ m^3/s
Velocity of flow,	$V_{f_1} = V_{f_2} = 2.0$ m/s
Dia. of suction pipe,	$D_s = 150$ mm $= 0.15$ m
Dia. of delivery pipe,	$D_d = 100$ mm $= 0.10$ m
Suction head,	$h_s = 6$ m (abs.)
Delivery head,	$h_d = 30$ m (abs.)
Outlet vane angle,	$\phi = 45°$
Power required to drive the pump,	$P = 16.186$/ kW

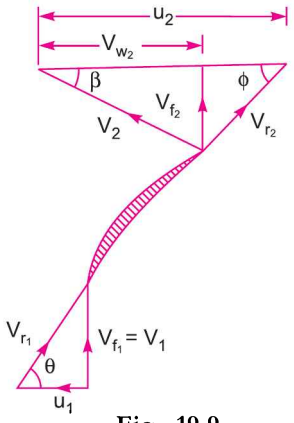


Fig. 19.9

(*i*) Vane angle of the impeller at inlet (θ).

From inlet velocity, we have $\tan \theta = \frac{V_{f_1}}{u_1} = \frac{2.0}{u_1}$, where $u_1 = \frac{\pi D_1 N}{60} = \frac{\pi \times 0.2 \times 1000}{60} = 10.47$ m/s

$$\therefore \quad \tan \theta = \frac{2.0}{10.47} = 0.191 \text{ or } \theta = \tan^{-1} .191 = \textbf{10° 48′. Ans.}$$

(*ii*) Overall efficiency of the pump (η_o).

Using equation (19.10), we have $\eta_o = \dfrac{\left(\frac{WH_m}{1000}\right)}{\text{S.P.}}$

where S.P. = Power required to drive the pump and equal to P here.

$$\eta_o = \frac{\left(\frac{\rho \times g \times Q \times H_m}{1000}\right)}{P} = \frac{\rho g \times Q \times H_m}{1000 \times P}$$

$$= \frac{1000 \times 9.81 \times .04 \times H_m}{1000 \times 16.186} = 0.02424 \, H_m \qquad \ldots(i)$$

Now H_m is given by equation (19.6) as

$$H_m = \left(\frac{p_o}{\rho g} + \frac{V_o^2}{2g} + Z_o\right) - \left(\frac{p_i}{\rho g} + \frac{V_i^2}{2g} + Z_i\right) \quad ...(ii)$$

where $\frac{p_o}{\rho g}$ = Pressure head at outlet of pump = h_d = 30 m

$\frac{V_o^2}{2g}$ = Velocity head at outlet of pump = $\frac{V_d^2}{2g}$

$\frac{p_i}{\rho g}$ = Pressure head at inlet of pump = h_s = 6 m

$\frac{V_i^2}{2g}$ = Velocity head at inlet of pump = $\frac{V_s^2}{2g}$

Z_o and Z_i = Vertical height at outlet and inlet of the pump from datum line.

If $Z_o = Z_i$ then equation(*ii*) becomes as

$$H_m = \left(30 + \frac{V_d^2}{2g}\right) - \left(6 + \frac{V_s^2}{2g}\right) \quad ...(iii)$$

Now $$V_d = \frac{\text{Discharge}}{\text{Area of delivery pipe}} = \frac{0.04}{\frac{\pi}{4}(D_d)^2} = \frac{.04}{\frac{\pi}{4}\times .1^2} = 5.09 \text{ m/s}$$

And $$V_s = \frac{.04}{\text{Area of suction pipe}} = \frac{.04}{\frac{\pi}{4}D_s^2} = \frac{.04}{\frac{\pi}{4}\times .15^2} = 2.26 \text{ m/s.}$$

Substituting these values in equation (*iii*), we get

$$H_m = \left(30 + \frac{5.09^2}{2\times 9.81}\right) - \left(6 + \frac{2.26^2}{2\times 9.81}\right)$$

$$= (30 + 1.32) - (6 + .26) = 31.32 - 6.26 = 25.06 \text{ m.}$$

Substituting the value of 'H_m' in equation (*i*), we get

$$\eta_o = .02424 \times 25.06 = 0.6074 = \mathbf{60.74\%. \text{ Ans.}}$$

(*iii*) **Manometric efficiency of the pump (η_{man}).**

Tangential velocity at outlet is given by

$$u_2 = \frac{\pi D_2 \times N}{60} = \frac{\pi \times 0.4 \times 1000}{60} = 20.94 \text{ m/s.}$$

From outlet velocity triangle, we have

$$\tan \phi = \frac{V_{f_2}}{u_2 - V_{w_2}} = \frac{2.0}{20.94 - V_{w_2}}$$

$$\therefore \quad 20.94 - V_{w_2} = \frac{2.0}{\tan \phi} = \frac{2.0}{\tan 45} = 2.0$$

$$\therefore \quad V_{w_2} = 20.94 - 2.0 = 18.94.$$

Using equation (19.8), $$\eta_{man} = \frac{gH_m}{V_{w_2} u_2} = \frac{9.81 \times 25.06}{18.94 \times 20.94} = 0.6198 = \mathbf{61.98\%. \text{ Ans.}}$$

Problem 19.9 *Find the power required to drive a centrifugal pump which delivers 0.04 m^3/s of water to a height of 20 m through a 15 cm diameter pipe and 100 m long. The overall efficiency of the pump is 70% and co-efficient of friction 'f' = 0.15 in the formula* $h_f = \dfrac{4fLV^2}{d \times 2g}$.

Solution. Given :

Discharge, $Q = 0.04\ m^3/s$

Height, $H_s = h_s + h_d = 20$ m

Dia. of pipe, $D_s = D_d = 15$ cm $= 0.15$ m

Length, $L_s + L_d = L = 100$ m

Overall efficiency, $\eta_o = 70\% = 0.70$

Co-efficient of friction, $f = .015$

Velocity of water in pipe, $V_s = V_d = V = \dfrac{\text{Discharge}}{\text{Area of pipe}} = \dfrac{0.04}{\frac{\pi}{4}(.15)^2} = 2.26$ m/s.

Frictional head loss in pipe,

$$(h_{f_s} + h_{f_d}) = \frac{4fLV^2}{d \times 2g} = \frac{4 \times .015 \times 100 \times 2.26^2}{.15 \times 2 \times 9.81} = 10.41 \text{ m.}$$

Using equation (19.7), we get manometric head as

$$H_m = (h_s + h_d) + (h_{f_s} + h_{f_d}) + \frac{V_d^2}{2g}$$

$$= 20 + 10.41 + \frac{2.26^2}{2 \times 9.81} \qquad (\because \quad h_s + h_d = H_s = 20 \text{ m})$$

$$= 30.41 + 0.26 = 30.67 \text{ m.}$$

Overall efficiency is given by equation (19.10) as

$$\eta_o = \frac{\left(\frac{WH_m}{1000}\right)}{\text{S.P.}} = \frac{\rho g \times Q \times H_m}{1000 \times \text{S.P.}}$$

$$\therefore \qquad \text{S.P.} = \frac{\rho g \times Q \times H_m}{1000 \times \eta_o} = \frac{1000 \times 9.81 \times .04 \times 30.67}{1000 \times 0.70} = \mathbf{17.19\ kW.\ Ans.}$$

S.P. is the power required to drive the centrifugal pump.

Problem 19.10 *Show that the pressure rise in the impeller of a centrifugal pump when frictional and other losses in the impeller are neglected is given by*

$$\frac{1}{2g}\left[V_{f_1}^2 + u_2^2 - V_{f_2}^2\ cosec^2\ \phi\right]$$

where V_{f_1} *and* V_{f_2} *are velocity of flow at inlet and outlet,*

u_2 *= tangential velocity of impeller at outlet, and* ϕ *= vane angle at outlet.*

Solution. Let suffix 1 represents the values at the inlet and suffix 2 represents the values at the outlet of the impeller.

Applying Bernoulli's equation at the inlet and outlet of the impeller and neglecting losses from inlet to outlet,

Total energy at inlet = Total energy at outlet – Work done by impeller on water

$$\frac{p_1}{\rho g}+\frac{V_1^2}{2g}+Z_1=\left(\frac{p_2}{\rho g}+\frac{V_2^2}{2g}+Z_2\right)-\text{Work done by impeller on water per kg of water}$$

$$=\frac{p_2}{\rho g}+\frac{V_2^2}{2g}+Z_2-\frac{V_{w_2}u_2}{g} \qquad \text{(taking flow radial at inlet)}$$

If inlet and outlet of the impeller are at the same height,

$$\frac{p_1}{\rho g}+\frac{V_1^2}{2g}=\frac{p_2}{\rho g}+\frac{V_2^2}{2g}-\frac{V_{w_2}u_2}{g} \qquad (\because \quad Z_1=Z_2)$$

$$\therefore \qquad \left(\frac{p_2}{\rho g}-\frac{p_1}{\rho g}\right)=\frac{V_1^2}{2g}-\frac{V_2^2}{2g}+\frac{V_{w_2}u_2}{g}$$

But $\frac{p_2}{\rho g}-\frac{p_1}{\rho g}$ = Pressure rise in impeller

$$\therefore \quad \text{Pressure rise in impeller} = \frac{V_1^2}{2g}-\frac{V_2^2}{2g}+\frac{V_{w_2}u_2}{g} \qquad ...(i)$$

From Fig. 19.9, we have

From inlet velocity triangle, $V_1 = V_{f_1}$ ...(*ii*)

From outlet velocity triangle, $\tan\phi=\frac{V_{f_2}}{(u_2-V_{w_2})}$ or $u_2-V_{w_2}=\frac{V_{f_2}}{\tan\phi}$

$$\therefore \qquad V_{w_2}=u_2-\frac{V_{f_2}}{\tan\phi}=u_2-V_{f_2}\cot\phi \qquad ...(iii)$$

Also
$$\begin{aligned}V_2^2 &= V_{f_2}^2+V_{w_2}^2=V_{f_2}^2+(u_2-V_{f_2}\cot\phi)^2\\ &=V_{f_2}^2+(u_2^2+V_{f_2}^2\cot^2\phi-2u_2V_{f_2}\cot\phi)\\ &=V_{f_2}^2+V_{f_2}^2\cot^2\phi+u_2^2-2u_2V_{f_2}\cot\phi\\ &=V_{f_2}^2(1+\cot^2\phi)+u_2^2-2u_2V_{f_2}\cot\phi\\ &=V_{f_2}^2\operatorname{cosec}^2\phi+u_2^2-2u_2V_{f_2}\cot\phi \qquad (\because 1+\cot^2\phi=\operatorname{cosec}^2\phi) \quad ...(iv)\end{aligned}$$

Substituting the values of V_1, V_{w_2} and V_2^2 given by equations (*ii*), (*iii*) and (*iv*) in equation (*i*), we get

$$\begin{aligned}\text{Pressure rise} &= \frac{V_{f_1}^2}{2g}-\frac{1}{2g}(V_{f_2}^2\operatorname{cosec}^2\phi+u_2^2-2u_2V_{f_2}\cot\phi)+\frac{(u_2-V_{f_2}\cot\phi)\times u_2}{g}\\ &=\frac{1}{2g}[V_{f_1}^2-V_{f_2}^2\operatorname{cosec}^2\phi-u_2^2+2u_2V_{f_2}\cot\phi+2u_2^2-2u_2V_{f_2}\times\cot\phi]\\ &=\frac{1}{2g}[V_{f_1}^2-V_{f_2}^2\operatorname{cosec}^2\phi+u_2^2]\\ &=\frac{1}{2g}[V_{f_1}^2+u_2^2-V_{f_2}^2\operatorname{cosec}^2\phi]. \qquad ...(19.12)\end{aligned}$$

Problem 19.11 *Find the rise in pressure in the impeller of a centrifugal pump through which water is flowing at the rate of 0.01 m^3/s. The internal and external diameters of the impeller are 15 cm and 30 cm respectively. The widths of the impeller at inlet and outlet are 1.2 cm and 0.6 cm. The pump is running at 1500 r.p.m. The water enters the impeller radially at inlet and impeller vane angle at outlet is 45°. Neglect losses through the impeller.*

Solution. Given :

Discharge, $Q = .10\ m^3/s$

Internal dia., $D_1 = 15\ cm = 0.15\ m$

External dia., $D_2 = 30\ cm = 0.30\ m$

Width at inlet, $B_1 = 1.2\ cm = 0.012\ m$

Width at outlet, $B_2 = 0.6\ cm = 0.006\ m$

Speed, $N = 1500$ r.p.m.

Vane angle at inlet, $\phi = 45°$

Velocity of flow at outlet, $$V_{f_1} = \frac{Q}{\text{Area of flow at inlet}} = \frac{.01}{\pi D_1 B_1} = \frac{.01}{\pi \times .15 \times .012} = 1.768 \text{ m/s}$$

Velocity of flow at outlet, $$V_{f_2} = \frac{Q}{\pi D_2 B_2} = \frac{.01}{\pi \times .30 \times .006} = 1.768 \text{ m/s.}$$

Tangential velocity of impeller at outlet,

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.30 \times 1500}{60} = 23.56 \text{ m/s.}$$

Using equation (19.12),

$$\text{Pressure rise} = \frac{1}{2g} [V_{f_1}^2 + u_2^2 - V_{f_2}^2 \operatorname{cosec}^2 \phi]$$

$$= \frac{1}{2g} [1.768^2 + 23.56^2 - 1.768^2 \operatorname{cosec}^2 45°]$$

But $$\operatorname{cosec}^2 45° = 1 + \cot^2 45° = 1 + \frac{1}{\tan^2 45°} = 1 + 1 = 2$$

$\therefore$ Pressure rise $$= \frac{1}{2 \times 9.81} [1.768^2 + 23.56^2 - 1.768^2 \times 2.0]$$

$$= \frac{1}{2 \times 9.81} [3.1258 + 555.07 - 6.25] = \mathbf{28.13\ m.\ Ans.}$$

Problem 19.12 *Prove that the manometric head of a centrifugal pump running at speed N and giving a discharge Q may be written as :*

$$H_{mano} = AN^2 + BNQ + CQ^2$$

where A, B and C are constants.

Solution. From equation (19.4), we know that the manometric head is equal to the head imparted by the impeller to the water minus the losses of head in the impeller and casing.

$\therefore$ Manometric head $$= \frac{V_{w_2} u_2}{g} - \text{Losses of head in impeller and casing}$$

or $$H_{mano} = \frac{V_{w_2} u_2}{g} - \frac{K V_2^2}{2g} \qquad ...(i)$$

where V_2 = Absolute velocity of water at outlet of impeller and

$K\dfrac{V_2^2}{2g}$ is the part of head not converted into pressure head and is actually lost in eddies.

Now u_2 = Velocity of impeller at outlet

$$= \frac{\pi D_2 N}{60}$$
$$= K_1 N$$

where $K_1 = \dfrac{\pi D_2}{60}$ and is a constant.

From equation (19.2 *A*), we know that

$$Q = \pi D_2 B_2 \times V_{f_2}$$

$$\therefore \quad V_{f_2} = \frac{Q}{\pi D_2 B_2} = K_2 Q$$

where $K_2 = \dfrac{1}{\pi D_2 B_2}$ and is a constant for a given pump.

From Fig. 19.10, it is clear that

$$\tan \phi = \frac{V_{f_2}}{u_2 - V_{w_2}}$$

$$\therefore \quad u_2 - V_{w_2} = \frac{V_{f_2}}{\tan \phi} = V_{f_2} \cot \phi$$

or
$$V_{w_2} = u_2 - V_{f_2} \cot \phi$$
$$= u_2 - K_2 Q \cot \phi \qquad (\because\ V_{f_2} = K_2 Q)$$
$$= u_2 - K_3 Q \qquad \text{where } K_3 = K_2 \cot \phi$$
$$= K_1 N - K_3 Q \qquad (\because\ u_2 = K_1 N)$$

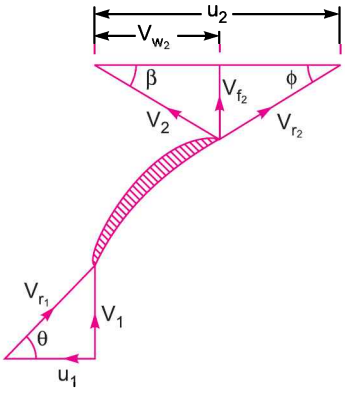


Fig. 19.10

Now from outlet velocity triangle, we know that

$$V_2^2 = V_{f_2}^2 + V_{w_2}^2$$
$$= (K_2 Q)^2 + (K_1 N - K_3 Q)^2 \qquad (\because\ V_{f_2} = K_2 Q \text{ and } V_{w_2} = K_1 N - K_3 Q)$$

Substituting the values of V_{w_2}, u_2 and V_2 in equation (*i*), we get

$$H_{mano} = \frac{(K_1 N - K_3 Q)(K_1 N)}{g} - \frac{K\left[K_2^2 Q^2 + K_1^2 N^2 + K_3^2 Q^2 - 2K_1 N \times K_3 Q\right]}{2g}$$

$$= \frac{1}{2g}\left[2(K_1^2 N^2 - K_1 K_3 NQ) - KK_2^2 Q^2 - KK_1^2 N^2 - KK_3^2\, Q^2 + 2KK_1 K_3 NQ\right]$$

$$= \frac{1}{2g}\left[N^2(2K_1^2 - KK_1^2) + NQ(2KK_1 K_3 - 2K_1 K_3) + Q^2(-KK_2^2 - KK_3^2)\right]$$

$$= AN^2 + BNQ + CQ^2$$

where $A = \dfrac{2K_1^2 - KK_1^2}{2g}$, $B = \dfrac{2KK_1 K_3 - 2K_1 K_3}{2g}$ and $C = \dfrac{-KK_2^2 - KK_3^2}{2g}$ and they are constant.

▶ 19.5 MINIMUM SPEED FOR STARTING A CENTRIFUGAL PUMP

If the pressure rise in the impeller is more than or equal to manometric head (H_m), the centrifugal pump will start delivering water. Otherwise, the pump will not discharge any water, though the impeller is rotating. When impeller is rotating, the water in contact with the impeller is also rotating. This is the case of forced vortex. In case of forced vortex, the centrifugal head or head due to pressure rise in the impeller

$$= \frac{\omega^2 r_2^2}{2g} - \frac{\omega^2 r_1^2}{2g} \qquad ...(i)$$

where ωr_2 = Tangential velocity of impeller at outlet = u_2, and
ωr_1 = Tangential velocity of impeller at inlet = u_1.

$\therefore$ Head due to pressure rise in impeller $= \frac{u_2^2}{2g} - \frac{u_1^2}{2g}$

The flow of water will commence only if

Head due to pressure rise in impeller $\geq H_m$ or $\frac{u_2^2}{2g} - \frac{u_1^2}{2g} \geq H_m$.

For minimum speed, we must have $\frac{u_2^2}{2g} - \frac{u_1^2}{2g} = H_m$ ...(19.13)

But from equation (19.8), we have $\eta_{man} = \frac{gH_m}{V_{w_2} u_2}$

$\therefore$ $H_m = \eta_{man} \times \frac{V_{w_2} u_2}{g}$.

Substituting this value of H_m in equation (19.13),

$$\frac{u_2^2}{2g} - \frac{u_1^2}{2g} = \eta_{man} \times \frac{V_{w_2} u_2}{g} \qquad ...(19.14)$$

Now $u_2 = \frac{\pi D_2 N}{60}$ and $u_1 = \frac{\pi D_1 N}{60}$.

Substituting the values of u_2 and u_1 in equation (19.14),

$$\frac{1}{2g}\left(\frac{\pi D_2 N}{60}\right)^2 - \frac{1}{2g}\left(\frac{\pi D_1 N}{60}\right)^2 = \eta_{man} \times \frac{V_{w_2} \times \pi D_2 N}{g \times 60}$$

Dividing by $\frac{\pi N}{g \times 60}$, we get $\frac{\pi N D_2^2}{120} - \frac{\pi N D_1^2}{120} = \eta_{man} \times V_{w_2} \times D_2$

or $\frac{\pi N}{120} [D_2^2 - D_1^2] = \eta_{man} \times V_{w_2} \times D_2$

$\therefore$ $N = \frac{120 \times \eta_{man} \times V_{w_2} \times D_2}{\pi \left[D_2^2 - D_1^2\right]}$ ...(19.15)

Equation (19.15) gives the minimum starting speed of the centrifugal pump.

Problem 19.13 *The diameters of an impeller of a centrifugal pump at inlet and outlet are 30 cm and 60 cm respectively. Determine the minimum starting speed of the pump if it works against a head of 30 m.*

Solution. Given :

Dia. of impeller at inlet, $D_1 = 30$ cm $= 0.30$ m

Dia. of impeller at outlet, $D_2 = 60$ cm $= 0.60$ m

Head, $H_m = 30$ cm

Let the minimum starting speed $= N$

Using equation (19.l3) for minimum speed,

$$\frac{u_2^2}{2g} - \frac{u_1^2}{2g} = H_m$$

where $u_2 = \frac{\pi \times D_2 \times N}{60} = \frac{\pi \times 0.6 \times N}{60} = 0.03141\ N$

$$u_1 = \frac{\pi \times D_1 \times N}{60} = \frac{\pi \times .3 \times N}{60} = 0.0157\ N$$

$\therefore \quad \frac{1}{2g}(0.3141\ N)^2 - \frac{1}{2g}(.0157\ N)^2 = 30$

or $(.03141\ N)^2 - (.0157\ N)^2 = 30 \times 2 \times g = 30 \times 2 \times 9.81$

or $$N^2 = \frac{30 \times 2 \times 9.81}{\left(.03141^2 - .0157^2\right)} = \frac{588.6}{.0009866 - .0002465} = 795297.9$$

$\therefore \quad N = \sqrt{795297.9} =$ **891.8 r.p.m. Ans.**

Problem 19.14 *The diameters of an impeller of a centrifugal pump at inlet and outlet are 30 cm and 60 cm respectively. The velocity of flow at outlet is 2.0 m/s and the vanes are set back at an angle of 45° at the outlet. Determine the minimum starting speed of the pump if the manometric efficiency is 70%.*

Solution. Given :

Diameter at inlet, $D_1 = 30$ cm $= 0.30$ m

Diameter at outlet, $D_2 = 60$ cm $= 0.60$ m

Velocity of flow at outlet, $V_{f_2} = 2.0$ m/s

Vane angle at outlet, $\phi = 45°$

Manometric efficiency, $\eta_{man} = 70\% = 0.70$.

Let the minimum starting speed $= N$.

From Fig. 19.9, for velocity triangle at outlet, we have

$$\tan\phi = \frac{V_{f_2}}{\left(u_2 - V_{w_2}\right)} \quad \text{or} \quad u_2 - V_{w_2} = \frac{V_{f_2}}{\tan\phi} = \frac{2.0}{\tan 45°} = 2.0$$

$\therefore \quad V_{w_2} = u_2 - 2.0$

But $$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.60 \times N}{60} = 0.03141\ \text{N}$$

$\therefore \quad V_{w_2} = (0.03141N - 2.0).$

Using equation (19.15) for minimum starting speed,

$$N = \frac{120 \times \eta_{man} \times V_{w_2} \times D_2}{\pi\left[D_2^2 - D_1^2\right]} = \frac{120 \times 0.70 \times (.03141\,N - 2.0) \times 0.6}{\pi\left[.6^2 - .3^2\right]}$$

$$= \frac{50.4(.03141\,N - 2.0)}{\pi[.36 - .09]} = 59.417\ [.03141\ N - 2.0]$$

$$= 1.866\ N - 118.834$$

or $\quad 1.866\,N - N = 118.834$ or $.886\,N = 118.834$

$\therefore \quad N = \dfrac{118.834}{0.866} =$ **137.22 r.p.m. Ans.**

Problem 19.15 *A centrifugal pump with 1.2 m diameter runs at 200 r.p.m. and pumps 1880 litres/s, the average lift being 6 m. The angle which the vanes make at exit with the tangent to the impeller is 26° and the radial velocity of flow is 2.5 m/s. Determine the manometric efficiency and the least speed to start pumping against a head of 6 m, the inner diameter of the impeller being 0.6 m.*

Solution. Given :

Dia. at outlet, $D_2 = 1.2$ m

Speed, $N = 200$ r.p.m.

Discharge, $Q = 1880$ litres/s $= 1.88$ m^3/s

Manometric head, $H_m = .6$ m

Angle of vane at outlet, $\phi = 26°$

Velocity of flow at outlet, $V_{f_2} = 2.5$ m/s

Dia. at inlet, $D_1 = 0.6$ m

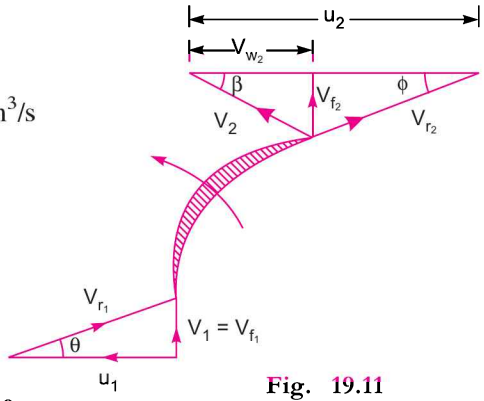


Fig. 19.11

(*i*) *Manometric efficiency* (η_{man})

Using equation (19.8), $\quad \eta_{man} = \dfrac{gH_m}{V_{w_2} \times u_2}$ $\quad ...(i)$

But $\quad u_2 = \dfrac{\pi D_2 N}{60} = \dfrac{\pi \times 1.2 \times 200}{60} = 12.56$ m/s

and $\quad \tan\phi = \dfrac{V_{f_2}}{u_2 - V_{w_2}}$ or $u_2 - V_{w_2} = \dfrac{V_{f_2}}{\tan\phi} = \dfrac{2.5}{\tan 26°} = 5.13$

$\therefore \quad V_{w_2} = u_2 - 5.13 = 12.56 - 5.13 = 7.43$ m/s.

Substituting these values in equation (*i*), we get

$$\eta_{man} = \frac{9.81 \times 6.0}{7.43 \times 12.56} = 0.63 = \mathbf{63\%}.\ \textbf{Ans.}$$

(*ii*) *Least speed to start the pump* :

Least speed to start the pump is given by equation(19.13),

$$\frac{u_2^2}{2g} - \frac{u_1^2}{2g} = H_m \qquad ...(ii)$$

where u_2 and u_1 are the tangential velocities of the vane at outlet and inlet respectively, corresponding to least speed of the pump.

But $u_2 = \omega \times r_2$ and $u_1 = \omega \times r_1$.

Substituting these values in equation (*ii*), we get

$$\frac{(\omega \times r_2)^2}{2g} - \frac{(\omega \times r_1)^2}{2g} = H_m = 6.0 \text{ or } \frac{\omega^2}{2g}\left[r_2^2 - r_1^2\right] = 6.0$$

or $$\frac{\omega^2}{2 \times 9.81}\left[0.6^2 - 0.3^2\right] = 6.0 \quad \left(\because \quad r_2 = \frac{D_2}{2} = \frac{1.2}{2} = 0.6 \text{ m and } r_1 = \frac{D_1}{2} = \frac{0.6}{2} = 0.3 \text{ m}\right)$$

$$\therefore \quad \omega^2 = \frac{6.0 \times 2.0 \times 9.81}{0.36 - .09} = 436 \quad \therefore \quad \omega = \sqrt{436} = 20.88 = \frac{2\pi N}{60}$$

$$\therefore \quad N = \frac{60 \times 20.88}{2 \times \pi} = \textbf{200 r.p.m. Ans.}$$

▶ 19.6 MULTISTAGE CENTRIFUGAL PUMPS

If a centrifugal pump consists of two or more impellers, the pump is called a multistage centrifugal pump. The impellers may be mounted on the same shaft or on different shafts. A multistage pump is having the following two important functions :

1. To produce a high head, and 2. To discharge a large quantity of liquid.

If a high head is to be developed, the impellers are connected in series (or on the same shaft) while for discharging large quantity of liquid, the impellers (or pumps) are connected in parallel.

19.6.1 Multistage Centrifugal Pumps for High Heads. For developing a high head, a number of impellers are mounted in series or on the same shaft as shown in Fig. 19.12.

The water from suction pipe enters the 1st impeller at inlet and is discharged at outlet with increased pressure. The water with increased pressure from the outlet of the 1st impeller is taken to the inlet of the 2nd impeller with the help of a connecting pipe as shown in Fig. 19.12. At the outlet of the 2nd impeller, the pressure of water will be more than the pressure of water at the outlet of the 1st impeller. Thus if more impellers are mounted on the same shaft, the pressure at the outlet will be increased further.

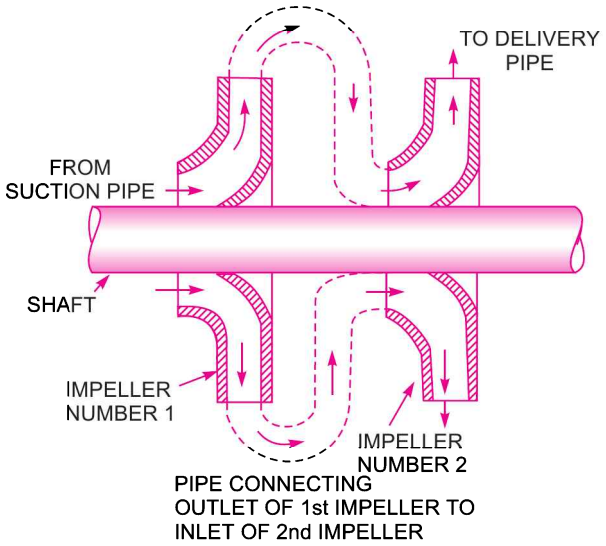


Fig. 19.12 *Two-stage pumps with impellers in series.*

Let n = Number of identical impellers mounted on the same shaft,

H_m = Head developed by each impeller.

Then total head developed

$$= n \times H_m \qquad ...(19.16)$$

The discharge passing through each impeller is same

19.6.2 Multistage Centrifugal Pumps for High Discharge. For obtaining high discharge, the pumps should be connected in parallel as shown in Fig. 19.13. Each of the pumps lifts the water from a common pump and discharges water to a common pipe to which the delivery pipes of each pump is connected. Each of the pump is working against the same head.

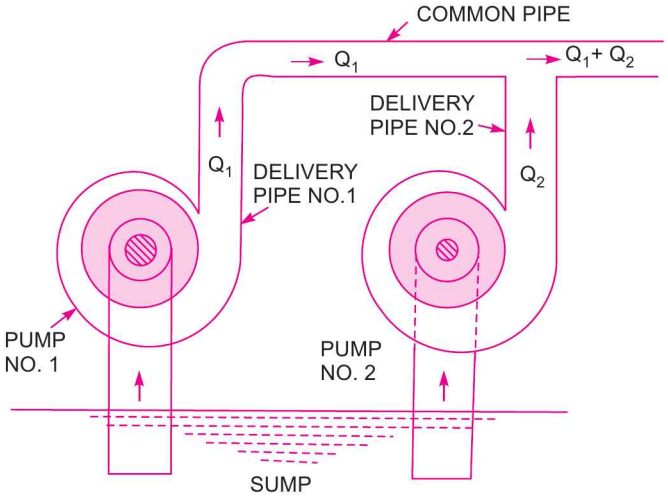


Fig. 19.13 *Pumps in parallel.*

Let n = Number of identical pumps arranged in parallel.

Q = Discharge from one pump.

$\therefore$ Total discharge $= n \times Q \qquad ...(19.17)$

Problem 19.16 *A three stage centrifugal pump has impellers 40 cm in diameter and 2 cm wide at outlet. The vanes are curved back at the outlet at 45° and reduce the circumferential area by 10%. The manometric efficiency is 90% and the overall efficiency is 80%. Determine the head generated by the pump when running at 1000 r.p.m. delivering 50 litres per second. What should be the shaft horse power ?*

Solution. Given :

Number of stages, $n = 3$

Dia. of impeller at outlet, $D_2 = 40$ cm $= 0.40$ m

Width at outlet, $B_2 = 2$ cm $= 0.02$ m

Vane angle at outlet, $\phi = 45°$

Reduction in area at outlet $= 10\% = 0.1$

$\therefore$ Area of flow at outlet $= 0.9 \times \pi D_2 \times B_2 = 0.9 \times \pi \times .4 \times .02 = 0.02262$ m^2

Manometric efficiency, $\eta_{man} = 90\% = 0.90$

Overall efficiency, $\eta_o = 80\% = 0.80$

Speed, $N = 1000$ r.p.m.

Discharge, $Q = 50$ litres/s $= 0.05$ m^3/s

Determine : (*i*) Head generated by the pump and

(*ii*) Shaft power.

Velocity of flow at outlet, $V_{f_2} = \dfrac{\text{Discharge}}{\text{Area of flow}} = \dfrac{0.05}{.02262} = 2.21$ m/s

Tangential velocity of impeller at outlet,

$$u_2 = \frac{\pi D_2 N}{60} = \frac{\pi \times 0.4 \times 1000}{60} = 20.94 \text{ m/s}$$

Refer to Fig. 19.9. From velocity triangle at outlet,

$$\tan \phi = \frac{V_{f_2}}{u_2 - V_{w_2}}$$

$$\therefore \quad u_2 - V_{w_2} = \frac{V_{f_2}}{\tan \phi} = \frac{2.21}{\tan 45^\circ} = 2.21 \text{ m/s}$$

$$\therefore \quad V_{w_2} = u_2 - 2.21 = 20.94 - 2.21 = 18.73 \text{ m/s}$$

Using equation (19.8), $\eta_{man} = \dfrac{gH_m}{V_{w_2} u_2}$, $0.90 = \dfrac{9.81 \times H_m}{18.73 \times 20.94}$

$$\therefore \quad H_m = \frac{0.90 \times 18.73 \times 20.94}{9.81} = 35.98 \text{ m.}$$

Using equation (19.16) for total head generated by pump,

$$= n \times H_m = 3 \times 35.98 = \textbf{107.94 m. Ans.}$$

$$\therefore \text{ Power output of the pump } = \frac{\text{Weight of water lifted} \times \text{Total head}}{1000}$$

$$= \frac{\rho g \times Q \times 107.94}{1000} = \frac{1000 \times 9.81 \times 0.05 \times 107.94}{1000} = 52.94 \text{ kW.}$$

Using equation (19.10), we have $\eta_o = \dfrac{\text{Power output of pump}}{\text{Power input to the pump}} = \dfrac{52.94}{\text{S.P.}}$

$$\therefore \text{ Shaft power } = \frac{52.94}{\eta_o} = \frac{52.94}{0.80} = \textbf{66.175 kW. Ans.}$$

Problem 19.17 *A four-stage centrifugal pump has four identical impellers, keyed to the same shaft. The shaft is running at 400 r.p.m. and the total manometric head developed by the multistage pump is 40 m. The discharge through the pump is 0.2 m^3/s. The vanes of each impeller are having outlet angle as 45°. If the width and diameter of each impeller at outlet is 5 cm and 60 cm respectively, find the manometric efficiency.*

Solution. Given :

Number of stage, $n = 4$

Speed, $N = 400$ r.p.m.

Total manometric head = 40 m

$\therefore$ Manometric head for each stage, $H_m = \dfrac{40}{4} = 10.0$ m

Discharge, $Q = 0.2$ m^3/s

Outlet vane angle, $\phi = 45°$

Width at outlet, $B_2 = 5$ cm $= 0.05$ m

Dia. at outlet, $D_2 = 60$ cm $= 0.6$ m

Tangential velocity of impeller at outlet, $u_2 = \dfrac{\pi D_2 N}{60} = \dfrac{\pi \times 0.6 \times 400}{60} = 12.56$ m/s

Velocity of flow at outlet, $V_{f_2} = \dfrac{\text{Discharge}}{\text{Area of flow}} = \dfrac{0.20}{\pi D_2 B_2} = \dfrac{0.20}{\pi \times 0.6 \times 0.5} = 2.122$ m/s

Refer to Fig. 19.9. From velocity triangle at outlet,

$$\tan \phi = \frac{V_{f_2}}{u_2 - V_{w_2}}$$

$$u_2 - V_{w_2} = \frac{V_{f_2}}{\tan \phi} = \frac{2.122}{\tan 45°} = 2.122 \text{ m/s}$$

$\therefore$ $V_{w_2} = u_2 - 2.122 = 12.56 - 2.122 = 10.438$

Using equation (19.8), $\eta_{man} = \dfrac{gH_m}{V_{w_2} u_2} = \dfrac{9.81 \times 10.0}{10.438 \times 12.56} =$ **0.7482** or **74.82%. Ans.**

▶ 19.7 SPECIFIC SPEED OF A CENTRIFUGAL PUMP (N_s)

The specific speed of a centrifugal pump is defined as the speed of a geometrically similar pump which would deliver *one cubic metre* of liquid per second against a head of *one metre*. It is denoted by 'N_s'.

19.7.1 Expression for Specific Speed for a Pump. The discharge, Q, for a centrifugal pump is given by the relation

$$Q = \text{Area} \times \text{Velocity of flow}$$
$$= \pi D \times B \times V_f \text{ or } Q \propto D \times B \times V_f \quad ...(i)$$

where D = Diameter of the impeller of the pump and

B = Width of the impeller.

We know that $B \propto D$

$\therefore$ From equation *(i)*, we have $Q \propto D^2 \times V_f$ ...(*ii*)

We also know that tangential velocity is given by

$$u = \frac{\pi DN}{60} \propto DN \quad ...(iii)$$

Now the tangential velocity (u) and velocity of flow (V_f) are related to the manometric head (H_m) as

$$u \propto V_f \propto \sqrt{H_m} \quad ...(iv)$$

Substituting the value of u in equation (*iii*), we get

$$\sqrt{H_m} \propto DN \text{ or } D \propto \frac{\sqrt{H_m}}{N}$$

Substituting the values of D in equation (*ii*),

$$Q \propto \frac{H_m}{N^2} \times V_f$$

$$\propto \frac{H_m}{N^2} \times \sqrt{H_m} \qquad [\because \text{ From equation } (iv),\ V_f \propto \sqrt{H_m}\ ]$$

$$\propto \frac{H_m^{3/2}}{N^2}$$

$$\therefore \qquad Q = \text{K}\ \frac{H_m^{3/2}}{N^2} \qquad \ldots(v)$$

where K is a constant of proportionality.

If $H_m = 1$ m and $Q = 1\ \text{m}^3/\text{s}$, N becomes $= N_s$.

Substituting these values in equation (*v*), we get

$$1 = K\ \frac{1^{3/2}}{N_s^2} = \frac{K}{N_s^2}$$

$$\therefore \qquad K = N_s^{\,2}$$

Substituting the value of K in equation (*v*), we get

$$Q = N_s^{\,2} \frac{H_m^{3/2}}{N^2} \quad \text{or} \quad N_s^{\,2} = \frac{N^2 Q}{H_m^{3/2}}$$

$$\therefore \qquad N_s = \frac{N\sqrt{Q}}{H_m^{3/4}} \qquad \ldots(19.18)$$

▶ 19.8 MODEL TESTING OF CENTRIFUGAL PUMPS

Before manufacturing the large sized pumps, their models which are in complete similarity with the actual pumps (also called prototypes) are made. Tests are conducted on the models and performance of the prototypes are predicted. The complete similarity between the model and actual pump (prototype) will exist if the following conditions are satisfied :

1. Specific speed of model = Specific speed of prototype

$$(N_s)_m = (N_s)_p \quad \text{or} \quad \left(\frac{N\sqrt{Q}}{H_m^{3/4}}\right)_m = \left(\frac{N\sqrt{Q}}{H_m^{3/4}}\right)_p \qquad \ldots(19.19)$$

2. Tangential velocity (u) is given by $u = \dfrac{\pi DN}{60}$ also $u \propto \sqrt{H_m}$

$$\therefore \qquad \sqrt{H_m} \propto DN$$

$$\therefore \qquad \frac{\sqrt{H_m}}{DN} = \text{Constant} \qquad \ldots(19.19\ A)$$

or $$\left(\frac{\sqrt{H_m}}{DN}\right)_m = \left(\frac{\sqrt{H_m}}{DN}\right)_p \quad ...(19.20)$$

3. From equation (*ii*) of Art. 19.7.1, we have

$$Q \propto D^2 \times V_f \qquad \text{where } V_f \propto u \propto DN$$
$$\propto D^2 \times D \times N$$
$$\propto D^3 \times N$$

$$\therefore \quad \frac{Q}{D^3N} = \text{Constant or } \left(\frac{Q}{D^3N}\right)_m = \left(\frac{Q}{D^3N}\right)_p \quad ...(19.21)$$

4. Power of the pump, $$P = \frac{\rho \times g \times Q \times H_m}{75}$$

$$\therefore \quad P \propto Q \times H_m$$
$$\propto D^3 \times N \times H_m \qquad (\because \quad Q \propto D^3N)$$
$$\propto D^3N \times D^2N^2 \qquad (\because \quad \sqrt{H_m} \propto DN)$$
$$\propto D^5N^3$$

$$\therefore \quad \frac{P}{D^5N^3} = \text{Constant} \quad \text{or} \quad \left(\frac{P}{D^5N^3}\right)_m = \left(\frac{P}{D^5N^3}\right)_p \quad ...(19.22)$$

Problem 19.18 *A single-stage centrifugal pump with impeller diameter of 30 cm rotates at 2000 r.p.m. and lifts 3 m^3 of water per second to a height of 30 m with an efficiency of 75%. Find the number of stages and diameter of each impeller of a similar multistage pump to lift 5 m^3 of water per second to a height of 200 metres when rotating at 1500 r.p.m.*

Solution. Given :

Single-stage pump :

Diameter of impeller, $D_1 = 30 \text{ cm} = 0.30 \text{ m}$

Speed, $N_1 = 2000$ r.p.m.

Discharge, $Q_1 = 3 \text{ m}^3/\text{s}$

Height, $H_{m_1} = 30$ m

Efficiency, $\eta_{man} = 75\% = 0.75$.

Multistage similar pump :

Discharge, $Q_2 = 5 \text{ m}^3/\text{s}$

Total height $= 200$ m

Let the height per stage $= H_{m_2}$

Speed, $N_2 = 1500$ r.p.m.

Diameter of each impeller $= D_2$

Specific speed should be same. Hence, applying equation (19.19) as

$$\left(\frac{N\sqrt{Q}}{H_m^{3/4}}\right)_1 = \left(\frac{N\sqrt{Q}}{H_m^{3/4}}\right)_2$$

$$\therefore \quad \frac{N_1\sqrt{Q_1}}{H_{m_1}^{3/4}} = \frac{N_2\sqrt{Q_2}}{H_{m_2}^{3/4}} \quad \text{or} \quad \frac{2000\times\sqrt{3}}{30^{3/4}} = \frac{1500\times\sqrt{5}}{H_{m_2}^{3/4}}$$

$$\therefore \quad H_{m_2}^{3/4} = \frac{1500\times\sqrt{5}\times 30^{3/4}}{2000\times\sqrt{3}} = \frac{1500}{2000}\times\sqrt{\frac{5}{3}}\times 12.818 = 12.411$$

$$\therefore \quad H_{m_2} = (12.411)^{4/3} = 28.71 \text{ m}$$

$$\therefore \quad \text{Number of stages} = \frac{\text{Total head}}{\text{Head per stage}} = \frac{200}{28.71} = 6.96 \simeq \textbf{7. Ans.}$$

Using equation (19.20), we have

$$\frac{\sqrt{H_{m_1}}}{D_1N_1} = \frac{\sqrt{H_{m_2}}}{D_2N_2} \quad \text{or} \quad \frac{\sqrt{30}}{0.30\times 2000} = \frac{\sqrt{28.71}}{D_2\times 1500}$$

$$\therefore \quad D_2 = \frac{0.30\times 2000\times\sqrt{28.71}}{1500\times\sqrt{30}} = 0.3913 \text{ m} = \textbf{391.3 mm. Ans.}$$

Problem 19.19 *Find the number of pumps required to take water from a deep well under a total head of 89 m. All the pumps are identical and are running at 800 r.p.m. The specific speed of each pump is given as 25 while the rated capacity of each pump is 0.16 m^3/s.*

Solution. Given :

Total head = 89 m
Speed, N = 800 r.p.m.
Specific speed, N_s = 25
Rate capacity, Q = 0.16 m^3/s
Let H_m = Head developed by each pump.

Using equation (19.18), $N_s = \dfrac{N\sqrt{Q}}{H_m^{3/4}}$

$$25 = \frac{800\times\sqrt{0.16}}{H_m^{3/4}}$$

$$\therefore \quad H_m^{3/4} = \frac{800\times\sqrt{0.16}}{25} = 12.8$$

$$\therefore \quad H_m = (12.8)^{4/3} = 29.94 \text{ m}$$

$$\therefore \quad \text{Number of pumps required} = \frac{\text{Total head}}{\text{Head developed by one pump}} = \frac{89}{29.94} \simeq \textbf{3. Ans.}$$

As the total head is more than the head developed by one pump, the pumps should be connected in series.

Problem 19.20 *Two geometrically similar pumps are running at the same speed of 1000 r.p.m. One pump has an impeller diameter of 0.30 metre and lifts water at the rate of 20 litres per second against a head of 15 metres. Determine the head and impeller diameter of the other pump to deliver half the discharge.*

Solution. Given :

For pump No. 1,

Speed, $N_1 = 1000$ r.p.m.

Diameter, $D_1 = 0.30$ m

Discharge, $Q_2 = 20$ litres/s $= 0.02$ m^3/s

Head, $H_{m_1} = 15$ m

For pump No. 2,

Speed, $N_2 = 1000$ r.p.m.

Discharge, $Q_2 = \frac{Q_1}{2} = \frac{20}{2} = 10$ litres/s $= 0.01$ m^3/s.

Let D_2 = Diameter of impeller

H_{m_2} = Head developed.

Using equation (19.19), $\frac{N_1\sqrt{Q_1}}{H_{m_1}^{3/4}} = \frac{N_2\sqrt{Q_2}}{H_{m_2}^{3/4}}$

$\therefore \quad \frac{1000 \times \sqrt{.02}}{15^{3/4}} = \frac{1000 \times \sqrt{.01}}{H_{m_2}^{3/4}}$

or $H_{m_2}^{3/4} = \frac{1000 \times \sqrt{.01} \times 15^{3/4}}{1000 \times \sqrt{.02}} = \sqrt{\frac{.01}{.02}} \times 7.622 = 5.389$

$\therefore \quad H_{m_2} = (5.389)^{4/3} =$ **9.44 m. Ans.**

Using equation (19.20), $\left(\frac{\sqrt{H_m}}{DN}\right)_1 = \left(\frac{\sqrt{H_m}}{DN}\right)_2$ or $\frac{\sqrt{H_{m_1}}}{D_1N_1} = \frac{\sqrt{H_{m_2}}}{D_2N_2}$

$$\frac{\sqrt{15}}{0.3 \times 1000} = \frac{\sqrt{9.44}}{D_2 \times 1000}$$

$\therefore \quad D_2 = \frac{\sqrt{9.44} \times 0.3}{\sqrt{15}} = 0.238$ m $=$ **238.0 mm. Ans.**

Problem 19.21 *The diameter of a centrifugal pump, which is discharging 0.03 m^3/s of water against a total head of 20 m is 0.40 m. The pump is running at 1500 r.p.m. Find the head, discharge and ratio of powers of a geometrically similar pump of diameter 0.25 m when it is running at 3000 r.p.m.*

Solution. Given :

Centrifugal pump,

Discharge, $Q_1 = .03$ m^3/s

Head, $H_{m_1} = 20$ m

Diameter, $D_1 = 0.40$ m

Speed, $N_1 = 1500$ r.p.m.

Geometrically similar pump,

Diameter, $D_2 = 0.25$ m

Speed, $N_2 = 3000$ r.p.m.

Let Head on similar group $= H_{m_2}$

Discharge on similar pump $= Q_2$

Using equation (19.21), $\left(\frac{Q}{D^3N}\right)_1 = \left(\frac{Q}{D^3N}\right)_2$

$$\therefore \quad \frac{Q_1}{D_1^3 N_1} = \frac{Q_2}{D_2^3 N_2}$$

$$\frac{.03}{.40^3 \times 1500} = \frac{Q_2}{0.25^3 \times 3000}$$

$$\therefore \quad Q_2 = \frac{.03 \times .25^3 \times 3000}{.40^3 \times 1500} = .03 \times \left(\frac{.25}{.40}\right)^3 \times 2.0 = \mathbf{0.01465\ m^3/s.\ Ans.}$$

Using equation (19.20), we have

$$\left(\frac{\sqrt{H_m}}{DN}\right)_1 = \left(\frac{\sqrt{H_m}}{DN}\right)_2$$

or $$\frac{\sqrt{H_{m_1}}}{D_1 N_1} = \frac{\sqrt{H_{m_2}}}{D_2 N_2} \quad \therefore \quad \frac{\sqrt{20}}{0.40 \times 1500} = \frac{\sqrt{H_{m_2}}}{0.25 \times 3000}$$

or $$\sqrt{H_{m_2}} = \frac{\sqrt{20} \times 0.25 \times 3000}{0.40 \times 1500} = 5.59$$

$$\therefore \quad H_{m_2} = (5.59)^2 = \mathbf{31.25\ m.\ Ans.}$$

Using equation (19.22), we have

$$\left(\frac{P}{D^5N^3}\right)_1 = \left(\frac{P}{D^5N^3}\right)_2$$

$$\therefore \quad \frac{P_1}{D_1^5 N_1^3} = \frac{P_2}{D_2^5 N_2^3} \quad \text{or} \quad \frac{P_1}{P_2} = \frac{D_1^5 N_1^3}{D_2^5 N_2^3} = \left(\frac{D_1}{D_2}\right)^5 \times \left(\frac{N_1}{N_2}\right)^3$$

$$= \left(\frac{0.40}{0.25}\right)^5 \times \left(\frac{1500}{3000}\right)^3 = 10.485 \times .125 = \mathbf{1.31.\ Ans.}$$

Problem 19.22 *A one-fifth scale model of a pump was tested in a laboratory at 1000 r.p.m. The head developed and the power input at the best efficiency point were found to be 8 m and 30 kW respectively. If the prototype pump has to work against a head of 25 m, determine its working speed, the power required to drive it and the ratio of the flow rates handled by the two pumps.*

Solution. Given :

One-fifth scale model means that the ratio of linear dimensions of a model and its prototype is equal to 1/5.

Speed of model, $N_m = 1000$ r.p.m.

Head of model, $H_m = 8$ m

Power of model, $P_m = 30$ kW

Head of prototype, $H_p = 25$ m

Let N_p = Speed of prototype

P_p = Power of prototype

Q_p = Flow rate of prototype

Q_m = Flow rate of model

(*i*) *Speed of prototype*

Using equation (19.20), we get

$$\left(\frac{\sqrt{H}}{DN}\right)_m = \left(\frac{\sqrt{H}}{DN}\right)_p \quad \text{or} \quad \frac{\sqrt{H_m}}{D_m N_m} = \frac{\sqrt{H_p}}{D_p N_p}$$

or
$$N_p = \frac{\sqrt{H_p}}{\sqrt{H_m}} \times \frac{D_m}{D_p} \times N_m$$

$$= \frac{\sqrt{25}}{\sqrt{8}} \times \frac{1}{5} \times 1000 \qquad \left(\because \frac{D_m}{D_p} = \frac{1}{5}\right)$$

$$= \mathbf{353.5 \text{ r.p.m. Ans.}}$$

(*ii*) *Power developed by prototype*

Using equation (19.22), we get

$$\left(\frac{P}{D^5 N^3}\right)_m = \left(\frac{P}{D^5 N^3}\right)_p \quad \text{or} \quad \frac{P_m}{D_m^5 N_m^3} = \frac{P_p}{D_p^5 N_p^3}$$

or
$$P_p = P_m \times \left(\frac{D_p}{D_m}\right)^5 \times \left(\frac{N_p}{N_m}\right)^3 = 30 \times 5^5 \times \left(\frac{353.5}{1000}\right)^3 \qquad \left(\because \frac{D_p}{D_m} = \frac{5}{1}\right)$$

$$= 30 \times 3125 \times 0.04419 = \mathbf{4143 \text{ kW. Ans.}}$$

(*iii*) *Ratio of the flow rates of two pumps (i.e., model and prototype)*

$$\left(\frac{Q}{D^3 N}\right)_m = \left(\frac{Q}{D^3 N}\right)_p \quad \text{or} \quad \frac{Q_m}{D_m^3 N_m} = \frac{Q_p}{D_p^3 N_p}$$

or
$$\frac{Q_p}{Q_m} = \frac{D_p^3 N_p}{D_m^3 N_m} = \left(\frac{D_p}{D_m}\right)^3 \times \frac{N_p}{N_m} = 5^3 \times \frac{353.5}{1000} \qquad \left(\because \frac{D_p}{D_m} = \frac{5}{1}\right)$$

$$= \mathbf{44.1875. \text{ Ans.}}$$

▶ 19.9 PRIMING OF A CENTRIFUGAL PUMP

Priming of a centrifugal pump is defined as the operation in which the suction pipe, casing of the pump and a portion of the delivery pipe upto the delivery valve is completely filled up from outside source with the liquid to be raised by the pump before starting the pump. Thus the air from these parts of the pump is removed and these parts are filled with the liquid to be pumped.

The work done by the impeller per unit weight of liquid per sec is known as the head generated by the pump. Equation (19.1) gives the head generated by the pump as = $\frac{1}{g} V_{w_2} u_2$ metre. This equation is independent of the density of the liquid. This means that when pump is running in air, the head generated is in terms of metre of air. If the pump is primed with water, the head generated is same metre of water. But as the density of air is very low, the generated head of air in terms of equivalent metre of water head is negligible and hence the water may not be sucked from the pump. To avoid this difficulty, priming is necessary.

▶ 19.10 CHARACTERISTIC CURVES OF CENTRIFUGAL PUMPS

Characteristic curves of centrifugal pumps are defined as those curves which are plotted from the results of a number of tests on the centrifugal pump. These curves are necessary to predict the behaviour and performance of the pump when the pump is working under different flow rate, head and speed. The following are the important characteristic curves for pumps :

1. Main characteristic curves,
2. Operating characteristic curves, and
3. Constant efficiency or Muschel curves.

19.10.1 Main Characteristic Curves. The main characteristic curves of a centrifugal pump consists of variation of head (manometric head, H_m), power and discharge with respect to speed. For plotting curves of manometric head *versus* speed, discharge is kept constant. For plotting curves of discharge *versus* speed, manometric head (H_m) is kept constant. And for plotting curves of power *versus* speed, the manometric head and discharge are kept constant. Fig. 19.14 shows main characteristic curves of a pump.

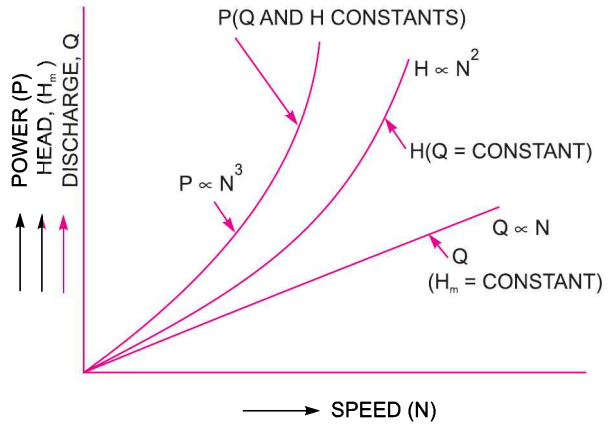


Fig. 19.14 *Main characteristic curves of a pump.*

For plotting the graph of H_m versus speed (N), the discharge is kept constant. From equation (19.19 *A*), it is clear that $\sqrt{H_m}/DN$ is a constant or $H_m \propto N^2$. This means that head developed by a pump is proportional to N^2. Hence the curve of H_m v/s N is a parabolic curves as shown in Fig. 19.14.

From equation (19.22), it is clear that P/D^5N^3 is a constant. Hence $P \propto N^3$. This means that the curve P v/s N is a cubic curve as shown in Fig. 19.14.

Equation (19.21), shows that $\frac{Q}{D^3N}$ = constant. This means $Q \propto N$ for a given pump. Hence the curve Q v/s N is a straight line as shown in Fig. 19.14.

19.10.2 Operating Characteristic Curves. If the speed is kept constant, the variation of manometric head, power and efficiency with respect to discharge gives the operating characteristics of the pump. Fig. 19.15 shows the operating characteristic curves of a pump.

The input power curve for pumps shall not pass through the origin. It will be slightly away from the origin on the y-axis, as even at zero discharge some power is needed to overcome mechanical losses.

The head curve will have maximum value of head when discharge is zero.

The output power curve will start from origin as at $Q = 0$, output power (ρQgH) will be zero.

The efficiency curve will start from origin as at $Q = 0$, $\eta = 0$ $\left(\because \quad \eta = \frac{\text{Output}}{\text{Input}}\right)$

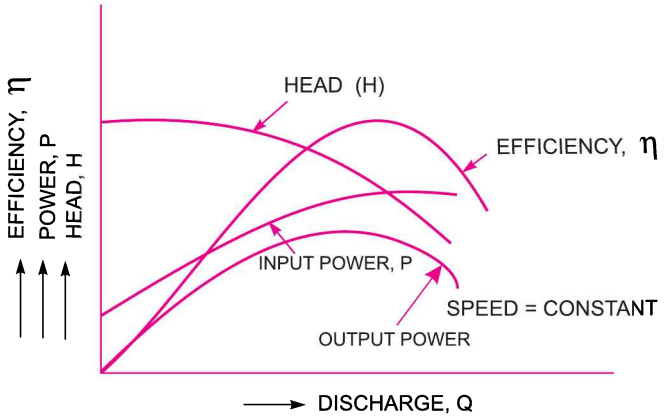


Fig. 19.15 *Operating characteristic curves of a pump.*

19.10.3 Constant Efficiency Curves. For obtaining constant efficiency curves for a pump, the head *versus* discharge curves and efficiency *versus* discharge curves for different speed are used. Fig. 19.16 (*a*) shows the head *versus* discharge curves for different speeds. The efficiency *versus* discharge curves for the different speeds are as shown in Fig. 19.16 (*b*). By combining these curves ($H \sim Q$ curves and $\eta \sim Q$ curves), constant efficiency curves are obtained as shown in Fig. 19.16 *(a)*.

For plotting the constant efficiency curves (also known as iso-efficiency curves), horizontal lines representing constant efficiencies are drawn on the $\eta \sim Q$ curves. The points, at which these lines cut the efficiency curves at various speeds, are transferred to the corresponding $H \sim Q$ curves. The points having the same efficiency are then joined by smooth curves. These smooth curves represents the iso-efficiency curves.

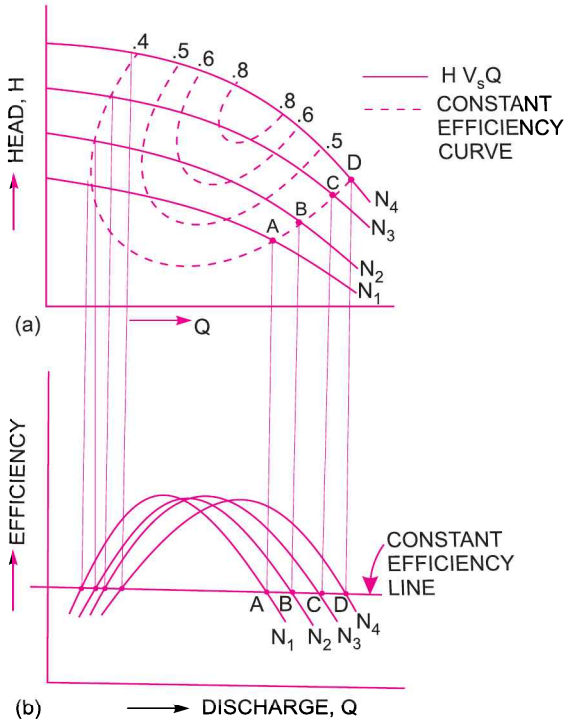


Fig. 19.16 *Constant efficiency curves of a pump.*

▶ 19.11 CAVITATION

Cavitation is defined as the phenomenon of formation of vapour bubbles of a flowing liquid in a region where the pressure of the liquid falls below its vapour pressure and the sudden collapsing of these vapour bubbles in a region of higher pressure. When the vapour bubbles collapse, a very high pressure is created. The metallic surfaces, above which these vapour bubbles collapse, is subjected to these high pressures, which cause pitting action on the surface. Thus cavities are formed on the metallic surface and also considerable noise and vibrations are produced.

Cavitation includes formation of vapour bubbles of the flowing liquid and collapsing of the vapour bubbles. Formation of vapour bubbles of the flowing liquid take place only whenever the pressure in any region falls below vapour pressure. When the pressure of the flowing liquid is less than its vapour pressure, the liquid starts boiling and vapour bubbles are formed. These vapour bubbles are carried along with the flowing liquid to higher pressure zones where these vapours condense and bubbles collapse. Due to sudden collapsing of the bubbles on the metallic surface, high pressure is produced and metallic surfaces are subjected to high local stresses. Thus the surfaces are damaged.

19.11.1 Precaution Against Cavitation. The following precautions should be taken against cavitation :

(*i*) The pressure of the flowing liquid in any part of the hydraulic system should not be allowed to fall below its vapour pressure. If the flowing liquid is water, then the absolute pressure head should not be below 2.5 m of water.

(*ii*) The special materials or coatings such as aluminium-bronze and stainless steel, which are cavitation resistant materials, should be used.

19.11.2 Effects of Cavitation. The following are the effects of cavitation :

(*i*) The metallic surfaces are damaged and cavities are formed on the surfaces.

(*ii*) Due to sudden collapse of vapour bubble, considerable noise and vibrations are produced.

(*iii*) The efficiency of a turbine decreases due to cavitation. Due to pitting action, the surface of the turbine blades becomes rough and the force exerted by water on the turbine blades decreases. Hence, the work done by water or output horse power becomes less and thus efficiency decreases.

19.11.3 Hydraulic Machines Subjected to Cavitation. The hydraulic machines subjected to cavitation are reaction turbines and centrifugal pumps.

19.11.4 Cavitation in Turbines. In turbines, only reaction turbines are subjected to cavitation. In reaction turbines the cavitation may occur at the outlet of the runner or at the inlet of the draft-tube where the pressure is considerably reduced (*i.e.*, which may be below the vapour pressure of the liquid flowing through the turbine). Due to cavitation, the metal of the runner vanes and draft-tube is gradually eaten away, which results in lowering the efficiency of the turbine. Hence, the cavitation in a reaction turbine can be noted by a sudden drop in efficiency. In order to determine whether cavitation will occur in any portion of a reaction turbine, the critical value of Thoma's cavitation factor (σ, sigma) is calculated.

Thoma's Cavitation Factor for Reaction Turbines. Prof. D. Thoma suggested a dimensionless number, called after his name Thoma's cavitation factor σ (sigma), which can be used for determining the region where cavitation takes place in reaction turbines. The mathematical expression for the Thoma's cavitation factor is given by

$$\sigma = \frac{H_b - H_s}{H} = \frac{(H_{atm} - H_v) - H_s}{H} \qquad ...(19.23)$$

where H_b = Barometric pressure head in m of water,

H_{atm} = Atmospheric pressure head in m of water,

H_v = Vapour pressure head in m of water,

H_s = Suction pressure at the outlet of reaction turbine in m of water or height of turbine runner above the tail water surface,

H = Net head on the turbine in m.

19.11.5 Cavitation in Centrifugal Pumps. In centrifugal pumps the cavitation may occur at the inlet of the impeller of the pump, or at the suction side of the pumps, where the pressure is considerably reduced. Hence if the pressure at the suction side of the pump drops below the vapour pressure of the liquid then the cavitation may occur. The cavitation in a pump can be noted by a sudden drop in efficiency and head. In order to determine whether cavitation will occur in any portion of the suction side of the pump, the critical value of Thoma's cavitation factor (σ) is calculated.

Thoma's Cavitation Factor for Centrifugal Pumps. The mathematical expression for Thoma's cavitation factor for centrifugal pump is given by

$$\sigma = \frac{(H_b) - H_S - h_{LS}}{H} = \frac{(H_{atm} - H_V) - H_S - h_{LS}}{H} \qquad ...(19.24)$$

where H_{atm} = Atmospheric pressure head in m of water or absolute pressure head at the liquid surface in pump,
H_V = Vapour pressure head in m of water,
H_S = Suction pressure head in m of water,
h_{LS} = Head lost due to friction in suction pipe, and
H = Head developed by the pump.

The value of Thoma's cavitation factor (σ) for a particular type of turbine or pump is calculated from equations (19.23) or (19.24). This value of Thoma's cavitation factor (σ) is compared with critical cavitation factor (σ_c) for that type of turbine pump. If the value of σ is greater than σ_c, the cavitation will not occur in that turbine or pump. The critical cavitation factor (σ_c) may be obtained from tables or empirical relationships.

The following empirical relationships are used for obtaining the value of σ_c for different turbines :

For Francis turbines,
$$\sigma_c = 0.625 \left(\frac{N_s}{380.78}\right)^2 \qquad ...(19.25)$$
$$\simeq 431 \times 10^{-8} \text{ Ns}^2 \qquad ...(19.26)$$

For Propeller turbines,
$$\sigma_c = 0.28 + \left[\frac{1}{7.5}\left(\frac{N_s}{380.78}\right)^3\right] \qquad ...(19.27)$$

In the above expressions N_s is in (r.p.m., kW, m) units. If N_s is in (r.p.m., h.p., m) units, the empirical relationships would be as follows :

For Francis turbines,
$$\sigma_c = 0.625 \left(\frac{N_s}{444}\right)^2 \qquad ...(19.28)$$
$$\simeq 317 \times 10^{-8} \times N_s^2 \qquad ...(19.29)$$

For Propeller turbines,
$$\sigma_c = 0.28 + \left[\frac{1}{7.5}\left(\frac{N_s}{444}\right)^3\right] \qquad ...(19.30)$$

Problem 19.23 *A Francis turbine has been manufactured to develop 15000 horse power at the head of 81 m and speed 375 r.p.m. The mean atmospheric pressure at the site is 1.03 kgf/cm^2 and vapour pressure 0.03 kgf/cm^2. Calculate the maximum permissible height of the runner above the tail water level to ensure cavitation free operation. The critical cavitation factor for Francis turbine is given by*

$$\sigma_c = 317 \times 10^{-8} \times N_s^2$$

where N_s is the specific speed of the turbine in M.K.S. units.

Solution. Given :

Horse power developed, $P = 15000$
Head, $H = 81$ m
Speed, $N = 375$ r.p.m.
Atmospheric pressure, $p_a = 1.03$ kgf/cm^2 $= 1.03 \times 9.81$ N/cm^2
$= 1.03 \times 9.81 \times 10^4$ N/m^2

$\therefore$ Atmospheric pressure head in meter of water,

$$H_{atm} = \frac{p_a}{\rho g} = \frac{1.03 \times 9.81 \times 10^4}{1000 \times 9.81} = 10.3 \text{ m}$$

Vapour pressure, $p_v = 0.03 \text{ kgf/cm}^2 = 0.03 \times 9.81 \text{ N/cm}^2 = 0.03 \times 9.81 \times 10^4 \text{ N/m}^2$

$\therefore$ Vapour pressure head in meter of water,

$$H_v = \frac{p_v}{\rho g} = \frac{0.03 \times 9.81 \times 10^4}{1000 \times 9.81} = 0.3 \text{ m}$$

Critical cavitation factor, $\sigma_c = 317 \times 10^{-8} \ N_s^2$ ...(*i*)

where N_s is the specific speed of the turbine in M.K.S. units *i.e.,* (r.p.m., h.p., m) units.

Now specific speed of turbine is given by

$$N_s = \frac{N\sqrt{P}}{H^{5/4}} = \frac{375 \times \sqrt{15000}\ *}{81^{5/4}} = 189 \text{ r.p.m.}$$

Substituting this value in equation (*i*), we get

$$\sigma_c = 317 \times 10^{-8} \times 189^2 = 0.1132$$

Now let H_s = Suction pressure head in meter of water at the outlet of Francis turbine or height of the turbine runner above the tail water surface.

Now using equation (19.23), we get

$$\sigma_c = \frac{H_{atm} - H_v - H_s}{H} \quad \text{or} \quad 0.1132 = \frac{10.3 - 0.3 - H_s}{81}$$

or $0.1132 \times 81 = 10 - H_s$ or $H_s = 10 - 0.1132 \times 81 =$ **0.8308 m. Ans.**

Hence, maximum permissible height is 0.83 m above the tail water level.

▶ 19.12 MAXIMUM SUCTION LIFT (or SUCTION HEIGHT)

Fig. 19.17 shows a centrifugal pump that lifts a liquid from a sump. The free surface of liquid is at a depth of h_s below the pump axis. The liquid is flowing with a velocity v_s in the suction pipe.

Let h_s = Suction height (or lift)

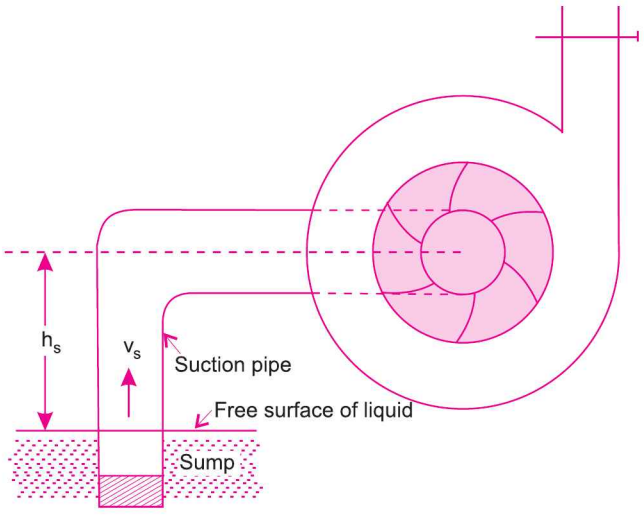


Fig. 19.17

* Here power P should be taken in horse power and not in kW.

Applying Bernoulli's equation at the free surface of liquid in the sump and section 1 in the suction pipe just at the inlet of the pump and taking the free surface of liquid as datum line, we get

$$\frac{p_a}{\rho g} + \frac{V_a^2}{2g} + Z_a = \frac{p_1}{\rho g} + \frac{V_1^2}{2g} + Z_1 + h_L \qquad ...(i)$$

where p_a = Atmospheric pressure on the free surface of liquid,
V_a = Velocity of liquid at the free surface of liquid $\simeq 0$,
Z_a = Height of free surface from datum line = 0,
p_1 = Absolute pressure at the inlet of pump,
V_1 = Velocity of liquid through suction pipe = v_s,
Z_1 = Height of inlet of pump from datum line = h_s,
h_L = Loss of head in the foot valve, strainer and suction pipe = h_{fs}.

Hence equation (*i*), after substituting the above values becomes as

$$\frac{p_a}{\rho g} + 0 + 0 = \frac{p_1}{\rho g} + \frac{v_s^2}{2g} + h_s + h_{fs}$$

or
$$\frac{p_a}{\rho g} = \frac{p_1}{\rho g} + \frac{v_s^2}{2g} + h_s + h_{fs}$$

or
$$\frac{p_1}{\rho g} = \frac{p_a}{\rho g} - \left(\frac{v_s^2}{2g} + h_s + h_{fs}\right) \qquad ...(ii)$$

For finding the maximum suction lift, the pressure at the inlet of the pump should not be less than the vapour pressure of the liquid. Hence for the limiting case, taking the pressure at the inlet of pump equal to vapour pressure of the liquid, we get

$p_1 = p_v$, where p_v = vapour pressure of the liquid in absolute units.

Now the equation (*ii*) becomes as

$$\frac{p_v}{\rho g} = \frac{p_a}{\rho g} - \left(\frac{v_s^2}{2g} + h_s + h_{fs}\right)$$

or
$$\frac{p_a}{\rho g} = \frac{p_v}{\rho g} + \frac{v_s^2}{2g} + h_s + h_{fs} \qquad (\because \quad p_1 = p_v) \ ...(iii)$$

Now
$$\frac{p_a}{\rho g} = \text{Atmospheric pressure head} = H_a \text{ (meter of liquid)}$$

$$\frac{p_v}{\rho g} = \text{Vapour pressure head} = H_v \text{ (meter of liquid)}$$

Now, equation (*iii*) becomes as

$$H_a = H_v + \frac{v_s^2}{2g} + h_s + h_{fs}$$

or
$$h_s = H_a - H_v - \frac{v_s^2}{2g} - h_{fs} \qquad ...(19.31)$$

Equation (19.31) gives the value of maximum suction lift (or maximum suction height) for a centrifugal pump. Hence, the suction height of any pump should not be more than that given by equation (19.31). If the suction height of the pump is more, then vaporization of liquid at inlet of pump will take place and there will be a possibility of cavitation.

▶ 19.13 NET POSITIVE SUCTION HEAD (NPSH)

The term NPSH (Net Positive Suction Head) is very commonly used in the pump industry. Actually the minimum suction conditions are more frequently specified in terms of NPSH.

The net positive suction head (NPSH) is defined as the *absolute* pressure head at the inlet to the pump, minus the vapour pressure head (in absolute units) plus the velocity head.

$\therefore$ NPSH = Absolute pressure head at inlet of the pump – vapour pressure head (absolute units) + velocity head

$$= \frac{p_1}{\rho g} - \frac{p_v}{\rho g} + \frac{v_s^2}{2g} \quad (\because \text{ Absolute pressure at inlet of pump} = p_1) \quad \ldots (19.32)$$

But from equation (*ii*) of Art. 19.12, the absolute pressure head at inlet of the pump is given by as

$$\frac{p_1}{\rho g} = \frac{p_a}{\rho g} - \left(\frac{v_s^2}{2g} + h_s + h_{f_s} \right)$$

Substituting this value in equation (19.32) , we get

$$\text{NPSH} = \left[\frac{p_a}{\rho g} - \left(\frac{v_s^2}{2g} + h_s + h_{f_s} \right) \right] - \frac{p_v}{\rho g} + \frac{v_s^2}{2g}$$

$$= \frac{p_a}{\rho g} - \frac{p_v}{\rho g} - h_s - h_{f_s}$$

$$= H_a - H_v - h_s - h_{f_s}$$

$$\left(\because \frac{p_a}{\rho g} = H_a = \text{Atmospheric pressure head}, \frac{p_v}{\rho g} = H_v = \text{Vapour pressure head} \right)$$

$$= \left[\left(H_a - h_s - h_{f_s} \right) - H_v \right] \quad \ldots (19.33)$$

The right hand side of equation (19.33) is the total suction head. Hence NPSH is equal to total suction head. Thus NPSH may also be defined as the total head required to make the liquid flow through the suction pipe to the pump impeller.

For any pump installation, a distinction is made between the required NPSH and the available NPSH. The value of required NPSH is given by the pump manufacturer. This value can also be determined experimentally. For determining its value, the pump is tested and minimum value of h_s is obtained at which the pump gives maximum efficiency without any objectional noise (*i.e.,* cavitation free). The required NPSH varies with the pump design, speed of the pump and capacity of the pump.

When the pump is installed, the available NPSH is calculated from equation (19.33). In order to have cavitation free operation of centrifugal pump, the available NPSH should be greater than the required NPSH.

▶ 19.14 CAVITATION IN CENTRIFUGAL PUMP

Thoma's cavitation factor is used to indicate whether cavitation will occur in pumps. Equation (19.24) gives the value of Thoma's cavitation factor for pumps as

$$\sigma = \frac{\left(H_{atm} - H_v \right) - H_s - h_{f_s}}{H}$$

$$= \frac{H_a - H_v - h_s - h_{f_s}}{H_m} \quad (\because H_s = h_s \text{ and } h_{L_s} = h_{f_s}) \ (H = H_m \text{ for pumps})$$

But from equation (19.33), we have

$$H_a - H_v - h_s - h_{f_s} = \text{NPSH} \qquad \text{(Net position suction head)}$$

$$\therefore \qquad \sigma = \frac{\text{NPSH}}{H_m} \qquad ...(19.34)$$

If the value of σ (calculated from equation 19.34) is less than the critical value, σ_c then cavitation will occur in the pumps. The value of σ_c depends upon the specific speed of the pump $\left(N_s = \frac{N\sqrt{Q}}{H_m^{3/4}}\right)$. The following empirical relation is used to determine the value of σ_c.

$$\sigma_c = 0.103 \left(\frac{N_s}{1000}\right)^{4/3}$$

$$= 0.103 \frac{N_s^{4/3}}{\left(10^3\right)^{4/3}} = \frac{0.103\, N_s^{4/3}}{10^4}$$

$$= 1.03 \times 10^{-3}\, N_s^{4/3} \qquad ...(19.35)$$

Problem 19.24 *A centrifugal pump rotating at 1000 r.p.m. delivers 160 litres/s of water against a head of 30 m. The pump is installed at a place where atmospheric pressure is 1 $\times 10^5$ P_a (abs.) and vapour pressure of water is 3 kPa (abs.). The head loss in suction pipe is equivalent to 0.2 m of water. Calculate :*

(i) Minimum NPSH, and

(ii) Maximum allowable height of the pump from free surface of water in the sump.

Solution. Given :

N = 1000 r.p.m.; Q = 160 litres/s = 0.16 m^3/s.; H_m = 30 m

$p_a = 1 \times 10^5$ Pa (abs.) = 1×10^5 N/m^2 (abs.) ; $p_v = 3$ kPa (abs.) = 3×10^3 N/m^2 (abs.)

$h_{f_s} = 0.2$ m.

(*i*) *Minimum NPSH*

Using equation (19.34), we get

$$\sigma = \frac{\text{NPSH}}{H_m}$$

From the above equation, it is clear that NPSH is directly proportional to Thoma's cavitation factor (σ). NPSH will be minimum when σ is minimum. But the minimum value of σ for no cavitation is σ_c. Hence when $\sigma = \sigma_c$ then NPSH will be minimum.

$$\therefore \qquad \sigma_c = \frac{(\text{NPSH})_{\min}}{H_m}$$

or $$(\text{NPSH})_{\min} = H_m \times \sigma_c \qquad ...(i)$$

Now the critical value of σ *i.e.*, σ_c is given by equation (19.35) as

$$\sigma_c = 1.03 \times 10^{-3} \times N_s^{4/3} \qquad ...(ii)$$

where N_s = Specific speed of pump = $\frac{N\sqrt{Q}}{H_m^{3/4}}$

$$= 1000 \times \frac{\sqrt{0.16}}{30^{3/4}} \quad (\because N = 1000 \text{ r.p.m.}, Q = 0.16 \text{ m}^3/\text{s and } H_m = 30 \text{ m})$$

Substituting the value of N_s in equation (*ii*), we get

$$\sigma_c = 1.03 \times 10^{-3} \times \left[\frac{1000 \times \sqrt{0.16}}{30^{3/4}}\right]^{4/3}$$

$$= 1.03 \times 10^{-3} \times \frac{1000^{4/3} \times 0.16^{2/3}}{30} = \frac{1.03 \times 10^{-3} \times 10^4 \times 0.2947}{30}$$

$$= 0.1012$$

Substituting the value of σ_c in equation (*i*), we get

$$(\text{NPSH})_{\text{min.}} = H_m \times 0.1012$$

$$= 30 \times 0.1012 = \textbf{3.036 m. Ans.} \qquad (\because \quad H_m = 30 \text{ m})$$

(*ii*) *Maximum allowable height of the pump from free surface of water in the sump* (*i.e.*, h_s)

Let $(h_s)_{\max}$ = Max. allowable height of pump from free surface of water.

Using equation (19.33)

$$\text{NPSH} = H_a - H_v - h_s - h_{f_s} \qquad ...(i)$$

From the above equation, it is clear that for a given value of atmospheric pressure head $\left(H_a = \frac{p_a}{\rho g}\right)$, given vapour pressure head $\left(H_v = \frac{p_v}{\rho g}\right)$ and given loss of head due to friction (h_{f_s}), the value of suction head (h_s) will be maximum if NPSH is minimum.

$$\therefore \qquad (\text{NPSH})_{\min} = H_a - H_v - (h_s)_{\max} - h_{f_s} \qquad ...(ii)$$

$$\therefore \qquad (h_s)_{\max} = H_a - H_v - h_{f_s} - (\text{NPSH})_{\min} \qquad ...(iii)$$

Now $$H_a = \frac{p_a}{\rho g} = \frac{1 \times 10^5}{1000 \times 9.81} = 10.193 \text{ m of water}$$

$$H_v = \frac{p_v}{\rho g} = \frac{3 \times 10^3}{1000 \times 9.81} = 0.305 \text{ m of water}$$

$$h_{f_s} = 0.2 \text{ m and } (\text{NPSH})_{\min} = 3.036 \text{ m}$$

Substituting the values of H_a, H_v, h_{f_s} and $(\text{NPSH})_{\text{min.}}$ in equation (*iii*), we get

$$(h_s)_{\max} = 10.193 - 0.305 - 0.2 - 3.036$$

$$= \textbf{6.652 m. Ans.}$$

HIGHLIGHTS

1. The hydraulic machine which converts the mechanical energy into pressure energy by means of centrifugal force is called centrifugal pump.
2. The centrifugal pump acts as a reverse of an inward radial flow reaction turbine. The work done by the impeller (rotating part of the pump) on the water per second per unit weight of water per second flowing through the pump is given as

$$= \frac{1}{g} V_{w_2} \times u_2$$

where V_{w_2} = Velocity of whirl at outlet, and

u_2 = Tangential velocity of wheel at outlet.

3. The vertical height of the centre-line of the centrifugal pump from the water surface in the pump is called the suction head (h_s).

4. Delivery head (h_d) is the vertical distance between the centre-line of the pump and the water surface in the tank to which water is lifted.

5. Manometric head (H_m) is the head against which a centrifugal pump has to work. It is given as

(*a*) $$H_m = \frac{V_{w_2} \times u_2}{g} - \text{Loss of head in impeller and casing}$$

$$= \frac{V_{w_2} \times u_2}{g} \text{ ...if losses in pump is zero}$$

(*b*) H_m = Total head at outlet – Total head at inlet of pump

$$= \left(\frac{p_o}{\rho g} + \frac{V_o^2}{2g} + Z_o\right) - \left(\frac{p_i}{\rho g} + \frac{V_i^2}{2g} + Z_i\right)$$

(*c*) $$H_m = h_s + h_d + h_{f_s} + h_{f_d} + \frac{V_d^2}{2g}.$$

6. The efficiencies of a pump are : (*i*) Manometric efficiency (h_{man}), (*ii*) Mechanical efficiency (η_m), and (*iii*) Overall efficiency (η_o). Mathematically they are given as

$$\eta_{man} = \frac{gH_m}{V_{w_2} \times u_2}$$

$$\eta_m = \frac{\frac{W}{g}\left(\frac{V_{w_2} \times u}{75}\right)}{\text{S.P.}}, \text{ where } W = w \times Q$$

$$\eta_o = \frac{W \times H_m}{1000 \times \text{S.P.}}$$

7. The minimum speed for starting a centrifugal pump is given by $N = \dfrac{120 \times \eta_{man} \times V_{w_2} \times D_2}{\pi\left[D_2^2 - D_1^2\right]}$.

8. If a centrifugal pump consists of two or more impellers, the pump is called a multistage pump. To produce a high head, the impellers are connected in series while to discharge a large quantity of liquid, the impellers are connected in parallel.

9. The specific speed of a centrifugal pump is defined as the speed at which a pump runs when the head developed is one metre and discharge is one cubic metre. Mathematically, it is given as

$$N_s = \frac{N\sqrt{Q}}{H_m^{3/4}}, \text{ where } H_m = \text{Manometric hand.}$$

10. For complete similarity between the model and actual centrifugal pump (prototype) the following conditions should be satisfied :

(*a*) $\left(\frac{N\sqrt{Q}}{H_m^{3/4}}\right)_{\text{model}} = \left(\frac{N\sqrt{Q}}{H_m^{3/4}}\right)_{\text{prototype}}$ (*b*) $\left(\frac{\sqrt{H_m}}{DN}\right)_{\text{model}} = \left(\frac{\sqrt{H_m}}{DN}\right)_{\text{prototype}}$

(*c*) $\left(\frac{Q}{D^3N}\right)_{\text{model}} = \left(\frac{Q}{D^3N}\right)_{\text{prototype}}$ (*d*) $\left(\frac{P}{D^5N^3}\right)_{\text{model}} = \left(\frac{P}{D^5N^3}\right)_{\text{prototype}}$

11. Characteristic curves are used for predicting the behaviour and performance of a pump when it is working under different flow rate, head and speed.

12. Cavitation is defined as the phenomenon of formation of vapour bubbles and sudden collapsing of the vapour bubbles.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Define a centrifugal pump. Explain the working of a single-stage centrifugal pump with sketches.

2. Differentiate between the volute casing and vortex casing for the centrifugal pump.

3. Obtain an expression for the work done by impeller of a centrifugal pump on water per second per unit weight of water.

4. Define the terms : suction head, delivery head, static head and manometric head.

5. What do you mean by manometric efficiency, mechanical efficiency and overall efficiency of a centrifugal pump?

6. How will you obtain an expression for the minimum speed for starting a centrifugal pump?

7. What is the difference between single-stage and multistage pumps? Describe multistage pump with (*a*) impellers in parallel, and (*b*) impellers in series.

8. Define specific speed of a centrifugal pump. Derive an expression for the same.

9. How does the specific speed of a centrifugal pump differ from that of a turbine ?

10. What is priming ? Why is it necessary ?

11. How the model testing of the centrifugal pumps are made?

12. What do you understand by characteristic curves of a pump? What is the significance of the characteristic curves?

13. Define cavitation. What are the effects of cavitation ? Give the necessary precautions against cavitation.

14. How will you determine the possibility of the cavitation to occur in the installation of a turbine or a pump?

15. Why are centrifugal pumps used sometimes in series and sometimes in parallel ? Draw the following characteristic curves for a centrifugal pump :

Head, power and efficiency *versus* discharge with constant speed.

16. Draw and discuss the operating characteristics of a centrifugal pump.

17. (*a*) What is cavitation and what are its causes ? How will you prevent the cavitation in hydraulic machines ?

(*b*) What is cavitation? State its effects on the performance of water turbines and also state how to prevent cavitation in water turbines.

18. Briefly state the significance of similarity parameters in hydraulic pumps.

19. The frictional torque T of a disc of diameter D rotating at a speed of N in a fluid of viscosity μ and density ρ in a turbulent flow is given by :

$$T = D^5 N^2 \rho \phi \left[\frac{\mu}{D^2 N \rho} \right]$$

Prove this by method of dimensions.

20. With a neat sketch, explain the principle and working of a centrifugal pump.

21. Explain the following terms as they are applied to a centrifugal pump :
(*i*) Static suction lift ; (*ii*) static suction head ; (*iii*) static discharge head ; and (*iv*) total static head.

22. (*a*) How does a volute casing differ from a vortex casing for the centrifugal pump ?
(*b*) What is priming ? Why is it necessary ?
(*c*) What do you mean by pump characteristics ? Briefly explain the uses of such characteristics.
(*Jawaharlal Nehru Technical University, S 2002*)

(B) NUMERICAL PROBLEMS

1. The internal and external diameters of the impeller of a centrifugal pump are 300 mm and 600 mm respectively. The pump is running at 1000 r.p.m. The vane angles at inlet and outlet are 20° and 30° respectively. The water enters the impeller radially and velocity of flow is constant. Determine the work done by the impeller per unit weight of water. [**Ans.** 68.89 Nm/N]

2. A centrifugal pump having outer diameter equal to two times the inner diameter and running at 1200 r.p.m. works against a total head of 75 m. The velocity of flow through the impeller is constant and equal to 3 m/s. The vanes are set back at an angle of 30° at outlet. If the outer diameter of the impeller is 600 mm and width at outlet is 50 mm, determine : (*a*) vane angle at inlet, (*b*) work done per second by impeller, (*c*) manometric efficiency. [**Ans.** (*a*) 9° 2′, (*b*) 346.37 kW, (*c*) 60%]

3. A centrifugal pump is running at 1000 r.p.m. The outlet vane angle of the impeller is 30° and velocity of flow at outlet is 3 m/s. The pump is working against a total head of 30 m and the discharge through the pump is 0.3 m^3/s. If the manometric efficiency of the pump is 75%, determine: (*i*) the diameter of the impeller, and (*ii*) the width of the impeller at outlet. [**Ans.** (*i*) 43.1 cm, (*ii*) 7.4 cm]

4. Find the power required to drive a centrifugal pump which delivers 0.02 m^3/s of water to a height of 30 m through a 10 cm diameter pipe and 90 m long. The overall efficiency of the pump is 70% and $f = .009$ in the formula

$$h_f = \frac{4 f l V^2}{d \times 2g}.$$

[**Ans.** 11.5 kW]

5. Find the rise in pressure in the impeller of a centrifugal pump through which water is flowing at the rate of 15 litre/s. The internal and external diameters of the impeller are 20 cm and 40 cm respectively. The widths of impeller at inlet and outlet are 1.6 cm and 0.8 cm. The pump is running at 1200 r.p.m. The water enters the impeller radially at inlet and impeller vane angle at outlet is 30°. Neglect losses through the impeller. [**Ans.** 31.85 m]

6. The diameters of an impeller of a centrifugal pump at inlet and outlet are 20 cm and 40 cm respectively. Determine the minimum speed for starting the pump if it works against a head of 25 m.
[**Ans.** 1221.2 r.p.m.]

7. The diameter of an impeller of a centrifugal pump at inlet and outlet are 300 mm and 600 mm respectively. The velocity of flow at outlet is 2.5 m/s and vanes are set back at an angle of 45° at outlet. Determine the minimum starting speed of the pump if the manometric efficiency is 75%. [**Ans.** 159.31 r.p.m.]

8. A three-stage centrifugal pump has impeller 40 cm in diameter and 2.5 cm wide at outlet. The vanes are curved back at the outlet at 30° and reduce the circumferential area by 15%. The manometric efficiency is 85% and overall efficiency is 75%. Determine the head generated by the pump when running at 12000 r.p.m. and discharge is 0.06 m^3/s. Find the shaft power also. [**Ans.** 138.75 m, 108.89 kW]

9. Find the number of pumps required to take water from a deep well under a total head of 156 m. Also the pumps are identical and are running at 1000 r.p.m. The specific speed of each pump is given as 20 while the rated capacity of each pump is 150 litre/s. [**Ans.** 3]

10. The diameter of a centrifugal pump, which is discharging 0.035 m^3/s of water against a total head of 25 m is 0.05 m. The pump is running at 1200 r.p.m. Find the head, discharge and ratio of powers of a geometrically similar pump of diameter 0.3 m when it is running at 2000 r.p.m. [**Ans.** 25 m, .0126 m^3/s, 2.777]

11. A centrifugal pump is to discharge 0.12 m^3 at a speed of 1400 r.p.m. against a head of 30 m. The diameter and width of the impeller at outlet are 25 cm and 5 cm respectively. If the manometric efficiency is 75%, determine the vane angle at outlet.

20 CHAPTER

RECIPROCATING PUMPS

▶ 20.1 INTRODUCTION

In the last chapter, we have defined the pumps as the hydraulic machines which convert the mechanical energy into hydraulic energy which is mainly in the form of pressure energy. If the mechanical energy is converted into hydraulic energy, by means of centrifugal force acting on the liquid, the pump is known as centrifugal pump. But if the mechanical energy is converted into hydraulic energy (or pressure energy) by sucking the liquid into a cylinder in which a piston is reciprocating (moving backwards and forwards), which exerts the thrust on the liquid and increases its hydraulic energy (pressure energy), the pump is known as reciprocating pump.

▶ 20.2 MAIN PARTS OF A RECIPROCATING PUMP

The following are the main parts of a reciprocating pump as shown in Fig. 20.1 :

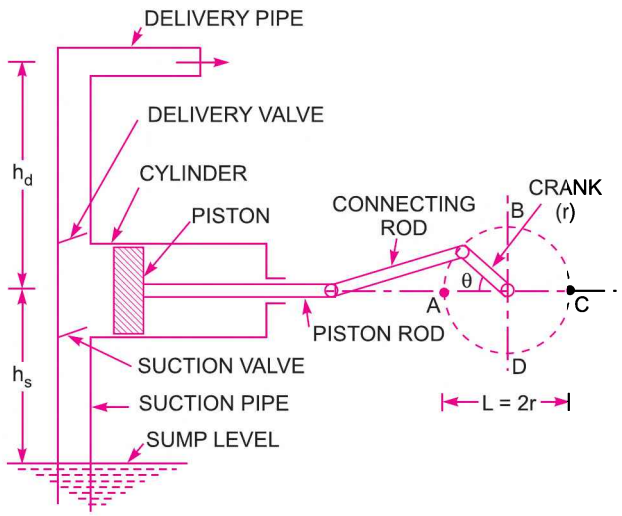


Fig. 20.1 *Main parts of a reciprocating pump.*

1. A cylinder with a piston, piston rod, connecting rod and a crank,
2. Suction pipe, 3. Delivery pipe,
4. Suction valve, and 5. Delivery valve.

▶ 20.3 WORKING OF A RECIPROCATING PUMP

Fig. 20.1 shows a single acting reciprocating pump, which consists of a piston which moves forwards and backwards in a close fitting cylinder. The movement of the piston is obtained by connecting the piston rod to crank by means of a connecting rod. The crank is rotated by means of an electric motor. Suction and delivery pipes with suction valve and delivery valve are connected to the cylinder. The suction and delivery valves are one way valves or non-return valves, which allow the water to flow in one direction only. Suction valve allows water from suction pipe to the cylinder which delivery valve allows water from cylinder to delivery pipe only.

When crank starts rotating, the piston moves to and fro in the cylinder. When crank is at A., the piston is at the extreme left position in the cylinder. As the crank is rotating from A to C, (*i.e.*, from $\theta = 0^o$ to $\theta = 180°$), the piston is moving towards right in the cylinder. The movement of the piston towards right creates a partial vacuum in the cylinder. But on the surface of the liquid in the sump atmospheric pressure is acting, which is more than the pressure inside the cylinder. Thus, the liquid is forced in the suction pipe from the sump. This liquid opens the suction valve and enters the cylinder.

When crank is rotating from C to A (*i.e.*, from $\theta = 180°$ to $\theta = 360°$), the piston from its extreme right position starts moving towards left in the cylinder. The movement of the piston towards left increases the pressure of the liquid inside the cylinder more than atmospheric pressure. Hence suction valve closes and delivery valve opens. The liquid is forced into the delivery pipe and is raised to a required height.

20.3.1 Discharge Through a Reciprocating Pump. Consider a single* acting reciprocating pump as shown in Fig. 20.1.

Let D = Diameter of the cylinder

A = Cross-sectional area of the piston or cylinder

$$= \frac{\pi}{4} D^2$$

r = Radius of crank

N = r.p.m. of the crank

L = Length of the stroke = $2 \times r$

h_s = Height of the axis of the cylinder from water surface in sump.

h_d = Height of delivery outlet above the cylinder axis (also called delivery head)

Volume of water delivered in one revolution or discharge of water in one revolution

$$= \text{Area} \times \text{Length of stroke} = A \times L$$

Number of revolution per second, $= \frac{N}{60}$

$\therefore$ Discharge of the pump per second,

Q = Discharge in one revolution × No. of revolution per second

$$= A \times L \times \frac{N}{60} = \frac{ALN}{60} \qquad ...(20.1)$$

* Single acting means the water is acting on one side of the piston only.

Weight of water delivered per second,

$$W = \rho \times g \times Q = \frac{\rho g A L N}{60}. \qquad ...(20.2)$$

20.3.2 Work done by Reciprocating Pump. Work done by the reciprocating pump per second is given by the reaction as

Work done per second = Weight of water lifted per second × Total height through which water is lifted

$$= W \times (h_s + h_d) \qquad ...(i)$$

where $(h_s + h_d)$ = Total height through which water is lifted.

From equation (20.2), Weight, W, is given by

$$W = \frac{\rho g \times A L N}{60}.$$

Substituting the value of W in equation (i), we get

$$\text{Work done per second} = \frac{\rho g \times A L N}{60} \times (h_s + h_d) \qquad ...(20.3)$$

$\therefore$ Power required to drive the pump, in kW

$$P = \frac{\text{Work done per second}}{1000} = \frac{\rho g \times A L N \times (h_s + h_d)}{60 \times 1000}$$

$$= \frac{\rho g \times A L N \times (h_s + h_d)}{60{,}000} \text{ kW} \qquad ...(20.4)$$

20.3.3 Discharge, Work done and Power Required to Drive a Double-acting Pump. In case of double-acting pump, the water is acting on both sides of the piston as shown in Fig. 20.2. Thus, we require two suction pipes and two delivery pipes for double-acting pump. When there is a suction stroke on one side of the piston, there is at the same time a delivery stroke on the other side of the piston. Thus for one complete revolution of the crank there are two delivery strokes and water is delivered to the pipes by the pump during these two delivery strokes.

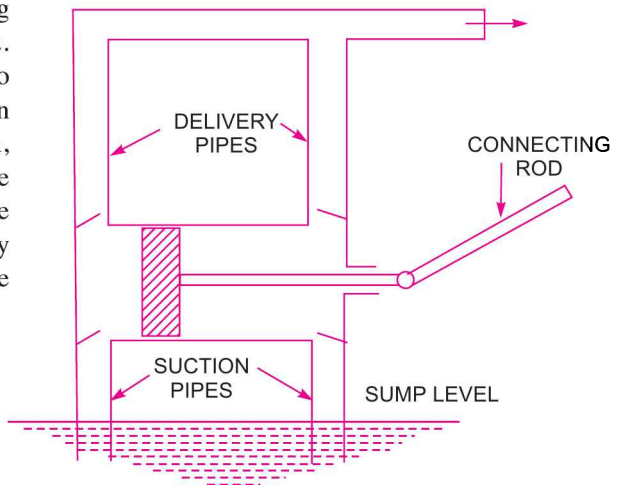


Fig. 20.2

Let D = Diameter of the piston,

d = Diameter of the piston rod

$\therefore$ Area on one side of the piston,

$$A = \frac{\pi}{4} D^2$$

Area on the other side of the piston, where piston rod is connected to the piston,

$$A_1 = \frac{\pi}{4} D^2 - \frac{\pi}{4} d^2 = \frac{\pi}{4} (D^2 - d^2).$$

∴ Volume of water delivered in one revolution of crank

$$= A \times \text{Length of stroke} + A_1 \times \text{Length of stroke}$$

$$= AL + A_1L = (A + A_1)L = \left[\frac{\pi}{4}D^2 + \frac{\pi}{4}\left(D^2 - d^2\right)\right] \times L$$

∴ Discharge of pump per second

= Volume of water delivered in one revolution × No. of revolution per second

$$= \left[\frac{\pi}{4}D^2 + \frac{\pi}{4}\left(D^2 - d^2\right)\right] \times L \times \frac{N}{60}$$

If 'd' the diameter of the piston rod is very small as compared to the diameter of the piston, then it can be neglected and discharge of pump per second,

$$Q = \left(\frac{\pi}{4}D^2 + \frac{\pi}{4}D^2\right) \times \frac{L \times N}{60} = 2 \times \frac{\pi}{4}D^2 \times \frac{L \times N}{60} = \frac{2ALN}{60} \quad ...(20.5)$$

Equation (20.5) gives the discharge of a double-acting reciprocating pump. This discharge is two times the discharge of a single-acting pump.

Work done by double-acting reciprocating pump

Work done per second = Weight of water delivered × Total height

$$= \rho g \times \text{Discharge per second} \times \text{Total height}$$

$$= \rho g \times \frac{2ALN}{60} \times (h_s + h_d) = 2\rho g \times \frac{ALN}{60} \times (h_s + h_d) \quad ...(20.6)$$

∴ Power required to drive the double-acting pump in kW,

$$P = \frac{\text{Work done per second}}{1000} = 2\rho g \times \frac{ALN}{60} \times \frac{(h_s + h_d)}{1000}$$

$$= \frac{2\rho g \times ALN \times (h_s + h_d)}{60{,}000} \quad ...(20.7)$$

▶ 20.4 SLIP OF RECIPROCATING PUMP

Slip of a pump is defined as the difference between the theoretical discharge and actual discharge of the pump. The discharge of a single-acting pump given by equation (20.1) and of a double-acting pump given by equation (20.5) are theoretical discharge. The actual discharge of a pump is less than the theoretical discharge due to leakage. The difference of the theoretical discharge and actual discharge is known as slip of the pump. Hence, mathematically.

$$\text{Slip} = Q_{th} - Q_{act} \quad ...(20.8)$$

But slip is mostly expressed as percentage slip which is given by,

$$\text{Percentage slip} = \frac{Q_{th} - Q_{act}}{Q_{th}} \times 100 = \left(1 - \frac{Q_{act}}{Q_{th}}\right) \times 100$$

$$= (1 - C_d) \times 100 \qquad \left(\because \frac{Q_{act}}{Q_{th}} = C_d\right) \quad ...(20.9)$$

where C_d = Co-efficient of discharge.

20.4.1 Negative Slip of the Reciprocating Pump. Slip is equal to the difference of theoretical discharge and actual discharge. If actual discharge is more than the theoretical discharge, the slip of the pump will become –ve. In that case, the slip of the pump is known as negative slip.

Negative slip occurs when delivery pipe is short, suction pipe is long and pump is running at high speed.

▶ 20.5 CLASSIFICATION OF RECIPROCATING PUMPS

The reciprocating pumps may be classified as :

1. According to the water being in contact with one side or both sides of the piston, and
2. According to the number of cylinders provided.

If the water is in contact with one side of the piston, the pump is known as single-acting. On the other hand, if the water is in contact with both sides of the piston, the pump is called double-acting. Hence, classification according to the contact of water is :

(*i*) Single-acting pump, and (*ii*) Double-acting pump.

According to the number of cylinder provided, the pumps are classified as :

(*i*) Single cylinder pump, (*ii*) Double cylinder pump, and

(*iii*) Triple cylinder pump.

Problem 20.1 *A single-acting reciprocating pump, running at 50 r.p.m., delivers 0.01 m^3/s of water. The diameter of the piston is 200 mm and stroke length 400 mm. Determine :*

(i) The theoretical discharge of the pump, (ii) Co-efficient of discharge, and (iii) Slip and the percentage slip of the pump.

Solution. Given :

Speed of the pump, $N = 50$ r.p.m.

Actual discharge, $Q_{act} = .01\ m^3/s$

Dia. of piston, $D = 200\ mm = .20\ m$

$\therefore$ Area, $A = \frac{\pi}{4}(.2)^2 = .031416\ m^2$

Stroke, $L = 400\ mm = 0.40\ m.$

(*i*) Theoretical discharge for single-acting reciprocating pump is given by equation (20.1) as

$$Q_{th} = \frac{A \times L \times N}{60} = \frac{.031416 \times .40 \times 50}{60} = \mathbf{0.01047\ m^3/s.\ Ans.}$$

(*ii*) Co-efficient of discharge is given by

$$C_d = \frac{Q_{act}}{Q_{th}} = \frac{0.01}{.01047} = \mathbf{0.955.\ Ans.}$$

(*iii*) Using equation (20.8), we get

$$\text{Slip} = Q_{th} - Q_{act} = .01047 - .01 = \mathbf{0.00047\ m^3/s.\ Ans.}$$

And percentage slip $= \frac{(Q_{th} - Q_{act})}{Q_{th}} \times 100 = \frac{(.01047 - .01)}{.01047} \times 100$

$$= \frac{.00047}{.01047} \times 100 = \mathbf{4.489\%.\ Ans.}$$

Problem 20.2 *A double-acting reciprocating pump, running at 40 r.p.m., is discharging 1.0 m^3 of water per minute. The pump has a stroke of 400 mm. The diameter of the piston is 200 mm. The delivery and suction head are 20 m and 5 m respectively. Find the slip of the pump and power required to drive the pump.*

Solution. Given:

Speed of pump, $N = 40$ r.p.m.

Actual discharge, $Q_{act} = 1.0 \text{ m}^3/\text{min} = \frac{1.0}{60} \text{ m}^3/\text{s} = 0.01666 \text{ m}^3/\text{s}$

Stroke, $L = 400 \text{ mm} = 0.40 \text{ m}$

Diameter of piston, $D = 200 \text{ mm} = 0.20 \text{ m}$

$\therefore$ Area, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} (.2)^2 = 0.031416 \text{ m}^2$

Suction head, $h_s = 5$ m

Delivery head, $h_d = 20$ m.

Theoretical discharge for double-acting pump is given by equation (20.5) as,

$$Q_{th} = \frac{2ALN}{60} = \frac{2 \times .031416 \times 0.4 \times 40}{60} = .01675 \text{ m}^3/\text{s}.$$

Using equation (20.8), Slip $= Q_{th} - Q_{act} = .01675 - .01666 =$ **.00009 m^3/s. Ans.**

Power required to drive the double-acting pump is given by equation (20.7) as,

$$P = \frac{2 \times \rho g \times ALN \times (h_s + h_d)}{60{,}000} = \frac{2 \times 1000 \times 9.81 \times .031416 \times .4 \times 40 \times (5 + 20)}{60{,}000}$$

= 4.109 kW. Ans.

▶ 20.6 VARIATION OF VELOCITY AND ACCELERATION IN THE SUCTION AND DELIVERY PIPES DUE TO ACCELERATION OF THE PISTON

It is mentioned in Art. 20.3 that when crank starts rotating, the piston moves forwards and backwards in the cylinder. At the extreme left position and right position of the piston in the cylinder, the velocity of the piston is zero. The velocity of the piston is maximum at the centre of the cylinder. This means that at the start of a stroke (may be suction or delivery stroke), the velocity of the piston is zero and this velocity becomes maximum at the centre of each stroke and again becomes zero at the end of each stroke. Thus at the beginning of each stroke, the piston will be having an acceleration and at the end of each stroke, the piston will be having a retardation. The water in the cylinder is in contact with the piston and hence the water, flowing from the suction pipe or to the delivery pipe will have an acceleration at the beginning of each stroke and a retardation at the end of each stroke. This means the velocity of flow of water in the suction and delivery pipe will not be uniform. Hence, an accelerative or retarding head will be acting on the water flowing through the suction or delivery pipe. This accelerative or retarding head will change the pressure inside the cylinder.

If the ratio of length of connecting rod to the radius of crank (*i.e., L/r*) is very large, then the motion of the piston can be assumed as simple harmonic in nature. Fig. 20.3 shows the cylinder of a reciprocating single-acting pump, fitted with a piston which is connected to the crank. Let the crank is rotating at a constant angular speed.

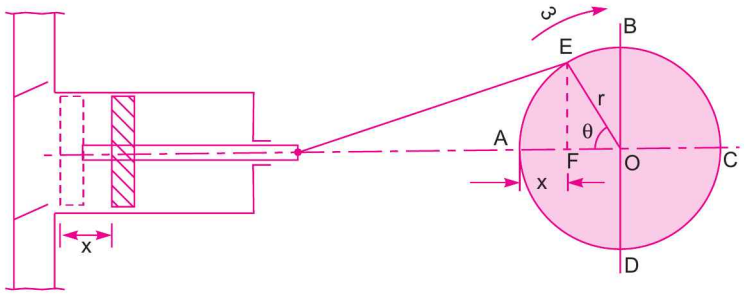


Fig. 20.3 *Velocity and acceleration of piston.*

Let ω = Angular speed of the crank in rad./s,

A = Area of the cylinder,

a = Area of the pipe (suction or delivery),

l = Length of the pipe (suction or delivery), and

r = Radius of the crank.

In the beginning, the crank is at A (which is called inner dead centre) and the piston in the cylinder is at a position shown by dotted lines. The crank is rotating with an angular velocity ω and let in time 't' seconds, the crank turns through an angle θ (in radians) from A (*i.e.*, inner dead centre). The displacement of the piston in time 't' is 'x' as shown in Fig. 20.3.

Now θ = Angle turned by crank in radians in time 't'

$= \omega t$ ...(*i*)

The distance x travelled by the piston is given as

x = Distance $AF = AO - FO$

$= r - r \cos \theta$ ($\because AO = r$, $FO = r \cos \theta$)

$= r - r \cos (\omega t)$ ($\because$ From (*i*), $\theta = \omega t$) ...(*ii*)

The velocity of the piston is obtained by differentiating equation (*ii*) with respect to 't' .

$\therefore$ Velocity of piston, $V = \dfrac{dx}{dt} = \dfrac{d}{dt}[r - r \cos (\omega t)]$

$= 0 - r [- \sin \omega t] \times \omega$ ($\because$ r is constant)

$= \omega r \sin \omega t.$ ...(20.10)

Now from continuity equation, the volume of water flowing into cylinder per second is equal to the volume of water flowing from the pipe per second.

$\therefore$ Velocity of water in cylinder $\times$ Area of cylinder

= Velocity of water in pipe $\times$ Area of pipe

or $V \times A = v \times a$ ($\because$ Velocity of water in cylinder = Velocity of piston = V)

where v = Velocity of water in pipe

$\therefore$ $v = \dfrac{V \times A}{a} = \dfrac{A}{a} \times V$

$= \dfrac{A}{a} \omega r \sin \omega t$ [$\because$ From (20.10), $V = \omega r \sin \omega t$] ...(20.11)

The acceleration of water in pipe is obtained by differentiating equation (20.11) with respect to *'t'*.

∴ Acceleration of water in pipe

$$= \frac{dv}{dt} = \frac{d}{dt}\left(\frac{A}{a}\omega r \sin \omega t\right) = \frac{A}{a}\omega^2 \, r \cos \omega t \qquad \text{....(20.12)}$$

Mass of water in pipe = ρ × Volume of water in pipe

= ρ × [Area of pipe × Length of pipe] = ρ × [$a \times l$] = ρal

∴ Force required to accelerate the water in the pipe

= Mass of water in pipe × Acceleration of water in pipe

$$= \rho al \times \frac{A}{a}\omega^2 \, r \cos \omega t$$

∴ Intensity of pressure due to acceleration

$$= \frac{\text{Force required to accelerate the water}}{\text{Area of pipe}}$$

$$= \frac{\rho al \times \frac{A}{a}\omega^2 r \cos \omega t}{a} = \rho l \times \frac{A}{a}\omega^2 r \cos \omega t$$

$$= \rho l \times \frac{A}{a}\omega^2 r \cos \theta \qquad (\because \ \omega t = \theta)$$

∴ Pressure head (h_a) due to acceleration

$$h_a = \frac{\text{Intensity of pressure due to acceleration}}{\text{Weight density of liquid}}$$

$$= \frac{\rho l \times \frac{A}{a}\omega^2 r \cos \theta}{\rho g} = \frac{l}{g} \times \frac{A}{a}\omega^2 r \cos \theta. \qquad \text{...(20.13)}$$

The pressure head due to acceleration in the suction and delivery pipes is obtained from equation (20.13) by using subscripts *'s'* and *'d'* as

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s}\omega^2 r \cos \theta \qquad \text{...(20.14)}$$

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d}\omega^2 r \cos \theta. \qquad \text{...(20.15)}$$

The pressure head (h_a) due to acceleration, given by equation (20.13) varies with θ. The values of *'h_a'* for different values of θ are :

1. When θ = 0°, $h_a = \frac{l}{g} \times \frac{A}{a}\omega^2 r$ as cos 0° =1
2. When θ = 90°, $h_a = 0$ as cos 90° = 0
3. When θ = 180°, $h_a = -\frac{l}{g} \times \frac{A}{a}\omega^2 r$ as cos 180° = – 1

$\therefore$ Maximum pressure head due to acceleration

$$(h_a)_{max} = \frac{l}{g} \times \frac{A}{a} \omega^2 r \qquad ...(20.16)$$

▶ 20.7 EFFECT OF VARIATION OF VELOCITY ON FRICTION IN THE SUCTION AND DELIVERY PIPES

The velocity of water in suction or delivery pipe is given by equation (20.11) as

$$v = \frac{A}{a} \omega r \sin \omega t = \frac{A}{a} \omega r \sin \theta \qquad ...(i)$$

Loss of head due to friction in pipes is given by

$$h_f = \frac{4 f l v^2}{d \times 2g} \qquad ...(ii)$$

where f = Co-efficient of friction, l = Length of pipe,
d = Diameter of pipe, and v = Velocity of water in pipe.

Substituting equation (i) into equation (ii), we get

$$h_f = \frac{4 f l}{d \times 2g} \times \left[\frac{A}{a} \omega r \sin \theta\right]^2 \qquad ...(20.17)$$

The variation of h_f with θ is parabolic. The loss of head due to friction in suction and delivery pipes is obtained from equation (20.17) by using subscripts *'s'* for suction pipe and *'d'* for delivery pipe as

$$h_{fs} = \frac{4 f l_s}{d_s \times 2g} \times \left[\frac{A}{a_s} \omega r \sin \theta\right]^2 \qquad ...(20.18)$$

$$h_{fd} = \frac{4 f l_d}{d_d \times 2g} \times \left[\frac{A}{a_d} \omega r \sin \theta\right]^2 \qquad ... (20.19)$$

The loss of head due to friction in pipes given by equation (20.17) varies with θ as :

1. When $\theta = 0°$, $\sin \theta = 0$ $\therefore$ $h_f = \frac{4 f l}{d \times 2g} \times 0 = 0$

2. When $\theta = 90°$, $\sin 90° = 1$ $\therefore$ $h_f = \frac{4 f l}{d \times 2g} \times \left[\frac{A}{a} \omega r\right]^2$

3. When $\theta = 180°$, $\sin 180° = 0$ $\therefore$ $h_f = 0$

$\therefore$ Maximum value of loss of head due to friction ;

$$(h_f)_{max} = \frac{4 f l}{d \times 2g} \times \left[\frac{A}{a} \omega r\right]^2 \qquad ...(20.20)$$

Problem 20.3 *The cylinder bore diameter of a single-acting reciprocating pump is 150 mm and its stroke is 300 mm. The pump runs at 50 r.p.m. and lifts water through a height of 25 m. The delivery pipe is 22 m long and 100 mm in diameter. Find the theoretical discharge and the theoretical power required to run the pump. If the actual discharge is 4.2 litres/s, find the percentage slip. Also determine the acceleration head at the beginning and middle of the delivery stroke.*

Solution. Given :

Dia. of cylinder, $D = 150 \text{ mm} = 0.15 \text{ m}$

$\therefore$ Area, $A = \left(\frac{\pi}{4}\right) \times 0.15^2 = 0.01767 \text{ m}^2$

Stroke, $L = 300 \text{ mm} = 0.3 \text{ m}$

Speed of pump, $N = 50$ r.p.m.

Total height through which water is lifted,

$H = 25 \text{ m}$

Length of delivery pipe, $l_d = 22 \text{ m}$

Diameter of delivery pipe, $d_d = 100 \text{ mm} = 0.1 \text{ m}$

Actual discharge, $Q_{act} = 4.2 \text{ litres/s} = \frac{4.2}{1000} \text{ m}^3/\text{s} = 0.0042 \text{ m}^3/\text{s}.$

(*i*) *Theoretical discharge* (Q_{th})

Theoretical discharge for a single-acting reciprocating pump is given by equation (20.1), as

$$Q_{th} = \frac{A \times L \times N}{60} = \frac{0.01767 \times 0.3 \times 50}{60} = 0.0044175 \text{ m}^3/\text{s}$$

$$= 0.0044175 \times 1000 \text{ litres/s} = \mathbf{4.4175 \text{ litres/s. Ans.}}$$

(*ii*) *Theoretical power* (P_t)

Theoretical power is given by, $P_t = \frac{(\text{Theoretical weight of water lifted/s}) \times \text{Total height}}{1000}$

$$= \frac{\rho \times g \times Q_{th} \times H}{1000}$$

$$= \frac{1000 \times 9.81 \times 0.0044175 \times 25}{1000} \qquad \left(\because Q_{th} = 0.0044175 \text{ m}^3/\text{s}\right)$$

$$= \mathbf{1.0833 \text{ kW. Ans.}}$$

(*iii*) *The percentage slip*

The percentage slip is given by,

$$\% \text{ slip} = \left(\frac{Q_{th} - Q_{act}}{Q_{th}}\right) \times 100 = \left(\frac{4.4175 - 4.2}{4.4175}\right) \times 100 = \mathbf{4.92\%. \text{ Ans.}}$$

(*iv*) *Acceleration head at the beginning of delivery stroke.*

The acceleration head in the delivery pipe is given by equation (20.15) as :

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d} \omega^2 r \times \cos\theta$$

where a_d = Area of delivery pipe $= \frac{\pi}{4} \times (0.1)^2 = 0.007854$

$$\omega = \text{Angular speed} = \frac{2\pi N}{60} = \frac{2\pi \times 50}{60} = 5.236$$

$$r = \text{Crank radius} = \frac{L}{2} = \frac{0.3}{2} = 0.15 \text{ m}$$

$$\therefore \quad h_{ad} = \frac{22}{9.81} \times \frac{0.01767}{0.007854} \times 5.236^2 \times 0.15 \times \cos\theta = 20.75 \times \cos\theta$$

At the beginning of delivery stroke, $\theta = 0°$ and hence $\cos\theta = 1$

$$\therefore \quad h_{ad} = \mathbf{20.75\ m.\ Ans.}$$

(*v*) *Acceleration head at the middle of delivery stroke.*

At the middle of delivery stroke, $\theta = 90°$ and hence $\cos\theta = 0$.

$$\therefore \quad h_{ad} = 20.75 \times 0 = \mathbf{0.\ Ans.}$$

▶ 20.8 INDICATOR DIAGRAM

The indicator diagram for a reciprocating pump is defined as the graph between the pressure head in the cylinder and the distance travelled by piston from inner dead centre for one complete revolution of the crank. As the maximum distance travelled by the piston is equal to the stroke length and hence the indicator diagram is a graph between pressure head and stroke length of the piston for one complete revolution. The pressure head is taken as ordinate and stroke length as abscissa.

20.8.1 Ideal Indicator Diagram. The graph between pressure head in the cylinder and stroke length of the piston for one complete revolution of the crank under ideal conditions is known as ideal indicator diagram. Fig. 20.4 shows the ideal indicator diagram, in which line *EF* represents the atmospheric pressure head equal to 10.3 m of water.

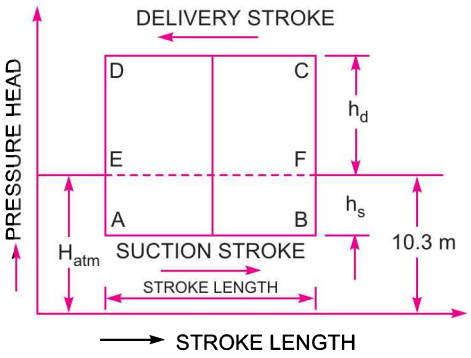


Fig. 20.4 *Ideal indicator diagram.*

Let H_{atm} = Atmospheric pressure head
= 10.3 m of water,
L = Length of the stroke,
h_s = Suction head, and
h_d = Delivery head.

During suction stroke, the pressure head in the cylinder is constant and equal to suction head (h_s), which is below the atmospheric pressure head (H_{atm}) by a height of h_s. The pressure head during suction stroke is represented by a horizontal line *AB* which is below the line *EF* by a height of 'h_s'.

During delivery stroke, the pressure head in the cylinder is constant and equal to delivery head (h_d), which is above the atmospheric head by a height of (h_d). Thus, the pressure head during delivery stroke is represented by a horizontal line *CD* which is above the line *EF* by a height of h_d. Thus, for one complete revolution of the crank, the pressure head in the cylinder is represented by the diagram *A-B-C-D-A*. This diagram is known as ideal indicator diagram.

Now from equation (20.3), we know that the work done by the pump per second

$$= \frac{\rho \times g \times ALN}{60} \times (h_s + h_d)$$

$$= K \times L(h_s + h_d) \qquad \left(\text{where } K = \frac{\rho g AN}{60} = \text{Constant}\right)$$

$$\propto L \times (h_s + h_d) \qquad \ldots(i)$$

But from Fig. 20.4, area of indicator diagram

$$= AB \times BC = AB \times (BF + FC) = L \times (h_s + h_d).$$

Substituting this value in equation (i), we get

Work done by pump $\propto$ Area of indicator diagram. ...(20.21)

20.8.2 Effect of Acceleration in Suction and Delivery Pipes on Indicator Diagram. The pressure head due to acceleration in the suction pipe is given by equation (20.14) as

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \omega^2 r \cos\theta$$

When $\theta = 0°$, $\cos\theta = 1$, and $h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \omega^2 r$

When $\theta = 90°$, $\cos\theta = 0$, and $h_{as} = 0$

When $\theta = 180°$, $\cos\theta = -1$, and $h_{as} = -\frac{l_s}{g} \times \frac{A}{a_s} \omega^2 r$.

Thus, the pressure head inside the cylinder during suction stroke will not be equal to 'h_s', as was the case for ideal indicator diagram, but it will be equal to the sum of 'h_s' and 'h_{as}'. At the beginning of suction stroke $\theta = 0°$, 'h_{as}' is +ve and hence the pressure head in the cylinder will be ($h_s + h_{as}$) below the atmospheric pressure head. At the middle of suction stroke $\theta = 90°$ and $h_{as} = 0$ and hence pressure head in the cylinder will be h_s below the atmospheric pressure head. At the end of suction stroke, $\theta = 180°$ and h_{as} is –ve and hence the pressure head in the cylinder will be ($h_s - h_{as}$) below the atmospheric pressure head. For suction stroke, the indicator diagram will be shown by $A'GB'$. Also the area of $A'AG$ = Area of BGB'.

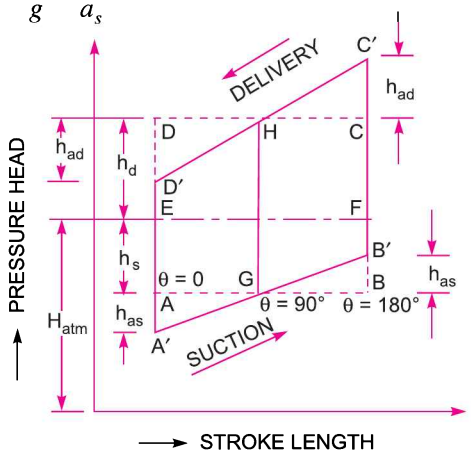


Fig. 20.5 *Effect of acceleration on indicator diagram.*

Similarly, the indicator diagram for the delivery stroke can be drawn. At the beginning of delivery stroke, h_{ad} is +ve and hence the pressure head in the cylinder will be ($h_d + h_{ad}$) above the atmospheric pressure head. At the middle of the delivery stroke, $h_{ad} = 0$ and hence pressure head in the cylinder is equal to h_d above the atmospheric pressure head. At the end of the delivery stroke, h_{ad} is –ve and hence pressure in the cylinder will be ($h_d - h_{ad}$) above the atmospheric pressure head. And thus the indicator diagram for delivery stroke is represented by the line $C'HD'$. Also, the area of $CC'H$ = Area of $DD'H$.

From Fig. 20.5, it is now clear that due to acceleration in suction and delivery pipe, the indicator diagram has changed from $ABCD$ to $A'B'C'D'$. But the area of indicator diagram $ABCD$ = Area $A'B'C'D'$. Now from equation (20.21), work done by pump is proportional to the area of indicator diagram. Hence the work done by the pump on the water remains same.

Problem 20.4 *The length and diameter of a suction pipe of a single-acting reciprocating pump are 5 m and 10 cm respectively. The pump has a plunger of diameter 15 cm and a stroke length of 35 cm. The centre of the pump is 3 m above the water surface in the pump. The atmospheric pressure head is 10.3 m of water and pump is running at 35 r.p.m. Determine :*

(*i*) *Pressure head due to acceleration at the beginning of the suction stroke,*
(*ii*) *Maximum pressure head due to acceleration, and*
(*iii*) *Pressure head in the cylinder at the beginning and at the end of the stroke.*

Solution. Given :

Length of suction pipe, $l_s = 5$ m

Diameter of suction pipe, $d_s = 10$ cm $= 0.1$ m

$\therefore$ Area, $a_s = \frac{\pi}{4} d_s^2 = \frac{\pi}{4} \times 0.1^2 = .007854 \text{ m}^2$

Diameter of plunger, $D = 15$ cm $= 0.15$ m

$\therefore$ Area of plunger, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} \times .15^2 = .01767 \text{ m}^2$

Stroke length, $L = 35$ cm $= 0.35$ m

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{0.35}{2} = 0.175$ m

Suction head, $h_s = 3$ m

Atmospheric pressure head, $H_{atm} = 10.3$ m of water

Speed of pump, $N = 35$ r.p.m.

Angular speed of the crank is given by,

$$\omega = \frac{2\pi N}{60} = \frac{2\pi \times 35}{60} = 3.665 \text{ rad/s.}$$

(*i*) The pressure head due to acceleration in the suction pipe is given by equation (20.14) as

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r \cos\theta$$

At the beginning of stroke, $\theta = 0^o$ and hence $\cos\theta = 1$

$\therefore$ $$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r = \frac{5}{9.81} \times \frac{.01767}{.007854} \times 3.665^2 \times .175 = \mathbf{2.695 \text{ m. Ans.}}$$

(*ii*) Maximum pressure head due to acceleration in suction pipe is given by equation (20.16), as

$$(h_{as})_{\max} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r = \mathbf{2.695 \text{ m. Ans.}}$$

(*iii*) Pressure head in the cylinder at the beginning of the suction stroke (Refer to Fig. 20.5)

$= h_s + h_{as} = 3.0 + 2.695 = 5.695.$

This pressure head in the cylinder is below the atmospheric pressure head.

$\therefore$ Absolute pressure head in the cylinder at the beginning of suction stroke

= Atmospheric pressure head – 5.695

= 10.3 – 5.695 = **4.605 m of water (abs.) Ans.**

Similarly, pressure head in the cylinder at the end of suction stroke

$= h_s - h_{as} = 3.0 - 2.695 = 0.305$ m below atmospheric pressure head

= 10.3 – 0.305 = **9.995 m of water (abs.) Ans.**

Problem 20.5 *If in Problem 20.4, the length and diameter of the delivery pipe are 30 m and 10 cm respectively and water is delivered by the pump to a tank which is 20 m above the centre of the pump, determine :*

(*i*) *Pressure head due to acceleration at the beginning of delivery stroke,*

(*ii*) *Pressure head in the cylinder at the beginning of the delivery stroke, and*

(*iii*) *Pressure head in the cylinder at the end of the delivery stroke.*

Solution. Given :

Length of delivery pipe, $l_d = 30$ m

Diameter of delivery pipe, $d_d = 10$ cm $= 0.1$ m

$\therefore$ Area of delivery pipe, $a_d = \frac{\pi}{4} d_d^2 = \frac{\pi}{4}(.1)^2 = .007854 \text{ m}^2$

Diameter of plunger, $D = 15$ cm $= 0.15$ m

$\therefore$ Area of plunger, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4}(.15)^2 = .01767 \text{ m}^2$

Stroke length, $L = 35$ cm $= 0.35$ m

Crank radius, $r = \frac{L}{2} = \frac{0.35}{2} = 0.175$ m

Delivery head, $h_d = 20$ m

Speed of pump, $N = 35$ r.p.m.

Angular speed, $\omega = \frac{2\pi N}{60} = \frac{2\pi \times 35}{60} = 3.665$ rad/s.

(*i*) Using equation (20.15), we get the pressure head due to acceleration in delivery pipe as

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d} \omega^2 r \cos\theta$$

At the beginning of delivery stroke, $\theta = 0^\circ$ and hence $\cos\theta = 1$.

$\therefore$ Pressure head due to acceleration at the beginning of delivery stroke becomes as

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d} \omega^2 r$$

$$= \frac{30}{9.81} \times \frac{.01767}{.007854} \times (3.665)^2 \times 0.175 = \mathbf{16.17 \text{ m. Ans.}}$$

(*ii*) From Fig. 20.5, the pressure head in the cylinder at the beginning of the delivery stroke

$= FC' = FC + CC' = (h_d + h_{ad})$ m of water above atmospheric head

$= 20 + 16.17 = 36.17$ m of water above atms.

$= 36.17 +$ Atmospheric pressure head

$= 36.17 + 10.3 =$ **46.47 m (abs.) Ans.**

(*iii*) The pressure head in the cylinder at the end of delivery stroke

$= ED'$ above atmospheric pressure head

$= (ED - DD') = (h_d - h_{ad})$

$= 20 - 16.17 = 3.83$ m of water above atms.

$= 3.83 + 10.3 =$ **14.13 m (abs.) Ans.**

Problem 20.6 *A single-acting reciprocating pump has piston diameter 12.5 cm and stroke length 30 cm. The centre of the pump is 4 m above the water level in the sump. The diameter and length of suction pipe are 7.5 cm and 7 m respectively. The separation occurs if the absolute pressure head in the cylinder during suction stroke falls below 2.5 m of water. Calculate the maximum speed at which the pump can run without separation. Take atmospheric pressure head = 10.3 m of water.*

Solution. Given :

Diameter of piston, $D = 12.5 \text{ cm} = 0.125 \text{ m}$

$\therefore$ Area, $A = \frac{\pi}{4}(.125)^2 = .01227 \text{ m}^2$

Stroke length, $L = 30 \text{ cm} = 0.30 \text{ m}$

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{0.30}{2} = 0.15 \text{ m}$

Suction head, $h_s = 4.0 \text{ m}$

Diameter of suction pipe, $d_s = 7.5 \text{ cm} = 0.075 \text{ m}$

$\therefore$ Area of suction pipe, $a_s = \frac{\pi}{4}(.075)^2 = .004418 \text{ m}^2$

Length of suction pipe, $l_s = 7.0 \text{ m}$

Separation pressure head, $h_{sep} = 2.5 \text{ m (absolute)}$

Atmospheric pressure head, $H_{atm} = 10.3 \text{ m}$

From the indicator diagram, drawn in Fig. 20.5, it is clear that the absolute pressure head during suction stroke is minimum at the beginning of the stroke. Thus, the separation can take place at the beginning of the stroke only. In that case the pressure head in the cylinder at the beginning of stroke becomes = h_{sep}.

But pressure head in the cylinder at the beginning of suction stroke

$= (h_s + h_{as})$ m below atmospheric pressure head

$=$ Atmospheric pressure head $- (h_s + h_{as})$ m absolute

$= H_{atm} - (h_s + h_{as})$ m (abs.)

$= 10.3 - (4.0 + h_{as})$

$\therefore$ $h_{sep} = 10.3 - (4.0 + h_{as})$

$2.5 = 10.3 - 4.0 - h_{as}$

or $h_{as} = 10.3 - 4.0 - 2.5 = 3.80 \text{ m}.$ ...(*i*)

But from equation (20.14), h_{as} at the beginning of suction stroke is given by the relation

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \omega^2 r \qquad (\because \theta = 0^\circ, \because \cos\theta = 1) \text{ ...(}ii\text{)}$$

Equating equations (*i*) and (*ii*), we get

$$3.80 = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r = \frac{7.0}{9.81} \times \frac{.01227}{.004418} \times \omega^2 \times .15$$

$\therefore$ $$\omega^2 = \frac{3.80 \times 9.81 \times .004418}{7.0 \times .01227 \times .15} = 12.783$$

or $\omega = \sqrt{12.783} = 3.575 \text{ radian/s}.$

But $\omega = \frac{2\pi N}{60}$

$$\therefore \quad N = \frac{60 \times \omega}{2\pi} = \frac{60 \times 3.575}{2\pi} = \textbf{34.14 r.p.m. Ans.}$$

Thus, the maximum speed at which the pump can run without separation is 34.14 r.p.m.

Problem 20.7 *The diameter and stroke length of a single-acting reciprocating pump are 100 mm and 300 mm respectively. The water is lifted to a height of 20 m above the centre of the pump. Find the maximum speed at which the pump may be run so that no separation occurs during the delivery stroke if the diameter and length of delivery pipe are 50 mm and 25 m respectively. Separation occurs if the absolute pressure head in the cylinder during delivery stroke falls below 2.50 m of water.*

Take atmospheric pressure head = 10.3 m of water.

Solution. Given :

Diameter of pump, $D = 100$ mm $= 0.1$ m

Stroke length , $L = 300$ mm $= 0.30$ m

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{0.30}{2} = 0.15$ m

Delivery head, $h_d = 20$ m

Diameter of delivery pipe, $d_d = 50$ mm $= 0.05$ m

Length of delivery pipe, $l_d = 25$ m

Separation pressure head, $h_{sep} = 2.5$ m (abs.)

Atmospheric pressure head, $H_{atm} = 10.3$ m of water.

Fig. 20.6 show the indicator diagram for delivery stroke only. The absolute pressure head during delivery stroke is minimum at the end of the stroke only. It means, if separation is to take place, it will occur only at the end of the delivery stroke where pressure head will be equal to separation pressure head (h_{sep}). The absolute pressure head at the end of delivery stroke from Fig. 20.6 is equal to $D'M$.

where $D'M = DM - DD'$

$= (DE + EM) - DD' \quad (\because DM = DE + EM)$

$= (h_d + H_{atm}) - h_{ad}$

$\therefore \quad h_{sep} = (h_d + H_{atm}) - h_{ad}$

or $\quad 2.5 = (20 + 10.3) - h_{ad}$

$\therefore \quad h_{ad} = (20 + 10.3) - 2.5 = 27.8$ m

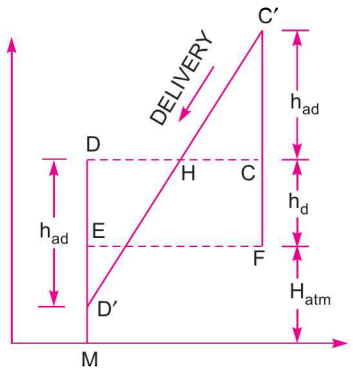


Fig. 20.6

But the acceleration head (h_{ad}) at the end of delivery stroke is given by

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d} \times \omega^2 r$$

$$27.8 = \frac{25}{9.81} \times \frac{\frac{\pi}{4}D^2}{\frac{\pi}{4}d_d^2} \times \omega^2 \times 0.15 = \frac{25}{9.81} \times \frac{D^2}{d_d^2} \times \omega^2 \times .15$$

$$= \frac{25}{9.81} \times \left(\frac{0.1}{.05}\right)^2 \times \omega^2 \times .15 = 1.529\omega^2$$

$$\therefore \quad \omega = \sqrt{\frac{27.8}{1.529}} = 4.264 \text{ radians/s.}$$

But we know, $\omega = \dfrac{2\pi N}{60}$

$\therefore$ $H = \dfrac{60 \times \omega}{2\pi} = \dfrac{60 \times 4.264}{2\pi} =$ **40.72 r.p.m. Ans.**

Problem 20.8 *A single-acting reciprocating pump raises water to a height of 20 m through a delivery pipe 35 m long and 140 mm in diameter. The bore and stroke of piston are 250 mm and 400 mm respectively. Cavitation occurs at 2.5 m of water absolute. Find the speed at which the pump can run without separation on delivery side if the pipe rises first vertically and then runs horizontally. Will there be any change in the maximum speed if the pipe first runs horizontally and then rises vertically.*

Solution. Given :

Delivery head, $h_d = 20$ m

Length of delivery pipe, $l_d = 35$ m

Dia. of delivery pipe, $d_d = 140$ mm $= 0.14$ m

Dia. of piston, $D = 250$ mm $= 0.25$ m

Stroke length, $L = 400$ mm $= 0.40$ m

$\therefore$ Crank radius, $r = \dfrac{L}{2} = \dfrac{0.40}{2} = 0.20$ m

Separation pressure head, $h_{sep} = 2.5$ m (abs.)

Atmospheric pressure head, $H_{atm} = 10.3$ m

The separation on delivery side can occur only at the end of delivery stroke as the pressure head during delivery stroke is minimum at the end of delivery stroke only. The acceleration head (h_{ad}) at the end of delivery stroke is given by,

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d} \times \omega^2 r = \frac{35}{9.81} \times \frac{\frac{\pi}{4} D^2}{\frac{\pi}{4} d_d^2} \times \omega^2 \times 0.20$$

$$= \frac{35}{9.81} \times \frac{0.25^2}{0.14^2} \times \omega^2 \times 0.20.$$

1st Case. The pipe rises first vertically and then horizontally as shown in Fig. 20.6 (*a*). In this case, the possibility of separation is at the point *C* at the end of the delivery stroke.

The pressure head at the end of delivery stroke at *B* will be equal to atmospheric pressure head plus delivery head minus acceleration head.

$\therefore$ Pressure head at $B = H_{atm} + h_d - h_{ad}$

The pressure head at *C* = Pressure head at $B - h_d$

$$= (H_{atm} + h_d - h_{ad}) - h_d = H_{atm} - h_{ad}$$

Now if separation is to take place at *C*, then the pressure head at *C* is 2.5 m.

Equating the two pressure heads at *C*, we get

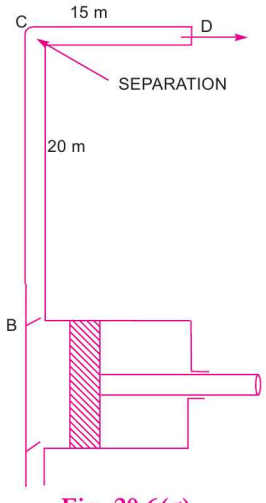


Fig. 20.6(*a*)

$$2.5 = H_{atm} - h_{ad}$$

or $$h_{ad} = H_{atm} - 2.5 = 10.3 - 2.5 = 7.8 \text{ m}$$

or $$\frac{35}{9.81} \times \frac{0.25^2}{0.14^2} \times \omega^2 \times 0.20 = 7.8$$

or $$\omega = \sqrt{\frac{9.81 \times 0.14^2 \times 7.8}{35 \times 0.25^2 \times 0.20}} = 1.85 \text{ rad/s}$$

$\therefore$ $$N = \frac{60 \times \omega}{2\pi} = \frac{60 \times 1.85}{2\pi} = \textbf{17.68 r.p.m. Ans.}$$

2nd Case. The pipe first runs horizontally and then rises vertically as shown in Fig. 20.6 (*b*).

In this case, the possibility of separation is at the point *C* at the end of delivery stroke. But the pressure head at *C* is same as pressure head at *B* and *C* are in the horizontal plane. Hence, at the end of delivery stroke, the pressure head at *B* is equal to $H_{atm} + h_d - h_{ad}$.

For this condition

$$h_{sep} = H_{atm} + h_d - h_{ad}$$

or $$2.5 = 10.3 + 20 - h_{ad}$$

or $$h_{ad} = 10.3 + 20 - 2.5 = 27.8$$

or $$\frac{35}{9.81} \times \frac{0.25^2}{0.14^2} \times \omega^2 \times 0.20 = 27.8$$

or $$\omega = \sqrt{\frac{9.81 \times 0.14^2 \times 27.8}{35 \times 0.25^2 \times 0.20}} = 3.495$$

$\therefore$ $$N = \frac{60 \times \omega}{2\pi} = \frac{60 \times 3.495}{2\pi}$$

$$= \textbf{33.37 r.p.m. Ans.}$$

$\therefore$ Change in maximum speed

$$= 33.36 - 17.68 = \textbf{15.69 r.p.m. Ans.}$$

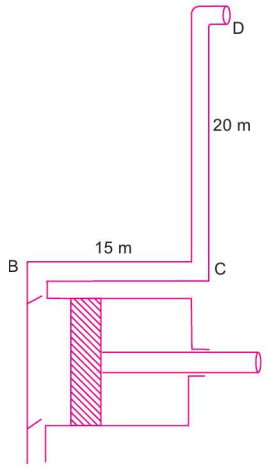


Fig. 20.6(*b*)

Problem 20.9 *A single-acting reciprocating pump has a plunger of 10 cm diameter and a stroke of length 200 mm. The centre of the pump is 4 m above the water level in the sump and 14 m below the level of water in a tank to which water is delivered by the pump. The diameter and length of suction pipe are 40 mm and 6 m while of the delivery pipe are 30 mm and 18 m respectively. Determine the maximum speed at which the pump may be run without separation, if separation occurs at 7.848 N/cm^2 below the atmospheric pressure. Take atmospheric pressure head = 10.3 m of water.*

Solution. Given :

Diameter of plunger, $D = 100 \text{ mm} = 0.10 \text{ m}$

Stroke length, $L = 200 \text{ mm} = 0.20 \text{ m}$

$\therefore$ Crank radius, $$r = \frac{L}{2} = \frac{0.20}{2} = 0.10 \text{ m}$$

Suction head, $h_s = 4 \text{ m}$

Delivery head, $h_d = 14 \text{ m}$

Dia. of suction pipe, $d_s = 40 \text{ mm} = 0.04 \text{ m}$

Length of suction pipe, $l_s = 6 \text{ m}$

Dia. of delivery pipe, $d_d = 30 \text{ mm} = .03 \text{ m}$

Length of delivery pipe, $l_d = 18 \text{ m}$

Separation pressure, $p_{sep} = \dfrac{7.848\,N}{\text{cm}^2} = \dfrac{7.848 \times 10^4}{\text{m}^2}$

$\therefore$ Separation pressure head, $h_{sep} = \dfrac{p_{sep}}{\rho g} = \dfrac{7.848 \times 10^4}{1000 \times 9.81} = 8.0$ m below atmosphere

$= (H_{atm} - 8.0)$ absolute $= (10.3 - 8.0) = 2.3$ m (abs.)

where H_{atm} = Atmospheric pressure head = 10.3 m.

(*i*) **Speed of pump without separation during suction stroke.** During suction stroke, possibility of separation is only at the beginning of the stroke. The pressure head in the cylinder at the beginning of suction stroke

$= (h_s + h_{as})$ m below atmospheric pressure head

$= 10.3 - (h_s + h_{as})$ m (abs.)

$\therefore \quad h_{sep} = 10.3 - (h_s + h_{as})$

or $\quad 2.3 = 10.3 - (h_s + h_{as}) = 10.3 - 4 - h_{as}$

$\therefore \quad h_{as} = 10.3 - 4 - 2.3 = 4$ m. ...(*i*)

But 'h_{as}' at the beginning of suction stroke is given by,

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r = \frac{6}{9.81} \times \frac{\frac{\pi}{4} D^2}{\frac{\pi}{4} d_s^2} \times \omega^2 \times .10$$

$$= \frac{6}{9.81} \times \left(\frac{0.1}{.04}\right)^2 \times \omega^2 \times .1 = 0.3822\,\omega^2 \qquad ...(ii)$$

Equating the values of h_{as} given by equations (*i*) and (*ii*),

$$4 = 0.3822\,\omega^2$$

$\therefore \quad \omega = \sqrt{\dfrac{4}{.382}} = 3.235$ rad/s

But ω is also $= \dfrac{2\pi N}{60}$

$\therefore \quad N = \dfrac{60 \times \omega}{2\pi} = \dfrac{60 \times 3.235}{2\pi} = 30.89$ r.p.m.

$\therefore$ Maximum speed of the pump without separation during suction stroke only is 30.89 r.p.m.

(*ii*) **Speed of pump without separation during delivery stroke.** During delivery stroke, the possibility of separation is only at the end of the delivery stroke. The pressure head in the cylinder at the end of the delivery stroke from Fig. 20.6

$$= (H_{atm} + h_d) - h_{ad} \text{ m (abs.)} = (10.3 + 14) - h_{ad.}$$

If separation is to be avoided this pressure should be equal to separation pressure head.

$$\therefore \quad h_{sep} = (10.3 + 14) - h_{ad} \text{ or } 2.30 = (10.3 + 14) - h_{ad}$$

$$\therefore \quad h_{ad} = 10.3 + 14.0 - 2.30 = 22.0 \text{ m}$$

But 'h_{ad}' is given by the relation

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d} \times \omega^2 \times r$$

$$\therefore \quad 22.0 = \frac{18.0}{9.81} \times \frac{\frac{\pi}{4} D^2}{\frac{\pi}{4} d_d^2} \times \omega^2 \times r = \frac{18.0}{9.81} \times \frac{D^2}{d_d^2} \times \omega^2 \times r$$

$$= \frac{18.0}{9.81} \times \left(\frac{.1}{.03}\right)^2 \times \omega^2 \times 0.10 = 2.04\ \omega^2$$

$$\therefore \quad \omega = \sqrt{\frac{22.0}{2.04}} = 3.284 \text{ rad/s.}$$

But $$\omega = \frac{2\pi N}{60}$$

$$\therefore \quad N = \frac{60 \times \omega}{2\pi} = \frac{60 \times 3.284}{2 \times \pi} = 31.36 \text{ r.p.m.}$$

$\therefore$ Maximum speed of the pump without separation during delivery stroke is 31.36 r.p.m.

Thus the maximum speed of the pump without separation during suction and delivery stroke is the minimum of these two speeds, *i.e.*, minimum of 30.89 and 31.36 r.p.m.

$\therefore$ Maximum speed = **30.89 r.p.m. Ans.**

20.8.3 Effect of Friction in Suction and Delivery Pipes on Indicator Diagram. The loss of head due to friction in suction and delivery pipes is given by equations (20.18) and (20.19) as

$$h_{fs} = \frac{4 f l_s}{d_s \times 2g} \times \left(\frac{A}{a_s} \omega r \sin\theta\right)^2 \text{ and } h_{fd} = \frac{4 f l_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \omega r \sin\theta\right)^2$$

It is clear from the above equations that the variation of h_{fs} or h_{fd} is parabolic with θ.

During the suction or delivery stroke, the pressure head inside the cylinder will change as given below :

(*i*) At the beginning of the suction or delivery stroke, $\theta = 0°$ and hence $\sin\theta = 0$. This means h_{fs} and $h_{fd} = 0$.

(*ii*) At the middle of the suction or delivery stroke, $\theta = 90°$ and hence $\sin\theta = 1$. This means

$$h_{fs} = \frac{4 f l_s}{d_s \times 2g} \times \left(\frac{A}{a_s} \omega r\right)^2 \text{ and } h_{fd} = \frac{4 f l_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \omega r\right)^2.$$

(*iii*) At the end of the suction or delivery stroke, $\theta = 180°$ and hence $\sin\theta = 0°$. This means h_{fs} and $h_{fd} = 0$.

As the variation of h_{fs} or h_{fd} with θ is parabolic in nature, the indicator diagram during suction and delivery stroke with friction in suction pipe and delivery pipe will be as shown in Fig. 20.7.

The area of the parabolas *AGB* and *CHD* represents the work done against friction in suction and delivery pipes.

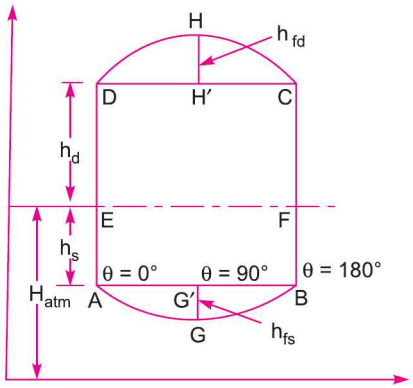


Fig. 20.7 *Effect of friction on indicator diagram.*

Now $$\text{area } AGB = AB \times \frac{2}{3} GG'$$

$$= AB \times \frac{2}{3} h_{fs} = L \times \frac{2}{3} h_{fs}$$

where $$h_{fs} = \frac{4 \times f \times l_s}{d_s \times 2g} \times \left(\frac{A}{a_s}\omega r\right)^2$$

Similarly, $$\text{area } CHD = CD \times \frac{2}{3} \times HH' = CD \times \frac{2}{3} h_{fd}$$

$$= L \times \frac{2}{3} h_{fd} \qquad (\because \quad CD = L = \text{Length of stroke})$$

where $$h_{fd} = \frac{4 f l_d}{d_d \times 2g} \times \left(\frac{A}{a_d}\omega r\right)^2.$$

20.8.4 Effect of Acceleration and Friction in Suction and Delivery Pipes on Indicator Diagram. Fig. 20.8 shows the combined effect of acceleration and friction in suction and delivery pipes. The pressure head in the cylinder during suction and delivery strokes will change as given below :

(*i*) At the beginning of the suction stroke, $\theta = 0°$ and hence h_{as} from equation (20.14) is equal to $\frac{l_s}{g} \times \frac{A}{a_s}\omega^2 r$. But $h_{fs} = 0$. Thus, the pressure head in the cylinder will be $(h_s + h_{as})$ below the atmospheric pressure head.

(*ii*) At the middle of the suction stroke, $h_{as} = 0$ but $h_{fs} = \frac{4 \times f \times l_s}{d_s \times 2g} \times \left(\frac{A}{a_s}\omega r\right)^2$. Thus, the pressure head in the cylinder will be $(h_s + h_{fs})$ below the atmospheric pressure head.

(*iii*) At the end of the suction stroke, $h_{as} = -\frac{l_s}{g} \times \frac{A}{a_s}\omega^2 r$ but $h_{fs} = 0$. Thus, the pressure head in the cylinder will be $(h_s - h_{as})$ below the atmospheric pressure head.

(*iv*) At the beginning of the delivery stroke, $h_{ad} = -\frac{l_d}{g} \times \frac{A}{a_d} \times \omega^2 r$ but $h_{fd} = 0$. Thus, the pressure head in the cylinder will be $(h_d + h_{ad})$ above the atmospheric pressure head.

(*v*) In the middle of the delivery stroke, $h_{ad} = 0$ and $h_{fd} = \frac{4 f l_d}{d_d \times 2g} \times \left(\frac{A}{a_d}\omega r\right)^2$. Thus the pressure head in the cylinder will be $(h_d + h_{fd})$ above the atmospheric pressure head.

(*vi*) At the end of the delivery stroke, $h_{ad} = -\frac{l_d}{g} \times \frac{A}{a_d} \times \omega^2 r$ and $h_{fd} = 0$. Thus, the pressure head in the cylinder will be $(h_d - h_{ad})$ above the atmospheric pressure head.

Thus, the indicator diagram with acceleration and friction in suction and delivery pipes will become as shown in Fig. 20.8.

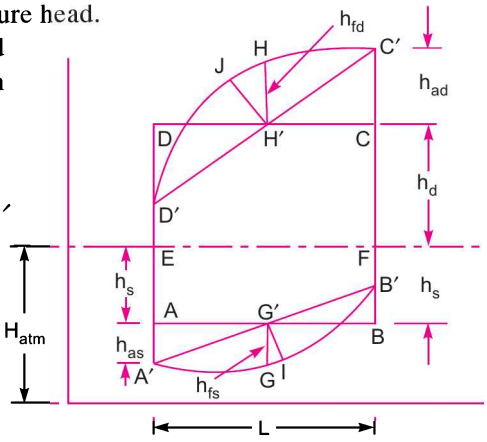


Fig. 20.8 *Effect of acceleration and friction on indicator diagram.*

Area of the indicator diagram $A'GB'\ C'HD'$

= Area of $A'G'B'C'H'D'$ + Area of parabola $A'GB'$ + Area of parabola $C'HD'$

But area of $A'G'B'C'H'D'$

= Area of $ABCD$

$= (h_s + h_d) \times L$

Area of parabola $A'GB'$

$$= A'B' \times \frac{2}{3} \times G'I = \frac{2}{3} \times (A'B' \times G'I)$$

$$= \frac{2}{3} \times (AB \times GG') = \frac{2}{3} \times L' h_{fs}$$

Similarly, area of parabola $C'HD'$

$$= C'D' \times \frac{2}{3} H'J = \frac{2}{3} (C'D' \times H'J)$$

$$= \frac{2}{3} \times (CD \times H'H) = \frac{2}{3}(L \times h_{fd}) = \frac{2}{3} L \times h_{fd}$$

$\therefore$ Area of indicator diagram

$$= (h_s + h_d) \times L + \frac{2}{3} \times L \times h_{fs} + \frac{2}{3} \times L \times h_{fd}$$

$$= \left(h_s + h_d + \frac{2}{3} h_{fs} + \frac{2}{3} h_{fd}\right) \times L$$

But from equation (20.21), we know that work done by pump is proportional to the area of the indicator diagram.

$\therefore$ Work done by pump per second $\propto \left(h_s + h_d + \frac{2}{3} h_{fs} + \frac{2}{3} h_{fd}\right) \times L$

$$= KL \left(h_s + h_d + \frac{2}{3} h_{fs} + \frac{2}{3} h_{fd}\right)$$

where K = a constant of proportionality

$$= \frac{\rho g A N}{60} \qquad \text{...for a single-acting}$$

$$= \frac{2\rho gAN}{60} \quad \text{...for a double-acting}$$

$\therefore$ Work done by pump per second for a single-acting

$$= \frac{\rho gALN}{60}\left(h_s + h_d + \frac{2}{3}h_{fs} + \frac{2}{3}h_{fd}\right) \quad \text{...(20.22)}$$

and for a double-acting $$= \frac{2\rho gALN}{60}\left(h_s + h_d + \frac{2}{3}h_{fs} + \frac{2}{3}h_{fd}\right) \quad \text{...(20.23)}$$

Problem 20.10 *A single-acting reciprocating pump has a stroke length of 15 cm. The suction pipe is 7 metre long and the ratio of the suction diameter to the plunger diameter is 3/4. The water level in the sump is 2.5 metres below the axis of the pump cylinder, and the pipe connecting the sump and pump cylinder is 7.5 cm diameter. If the crank is running at 75 r.p.m., determine the pressure head on the piston :*

(i) in the beginning of the suction stroke, (ii) in the end of the suction stroke, and (iii) in the middle of the suction stroke.

Take co-efficient of friction as 0.01.

Solution. Given :

Stroke length, $L = 15 \text{ cm} = 0.15 \text{ m}$

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{0.15}{2} = 0.075 \text{ m}$

Length of suction pipe, $l_s = 7.0 \text{ m}$

$$\frac{\text{Suction pipe diameter}}{\text{Plunger diameter}} = \frac{d_s}{D} = \frac{3}{4}$$

$\therefore$ $$\frac{\text{Area of suction pipe}}{\text{Area of plunger}} = \frac{a_s}{A} = \left(\frac{3}{4}\right)^2 = \frac{9}{16}$$

Suction head, $h_s = 2.5$

Diameter of suction pipe, $d_s = 7.5 \text{ cm} = 0.075 \text{ m}$

Crank speed, $N = 75 \text{ r.p.m.}$

$\therefore$ Angular speed, $\omega = \frac{2\pi N}{60} = \frac{2 \times \pi \times 75}{60} = 2.5\ \pi \text{ rad/s.}$

Friction co-efficient, $f = 0.01$

The pressure head due to acceleration in suction pipe is given by equation (20.14), as

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 \times r \cos\theta$$

$$= \frac{7.0}{9.81} \times \frac{16}{9} \times (2.5\ \pi)^2 \times 0.075 \cos\theta \qquad \left(\because \frac{A}{a_s} = \frac{16}{9}\right)$$

$$= 5.87 \cos\theta$$

The loss of head due to friction in suction pipe is given by equation (20.18)as

$$h_{fs} = \frac{4fl_s}{d_s \times 2g} \times \left(\frac{A}{a_s}\omega r \sin\theta\right)^2$$

$$= \frac{4 \times 0.01 \times 7.0}{0.075 \times 2 \times 9.81} \times \left(\frac{16}{9} \times 2.5\pi \times 0.075 \times \sin \theta \right)^2 = 0.208 \sin^2 \theta.$$

(*i*) *Pressure head on the piston in the beginning of suction stroke :*

(Refer to Fig. 20.8). At the beginning of suction stroke, $\theta = 0^o$ and pressure head

$= (h_s + h_{as})$ m below atmospheric pressure head

$= 2.5 + 5.87 =$ **8.37 m vacuum. Ans.**

(*ii*) *Pressure head on the piston at the end of suction stroke :*

At the end of the suction stroke, $\theta = 180^o$ and hence cos θ

$= -1$ and sin $\theta = 0$

The pressure head $= (h_s - h_{as})$ m below atmospheric pressure head

$= H_{atm} - (h_s - h_{as})$ m abs. $= H_{atm} - (2.5 - 5.87)$ m abs.

$= H_{atm} - (-3.37)$ m abs. $= H_{atm} + 3.37$ m abs. = **3.37 m (gauge). Ans.**

(*iii*) *Pressure head on the piston in the middle of suction stroke :*

In the middle of suction stroke, $\theta = 90°$ and hence cos $\theta = 0$ and sin $\theta = 1$. The pressure head

$= (h_s + h_{fs})$ m below atmospheric pressure head

= (2.5 + 0.208) m vacuum = **2.708 m vacuum. Ans.**

Problem 20.11 *The diameter and stroke length of a single-acting reciprocating pump are 12 cm and 20 cm respectively. The lengths of suction and delivery pipes are 8 m and 25 m respectively and their diameters are 7.5 cm. If the pump is running at 40 r.p.m. and suction and delivery heads are 4 m and 14 m respectively, find the pressure head in the cylinder :*

(*i*) *at the beginning of the suction and delivery stroke,*

(*ii*) *in the middle of suction and delivery stroke, and*

(*iii*) *at the end of the suction and delivery stroke.*

Take atmospheric pressure head = 10.30 metres of water and f = .009 for both pipes.

Solution. Given :

Diameter of cylinder, $D = 12$ cm $= 0.12$ m

Stroke length, $L = 20$ cm $= 0.20$ m

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{.20}{2} = 0.10$ m

Length of suction pipe, $l_s = 8$ m

Length of delivery pipe, $l_d = 25$ m

Dia. of suction pipe, $d_s = 7.5$ cm $= 0.075$ m

Dia. of delivery pipe, $d_d = 7.5$ cm $= 0.075$ m

Speed of pump, $N = 40$ r.p.m.

Suction head, $h_s = 4$ m

Delivery head, $h_d = 14$ m

Atmospheric pressure head, $H_{atm} = 10.3$ m of water

Co-efficient of friction, $f = .009$.

We know that angular velocity, $\omega = \frac{2\pi N}{60} = \frac{2\pi \times 40}{60} = 4.188$ rad/s.

Using equation (20.14), the pressure head due to acceleration in suction pipe is obtained as

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r \cos\theta = \frac{8}{9.81} \times \frac{\frac{\pi}{4}D^2}{\frac{\pi}{4}d_s^2} \times 4.188^2 \times 0.1 \times \cos\theta$$

$$= \frac{8}{9.81} \times \left(\frac{D}{d_s}\right)^2 \times 4.188^2 \times 0.1 \times \cos\theta$$

$$= \frac{8}{9.81} \times \left(\frac{.120}{.075}\right)^2 \times 4.188^2 \times 0.1 \times \cos\theta = 3.66 \times \cos\theta \text{ m.}$$

Similarly, the pressure head due to acceleration in delivery pipe is obtained from equation (20.15) as

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d} \times \omega^2 r \cos\theta$$

$$= \frac{25}{9.81} \times \frac{D^2}{d_d^2} \times 4.188^2 \times 0.1 \times \cos\theta$$

$$= \frac{25}{9.81} \times \left(\frac{.120}{.075}\right)^2 \times 4.188^2 \times 0.1 \times \cos\theta = 11.44 \times \cos\theta \text{ m.}$$

Using equation (20.18) for the loss of head due to friction in suction pipe,

$$h_{fs} = \frac{4fl_s}{d_s \times 2g} \times \left(\frac{A}{a_s}\omega r \sin\theta\right)^2$$

$$= \frac{4 \times .009 \times 8}{.075 \times 2 \times 9.81} \times \left(\frac{\frac{\pi}{4}D^2}{\frac{\pi}{4}d_s^2} \times 4.188 \times .1 \times \sin\theta\right)^2$$

$$= \frac{4 \times .009 \times 8}{.075 \times 2 \times 9.81} \times \left(\frac{.12^2}{.075^2} \times 4.188 \times .1\right)^2 \times \sin^2\theta = 0.225 \sin^2\theta.$$

Similarly, loss of head due to friction in delivery pipe is obtained from equation (20.19) as

$$h_{fd} = \frac{4 \times f \times l_d}{d_d \times 2g} \times \left[\frac{A}{a_d}\omega r \sin\theta\right]^2$$

$$= \frac{4 \times .009 \times 25}{0.075 \times 2 \times 9.81}\left[\frac{D^2}{d_d^2} \times 4.188 \times 0.1 \times \sin\theta\right]^2$$

$$= 0.6116\left[\frac{.12^2}{.075^2} \times 4.188 \times 0.1\right]^2 \times \sin^2\theta = 0.703 \sin^2\theta.$$

(*i*) **Pressure head in the cylinder at the beginning of suction and delivery strokes (Fig. 20.8).** At the beginning of suction and delivery strokes, $\theta = 0°$.

$\therefore$ $h_{as} = 3.66 \cos 0° = 3.66$ m

and $h_{ad} = 11.44 \cos 0° = 11.44$ m

Pressure head in the cylinder at the beginning of suction stroke

$= (h_s + h_{as})$ below atmospheric pressure head

$= H_{atm} - (h_s + h_{as})$ m (abs.)

$= 10.3 - (1 + 3.66) = 10.3 - 7.66 =$ **2.64 m (abs.) Ans.**

Pressure head in the cylinder at the beginning of delivery stroke

$= (h_d + h_{ad})$ above atmospheric pressure head

$= H_{atm} + (h_d + h_{ad})$ m (abs.)

$= 10.3 + (14 + 11.44) =$ **35.74 m (abs.) Ans.**

(*ii*) **Pressure head in the cylinder in the middle of suction and delivery strokes.** In the middle of suction and delivery strokes, $\theta = 90°$.

$\therefore$ $h_{as} = 0$, $h_{ad} = 0$, $h_{fs} = 0.225 \sin^2 90 = 0.225$ m and

$h_{fd} = 0.703 \sin^2 90 = 0.703$ m.

Pressure head in the cylinder at the beginning of suction stroke

$= (h_s + h_{fs})$ below the atmospheric pressure head

$= H_{atm} - (h_s + h_{fs})$ m (abs.)

$= 10.3 - (4 + 0.225) = 10.3 - 4.225 =$ **6.075 m (abs.) Ans.**

Pressure head in the cylinder at the beginning of delivery stroke

$= (h_d + h_{fd})$ above atmospheric pressure head

$= H_{atm} + (h_d + h_{fd})$ m (abs.)

$= 10.3 + (14 + 0.703) =$ **125.003 m (abs.) Ans.**

(*iii*) **Pressure head in the cylinder at the end of suction and delivery strokes.** At the end of suction and delivery strokes, $\theta = 180°$.

$\therefore$ $\cos \theta = -1$ and $\sin \theta = 0$

$\therefore$ $h_{as} = 3.66 \times (-1) = -3.66$ m

and $h_{ad} = 11.44 \times (-1) = -11.44$ m.

Pressure head in the cylinder at the end of suction stroke

$= (h_s - 3.66)$ below the atmospheric pressure head

$= H_{atm} - (h_s - 3.66)$ m (abs.) $= 10.3 - (4 - 3.66) =$ **9.96 m (abs.) Ans.**

Pressure head in the cylinder at the end of delivery stroke

$= (h_d - 11.44)$ above the atmospheric pressure head

$= H_{atm} + (h_d - 11.44)$ m (abs.)

$= 10.3 + (14 - 11.44) =$ **12.86 m (abs.) Ans.**

Problem 20.12 *For the single-acting reciprocating pump, given in Problem 20.11, find the power required to drive the pump, if water is flowing through the pump.*

Solution. Using equation (20.22) for the work done by the pump per second for a single-acting, we get

$$\text{Work done per sec} = \frac{\rho g A L N}{60}\left(h_s + h_d + \frac{2}{3}h_{fs} + \frac{2}{3}h_{fd}\right)$$

where $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} \times .12^2 = 0.01131 \text{ m}^2$

L = Stroke length = 0.20 m, N = Speed = 40 r.p.m.

ρ = Density of water = 1000 kg/m^3, h_s = 4 m, h_d = 14 m

h_{fs} = Maximum loss of head due to friction in suction pipe = 0.225 m

h_{fd} = Maximum loss of head due to friction in delivery pipe = 0.703 m

$$\therefore \quad \text{Work done per second} = \frac{1000 \times 9.81 \times 0.01131 \times 0.20 \times 40}{60}\left(4 + 14 + \frac{2}{3} \times .225 + \frac{2}{3} \times .703\right)$$

$$= 14.793 \times (4 + 14 + 0.15 + 0.468) = 275.42 \text{ Nm/s}$$

$\therefore$ Power required to drive the pump in kW

$$= \frac{\text{Work done per second}}{1000} = \frac{275.42}{1000} = 0.2754 \text{ kW}.$$

20.8.5 Maximum Speed of a Reciprocating Pump. Maximum speed of a reciprocating pump is determined from the fact that the pressure in the cylinder during suction and delivery stroke, should not fall below the vapour pressure of the liquid, flowing through suction and delivery pipe. If the pressure in the cylinder is below the vapour pressure, the dissolved gases will be liberated from the liquid and cavitation * will take place. Also the continuous flow of liquid will not exist which means separation of liquid will take place. The pressure at which separation takes place is known as separation pressure and the head corresponding to separation pressure is called separation pressure head. It is denoted by h_{sep}. For water, the limiting value of separation pressure head (h_{sep}) is 7.8 below atmospheric pressure head or 10.3 – 7.8 = 2.5 m abs. The separation may take place during the suction stroke or during delivery stroke. The maximum speed of the reciprocating pump during suction and delivery strokes is calculated as :

(a) **Maximum Speed during Suction Stroke.** From the indicator diagram, drawn in Fig. 20.8, it is clear that the absolute pressure head during suction stroke is minimum at the beginning of the stroke. Thus, the separation can take place at the beginning of the stroke only. In that case, the abs. pressure head in the cylinder at the beginning of the stroke will be equal to separation pressure head (h_{sep}).

$$\therefore \quad h_{sep} = H_{atm} - (h_s + h_{as}) \text{ (abs.)}$$

or
$$h_{as} = H_{atm} - h_s - h_{sep} \qquad ...(i)$$

Generally, the values of h_{sep} and h_s (suction head) are given and hence 'h_{as}', the pressure head due to acceleration at the beginning of suction stroke can be obtained. The value of 'h_{as}' is also given by equation (20.14) as

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r \qquad ...(ii)$$

Equating the two values of h_{as} given by equations (i) and (ii)

$$H_{atm} - h_s - h_{sep} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r \qquad ...(iii)$$

* Please refer to Art. 19.11 on page 980 for definition.

From equation (*iii*), the value of the ω (and hence speed *N*) can be obtained. This speed is the maximum speed of the reciprocating pump without separation during suction stroke.

(b) **Maximum Speed during Delivery Stroke.** During delivery stroke, the probability of separation is only at the end of the delivery stroke. The pressure head in the cylinder at the end of the delivery stroke from Fig. 20.8.

$$= (H_{atm} + h_d) - h_{ad} \text{ m (abs.)}$$

If separation is to be avoided, the above pressure head should be more than the separation pressure head (h_{sep}). In the limiting case

$$h_{sep} = (H_{atm} + h_d) - h_{ad} \text{ or } h_{ad} = (H_{atm} + h_d) - h_{sep}$$

But 'h_{ad}', the pressure head due to acceleration at the end of the delivery stroke is also given by

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d} \times \omega^2 \times r$$

Equating the two values of h_{ad}, we get

$$(H_{atm} + h_d) - h_{sep} = \frac{l_d}{g} \times \frac{A}{a_d} \omega^2 \times r \qquad ...(iv)$$

From the above equation (*iv*), the value of ω and hence speed *N*, can be calculated. This is the maximum speed of the reciprocating pump without separation during delivery stroke only.

The minimum of the two speeds given by above two cases (*a*) and (*b*) is the maximum speed of the reciprocating pump without separation during suction and delivery strokes.

Problem 20.13 *Find the maximum speed of a single-acting reciprocating pump to avoid separation, which occurs at 3.0 m of water (abs.) The pump has a cylinder of diameter 10 cm and a stroke length of 20 cm.The pump draws water from a sump and delivers to a tank. The water level in the sump is 3.5 m below the pump axis and in the tank the water level is 13 m above the pump axis. The diameter and length of the suction pipe are 4 cm and 5 m while of delivery pipe the diameter and length are 3 cm, 20 m respectively. Take atmospheric pressure head = 10.3 m of water.*

Solution. Given :

Separation pressure head, $h_{sep} = 3.0$ m of water (abs.)
Dia. of cylinder, $D = 10$ cm $= 0.10$ m
Stroke length, $L = 20$ cm $= 0.20$ m

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{0.20}{3} = 0.10$ m

Suction head, $h_s = 3.5$ m
Delivery head, $h_d = 13$ m
Dia. of suction pipe, $d_s = 4$ cm $= .04$ m
Length of suction pipe, $l_s = 5$ m
Dia. of delivery pipe, $d_d = 3$ cm $= .03$ m
Length of delivery pipe, $l_d = 20$ m
Atmos. pressure head, $H_{atm} = 10.3$ m.

(*a*) Maximum speed during suction stroke without separation is obtained from the relation, given by equation (*iii*),

$$H_{atm} - h_s - h_{sep} = \frac{l_s}{g} \times \frac{A}{a_s} \times \omega^2 r$$

or $$10.3 - 3.5 - 3.0 = \frac{5}{9.81} \times \frac{\frac{\pi}{4}D^2}{\frac{\pi}{4}d_s^2} \times \omega^2 \times 0.10 \qquad \left(\because\ A = \frac{\pi}{4}D^2,\ a_s = \frac{\pi}{4}d_s^2\right)$$

or $$3.8 = \frac{5}{9.81} \times \frac{.1 \times .1}{.04 \times .04} \times \omega^2 \times 0.1 = .3185\ \omega^2$$

$\therefore$ $$\omega = \sqrt{\frac{3.8}{.3185}} = 3.454 \text{ rad/s.}$$

But $$\omega = \frac{2\pi N}{60} \quad \therefore \quad \frac{2\pi N}{60} = 3.454$$

$\therefore$ $$N = \frac{60 \times 3.454}{2\pi} = 32.98 \text{ r.p.m.} \qquad ...(i)$$

(*b*) Maximum speed during delivery stroke without separation is obtained from equation (*iv*),

$$(H_{atm} + h_d) - h_{sep} = \frac{l_d}{g} \times \frac{A}{a_d} \times \omega^2 \times r$$

$$(10.3 + 13.0) - 3.0 = \frac{20}{9.81} \times \frac{D^2}{d_d^2} \times \omega^2 \times .1 = \frac{20}{9.81} \times \frac{0.1 \times .1}{.03 \times .03} \times \omega^2 \times .1$$

or $$20.3 = 2.265\ \omega^2$$

$\therefore$ $$\omega = \sqrt{\frac{20.3}{2.265}} = 2.994 \text{ rad/s.}$$

But $$\omega = \frac{2\pi N}{60} = 2.994$$

$\therefore$ $$N = \frac{60 \times 2.994}{2\pi} = 28.59 \text{ r.p.m.} \qquad ...(ii)$$

The minimum of the two speeds given by equations (*i*) and (*ii*) is the maximum speed of the pump, without separation.

$\therefore$ Maximum speed without separation = **28.59 r.p.m. Ans.**

▶ 20.9 AIR VESSELS

An air vessel is a closed chamber containing compressed air in the top portion and liquid (or water) at the bottom of the chamber. At the base of the chamber there is an opening through which the liquid (or water) may flow into the vessel or out from the vessel. When the liquid enters the air vessel, the air gets compressed further and when the liquid flows out the vessel, the air will expand in the chamber.

An air vessel is fitted to the suction pipe and to the delivery pipe at a point close to the cylinder of a single-acting reciprocating pump :

(*i*) to obtain a continuous supply of liquid at a uniform rate,

(*ii*) to save a considerable amount of work in overcoming the frictional resistance in the suction and delivery pipes, and

(*iii*) to run the pump at a high speed without separation.

Fig. 20.9 shows the single-acting reciprocating pump to which air vessels are fitted to the suction and delivery pipes. The air vessels act like an intermediate reservoir. During the first half of the suction stroke, the piston moves with acceleration, which means the velocity of water in the suction pipe is more than the mean velocity and hence the discharge of water entering the cylinder will be more than the mean discharge. This excess quantity of water will be supplied from the air vessel to the cylinder in such a way that the velocity in the suction pipe below the air vessel is equal to mean velocity of flow. During the second half of the suction stroke, the piston moves with retardation and hence velocity of flow in the suction pipe is less than the mean velocity of flow. Thus, the discharge entering the cylinder will be less than the mean discharge. The velocity of water in the suction pipe due to air vessel is equal to mean velocity of flow and discharge required in cylinder is less than the mean discharge. Thus, the excess water flowing in suction pipe will be stored into air vessel, which will be supplied during the first half of the next suction stroke.

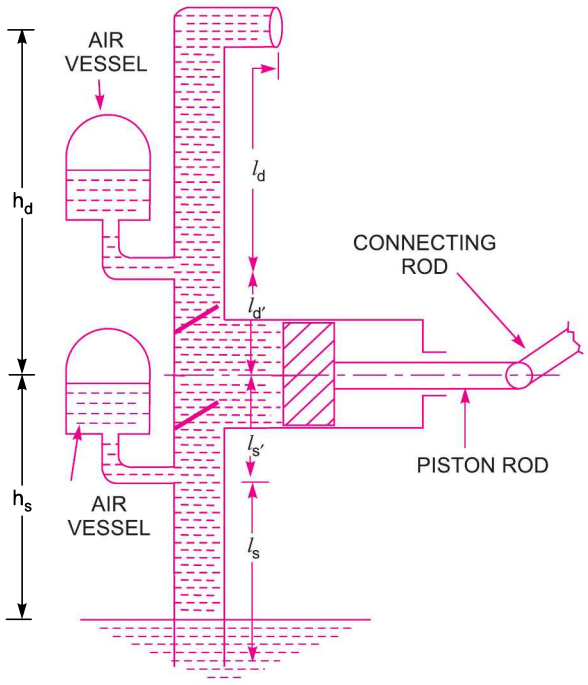


Fig. 20.9 *Air vessels fitted to reciprocating pump.*

When the air vessel is fitted to the delivery pipe, during the first half of delivery stroke, the piston moves with acceleration and forces the water into the delivery pipe with a velocity more than the mean velocity. The quantity of water in excess of the mean discharge will flow into the air vessel. This will compress the air inside the vessel. During the second half of the delivery stroke, the piston moves with retardation and the velocity of water in the delivery pipe will be less than the mean velocity. The water already stored into the air vessel will start flowing into the delivery pipe and the velocity of flow in the delivery pipe beyond the point to which air vessel is fitted will become equal to the mean velocity. Hence, the rate of flow of water in the delivery pipe will be uniform.

Let A = Cross-sectional area of the cylinder,

a = Cross-sectional area of suction or delivery pipe,

l_d = Length of delivery pipe beyond the air vessel,
l_d' = Length of delivery pipe between cylinder and air vessel,
l_s' = Length of suction pipe between cylinder and air vessel,
l_s = Length of suction pipe below air vessel,
h_{ad} = Pressure head due to acceleration in delivery pipe,
h_{as} = Pressure head due to acceleration in suction pipe,
h_{fd} = Loss of head due to friction in delivery pipe beyond the air vessel,
h_{fd}' = Loss of head due to friction in delivery pipe between cylinder and air vessel,
h_{fs} = Loss of head due to friction in suction pipe below the air vessel, and
h_{fs}' = Loss of head due to friction in suction pipe between cylinder and air vessel.

The effect of acceleration will be observed only in the lengths l_d' and l_s' which may be made very small by fitting air vessels very close to the cylinder. The velocity of flow of water in the length l_d and l_s will be equal to mean velocity of flow.

For a single-acting, discharge per second is given by equation (20.1), as

$$Q = \frac{ALN}{60}, \qquad \text{where } L = \text{Length of stroke}$$

$\therefore$ Mean velocity,
$$\bar{V} = \frac{\text{Discharge}}{\text{Area of pipe}} = \frac{Q}{a} = \frac{ALN}{60a}$$

$$= \frac{AL}{60a} \times \frac{60\omega}{2\pi} \qquad \left(\because \ \omega = \frac{2\pi N}{60} \text{ or } N = \frac{60\omega}{2\pi}\right)$$

$$= \frac{A}{a} \times L \times \frac{\omega}{2\pi} = \frac{A}{a} \times 2r \times \frac{\omega}{2\pi} \qquad (\because \ L = 2r)$$

$$= \frac{A}{a} \times \frac{\omega r}{\pi} \qquad ...(20.24)$$

The velocity of water in the suction or delivery pipes for the lengths l_s' and l_d' due to acceleration and retardation of the piston is given by equation (20.11) as

$$v = \frac{A}{a}\omega r \sin \omega t = \frac{A}{a}\omega r \sin \theta \qquad (\because \ \theta = \omega t)$$

(a) **Pressure head in the cylinder during delivery stroke.** The pressure head due to acceleration in the delivery pipe of length l_d' (between air vessel and cylinder) is given by equation (20.15) as

$$h_{ad} = \frac{l_d'}{g} \times \frac{A}{a_d}\omega^2 r \cos \theta \qquad ...(i)$$

Loss of head due to friction in the delivery pipe for lengths l_d' is given as

$$h_{fd}' = \frac{4f \times l_d' \times v^2}{d \times 2g}$$

where for delivery pipe, $v = \frac{A}{a_d}\omega r \sin \theta$

d = dia. of delivery pipe = d_d

$$\therefore \qquad h_{fd}' = \frac{4f \times l_d'}{d_d \times 2g} \times \left(\frac{A}{a_d}\omega r \sin\theta\right)^2 \qquad ...(ii)$$

Loss of head due to friction in the delivery pipe for the length beyond the air vessel (*i.e.,* length l_d),

$$h_{fd} = \frac{4f \times l_d \times \left(\overline{V}_d\right)^2}{d_d \times 2g}$$

where $\overline{V}_d$ = Mean velocity in delivery pipe = $\frac{A}{a_d} \times \frac{\omega r}{\pi}$ [from equation (20.24)]

$$\therefore \qquad h_{fd} = \frac{4f \times l_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 \qquad ...(iii)$$

The pressure head in the cylinder during delivery stroke is given as :

(*i*) At the beginning of the delivery stroke, $\theta = 0°$, $\sin\theta = 0$ and $\cos\theta = 1$ and hence total pressure head

$= (h_d + h_{ad} + h_{fd}' + h_{fd})$ + velocity head at the outlet of delivery

$$= h_d + h_{ad} + h_{fd}' + h_{fd} + \frac{\overline{V}_d^2}{2g} \qquad (\because \text{ Velocity at outlet is equal to mean velocity})$$

$$= h_d + \frac{l_d'}{g} \times \frac{A}{a_d}\omega^2 r + 0 + \frac{4f \times l_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 + \frac{\left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2}{2g} \qquad \left(\because \overline{V_d} = \frac{A}{a_d} \times \frac{\omega r}{\pi}\right)$$

$$= h_d + \frac{l_d'}{g} \times \frac{A}{a_d}\omega^2 r + \frac{4fl_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 \qquad ...(20.25)$$

(*ii*) In the middle of the stroke, $\theta = 90°$, $\sin\theta = 1$ and $\cos\theta = 0$ and hence total pressure head

$$= h_d + h_{ad} + h_{fd}' + h_{fd} + \frac{\overline{V}_d^2}{2g} \text{ above atmospheric pressure head}$$

$$= h_d + 0 + \frac{4f \times l_d'}{d_d \times 2g} \times \left(\frac{A}{a_d}\omega r\right)^2 + \frac{4f \times l_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2$$

$$= h_d + \frac{4f \times l_d'}{d_d \times 2g} \times \left(\frac{A}{a_d}\omega r\right)^2 + \frac{4f \times l_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 \qquad ...(20.26)$$

(*iii*) At the end of the delivery stroke, $\theta = 180°$, $\sin\theta = 0$ and $\cos\theta = -1$ and hence total pressure head

$$= h_d - \frac{l_d'}{g} \times \frac{A}{a_d}\omega^2 r + \frac{4f \times l_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 \qquad ...(20.27)$$

In equations (20.25), (20.26) and (20.27), the quantities

$$\left(\frac{l_d'}{g} \times \frac{A}{a_d}\omega^2 r\right) \text{ and } \left[\frac{4f \times l_d'}{d_a \times 2g} \times \left(\frac{A}{a_d}\omega r\right)^2\right]$$

are very small and can be neglected.

(*b*) **Pressure head in the cylinder during suction stroke.** The pressure head due to acceleration in the suction pipe of length l_s' (between air vessel and cylinder) is given as

$$h_{as}' = \frac{l_s'}{g} \times \frac{A}{a_s} \omega^2 r \cos\theta$$

Loss of friction head in the suction pipe of length l_s' given as

$$h_{fs}' = \frac{4f \times l_s' \times v_s^2}{d_s \times 2g}$$

where for suction pipe, the velocity (v_s) is given by equation (20.11) as $v_s = \frac{A}{a_s} \omega r \sin\theta$

Substituting the value of v_s in h_{fs}', we get

$$h_{fs}' = \frac{4 \times f \times l_s'}{d_s \times 2g} \times \left(\frac{A}{a_s} \omega r \sin\theta\right)^2$$

Loss of head due to friction in the suction pipe for the length below the air vessel (*i.e.,* lengths l_s),

$$h_{fs} = \frac{4f \times l_s}{d_s \times 2g} \times \left(\overline{V_s}\right)^2$$

where $\overline{V_s}$ is the mean velocity of flow and is given by equation (20.24) as $\overline{V_s} = \frac{A}{a_s} \times \frac{\omega r}{\pi}$

$$\therefore \qquad h_{fs} = \frac{4fl_s}{d_s \times 2g} \times \left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2$$

Hence the pressure head in the cylinder during suction stroke is given as

(*i*) At the beginning of suction stroke $\theta = 0°$, $\sin\theta = 0$ and $\cos\theta = 1$ and hence pressure head

$$= \left(h_s + h_{as}' + h_{fs}' + h_{fs}\right)_{\theta = 0°} + \frac{\overline{V_s}^2}{2g} \text{ below atmospheric pressure head}$$

$$= h_s + \frac{l_s'}{g} \times \frac{A}{a_s} \omega^2 r + 0 + \frac{4fl_s}{d_s \times 2g} \times \left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2$$

$$= h_s + \frac{l_s'}{g} \times \frac{A}{a_s} \omega^2 r + \frac{4fl_s}{d_s \times 2g}\left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2 \qquad \text{...(20.28)}$$

below atmospheric pressure head.

(*ii*) In the middle of suction stroke, $\theta = 90°$, $\sin\theta = 1$ and $\cos\theta = 0$ and hence pressure head

$$= \left(h_s + h_{as}' + h_{fs}' + h_{fs}\right)_{\theta = 90°} + \frac{\overline{V_s}^2}{2g} \text{ below atmospheric pressure head}$$

$$= h_s + 0 + \frac{4f \times l_s'}{d_s \times 2g} \times \left(\frac{A}{a_s} \omega r\right)^2 + \frac{4fl_s}{d_s \times 2g} \times \left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2$$

$$= h_s + \frac{4fl_s'}{d_s \times 2g} \times \left(\frac{A}{a_s}\omega r\right)^2 + \frac{4fl_s}{d_s \times 2g} \times \left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2 \quad ...(20.29)$$

(*iii*) At the end of the suction stroke, $\theta = 180°$, $\sin\theta = 0$ and $\cos\theta = -1$.

Hence pressure head = $\left(h_s + h_{as}' + h_{fs}' + h_{fs}\right)_{\theta = 180°} + \frac{\overline{V_s}^2}{2g}$ below atmospheric pressure head

$$= h_s - \frac{l_s'}{g} \times \frac{A}{a_s}\omega^2 r + 0 + \frac{4fl_s}{d_s \times 2g} \times \left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2}\left(\frac{A}{a_s} \times \frac{\omega r}{\pi}\right)^2 \quad ...(20.30)$$

In equations (20.28), (20.29) and (20.30), the quantities $\left(\frac{l_s'}{g} \times \frac{A}{a_s}\omega^2 r\right)$ and $\left[\frac{4f \times l_s'}{d_s \times 2g} \times \left(\frac{A}{a_s}\omega r\right)^2\right]$ are very small and may be neglected.

(*c*) **Work done by reciprocating pump with air vessels.** Work done by reciprocating pump fitted with air vessels to the suction and delivery pipe

$$= \text{Weight of water discharged per second} \times \left[h_s + h_d + h_{fs} + h_{fd} + \frac{\overline{V_s}^2}{2g} + \frac{\overline{V_d}^2}{2g} + \frac{2}{3}h_{fs}' + \frac{2}{3}h_{fd}'\right] \text{ N-m/s.}$$

where weight of water discharged per second for a single-acting pump, $W = \frac{wALN}{60} = \frac{\rho gALN}{60}$.

Also the values of h_{fs}', h_{fd}', $\frac{\overline{V_s}^2}{2g}$ and $\frac{\overline{V_d}^2}{2g}$ are very small and hence they can be neglected.

$\therefore$ Work done per sec $= \frac{\rho gALN}{60}\left[h_s + h_d + h_{fs} + h_{fd}\right]$ ...(20.31)

(*d*) **Work saved by fitting air vessel.** By fitting air vessel the head loss due to friction in suction and delivery pipe is reduced. This reduction in the head loss saves a certain amount of energy, which can be calculated by finding the work done against friction without air vessel and with air vessel. The difference of the two gives the saving in work done.

(*i*) **Work done against friction without air vessels.** Consider a single-acting reciprocating pump without any air vessels on the pipes. The velocity of flow through the pipes is given by equation (20.11) as

$$v = \frac{A}{a}\,\omega \times r \sin\theta$$

and loss of head due to friction is given by equation (20.17) as $h_f = \frac{4fl}{d \times 2g} \times \left[\frac{A}{a}\omega r \sin\theta\right]^2$.

The variation of h_f with θ is parabolic in nature and hence indicator diagram for the loss of head due to friction in pipes will be a parabola. The work done by pump against friction per stroke is equal to the area of the indicator diagram due to friction.

∴ Work done by pump per stroke against friction,

$$W_1 = \text{Area of the parabola} = \frac{2}{3} \times \text{Base} \times \text{Height}$$

$$= \frac{2}{3} \times L \times \left[\frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \omega r \right)^2 \right] \qquad (\because \quad \text{Height} = h_f \text{ at } \theta = 90°)$$

$$= \frac{2}{3} \times L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \omega r \right)^2. \qquad ...(20.32)$$

(*ii*) **Work done against friction with air vessels.** By fitting an air vessel to the pump, the velocity of flow through the pipes (except for lengths l_s' and l_d' which may be considered negligible) is uniform and equal to mean velocity of flow, which is given by equation (20.24) as

$$\overline{V} = \frac{A}{a} \times \frac{\omega r}{\pi}$$

∴ Loss of head due to friction with air vessel is given as

$$= \frac{4fl \times \overline{V}^2}{d \times 2g} = \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \times \frac{\omega r}{\pi} \right)^2$$

The head loss due to friction with air vessel is independent of θ and hence indicator diagram will be a rectangle.

∴ Work done by pump per stroke against friction,

$$W_2 = \text{Area of the rectangle}$$

$$= \text{Base} \times \text{Height} = L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \times \frac{\omega r}{\pi} \right)^2$$

$$= \frac{1}{\pi^2} \times L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \omega r \right)^2 \qquad ...(20.33)$$

The work given by equation (20.33) is less than the work given by equation (20.32). Hence, by fitting an air vessel work is saved.

(*iii*) **Work saved in a single-acting reciprocating pump.** Hence, saving in work done per stroke is obtained by subtracting equation (20.33) from equation (20.32).

∴ Work saved per stroke $= W_1 - W_2$

$$= \frac{2}{3} L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \omega r \right)^2 - \frac{1}{\pi^2} L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \omega r \right)^2$$

$$= L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \omega r \right)^2 \left[\frac{2}{3} - \frac{1}{\pi^2} \right] \qquad ...(20.34)$$

The percentage of the work saved per stroke

$$= \left(\frac{W_1 - W_2}{W_1} \right) \times 100 = \frac{L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \omega r \right)^2 \left[\frac{2}{3} - \frac{1}{\pi^2} \right]}{\frac{2}{3} L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \omega r \right)^2} \times 100$$

$$= \frac{\left(\frac{2}{3}\right) - \left(\frac{1}{\pi^2}\right)}{\left(\frac{2}{3}\right)} \times 100 = 84.8\%.$$

(*iv*) **Work saved in a double-acting reciprocating pump.** The work lost in friction per stroke in case of double-acting reciprocating pump without air vessel is the same as given in case of single-acting reciprocating pump. Hence, it is given by equation (20.32) as

$$W_1 = \frac{2}{3} \times L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a}\omega r\right)^2$$

When the air vessel is fitted to the pipe near the cylinder, the mean velocity of flow, $\overline{V}$ for double-acting is given by,

$$\overline{V} = \frac{\text{Discharge}}{\text{Area of pipe}} = \frac{Q}{a} = \frac{2ALN}{60a}$$

$$= \frac{2A \times 2r}{60a} \times \frac{60\omega}{2\pi} \qquad \left(\because \quad L = 2r \text{ and } N = \frac{60\omega}{2\pi}\right)$$

$$= \frac{2A}{a} \times \frac{\omega r}{\pi}$$

$\therefore$ Loss of head due to friction for double-acting

$$= \frac{4fl \times \overline{V^2}}{d \times 2g} = \frac{4fl}{d \times 2g} \times \left(\frac{2A}{a} \times \frac{\omega r}{\pi}\right)^2$$

$\therefore$ Work lost due to friction per stroke

$$W_2 = \text{Area of the rectangle}$$

$$= L \times \frac{4fl}{d \times 2g} \times \left(\frac{2A}{a} \times \frac{\omega r}{\pi}\right)^2$$

$$= \frac{4}{\pi^2} \times L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \times \omega r\right)^2 \qquad \text{....(20.34A)}$$

$\therefore$ Saving in work done per stroke $= \dfrac{W_1 - W_2}{W_1}$

$$= \frac{\frac{2}{3} \times L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \times \omega r\right)^2 - \frac{4}{\pi^2} \times L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a}\omega r\right)^2}{\frac{2}{3} \times L \times \frac{4fl}{d \times 2g} \times \left(\frac{A}{a} \times \omega r\right)^2} = \frac{\left(\frac{2}{3}\right) - \left(\frac{4}{\pi^2}\right)}{\left(\frac{2}{3}\right)} = 0.392 = 39.2\%.$$

(*e*) **Discharge of liquid into and from the air vessel. 1.** Let the air vessel is fitted to both suction and delivery pipes of a *single-acting* reciprocating pump. Due to air vessel, the liquid in suction and delivery pipes beyond air vessel will be moving with a constant mean velocity. This mean velocity in the pipes is given by equation (20.24) as

$$\overline{V} = \frac{A}{a} \times \frac{\omega r}{\pi}$$

The mean discharge ($\overline{Q}$) in the pipes (suction or delivery) will be equal to,

$$\overline{Q} = \overline{V} \times \text{area of pipe}$$

$$= \left(\frac{A}{a} \times \frac{\omega r}{\pi}\right) \times a \qquad (\because \quad a = \text{area of pipe})$$

$$= \frac{A \times \omega \times r}{\pi} \qquad ...(20.35)$$

The velocity of piston in the cylinder at any instant is given by equation (20.10) as

$$V = \omega r \times \sin(\omega t) = \omega r \times \sin\theta \qquad (\because \quad \omega t = \theta)$$

Hence instantaneous discharge to or from the cylinder of the pump will be as

$$Q_i = \text{Velocity of piston} \times \text{Area of piston}$$

$$= (\omega r \sin\theta) \times A \qquad (\because \quad A = \text{Area of piston})$$

$$= A\omega r \sin\theta \qquad ...(20.36)$$

The difference of the two discharges given by equations (20.36) and (20.35) will be equal to the rate of flow of liquid into or from the air vessel.

$\therefore$ Rate of flow of liquid into the air vessel

$$= \left(A\omega r \sin\theta - \frac{A\omega r}{\pi}\right) = A\omega r\left(\sin\theta - \frac{1}{\pi}\right) \qquad ...(20.37)$$

(*i*) *For air vessel fitted to delivery pipe.* The liquid will be flowing into the air vessel if equation (20.37) is positive. But if equation (20.37) is negative, then liquid will flow from the air vessel. And if equation (20.37) is zero, then no flow is taking place from or to the air vessel.

(*ii*) *For air vessel fitted to suction pipe.* If equation (20.37) is positive, then liquid is flowing from the air vessel. If equation (20.37) is negative, then liquid is flowing into the air vessel.

For no flow of liquid into or from the air vessel, the equation (20.37) should be zero.

$$\therefore \qquad A\omega r\left(\sin\theta - \frac{1}{\pi}\right) = 0 \quad \text{or} \quad \sin\theta - \frac{1}{\pi} = 0$$

or $$\sin\theta = \frac{1}{\pi} = 0.3183$$

$\therefore$ $$\theta = 18° 34' \text{ or } 161° 26'$$

2. For double-acting pump. The discharge for double-acting is given by equation (20.5) as,

$$Q = \frac{2ALN}{60} \qquad \left(\omega = \frac{2\pi N}{60} \text{ or } N = \frac{60\,\omega}{2\pi}\right)$$

$$= \frac{2AL}{60} \times \frac{60\omega}{2\pi} = \frac{AL\omega}{\pi} = \frac{A \times 2r \times \omega}{\pi} \qquad (\because \quad L = 2r)$$

$$= \frac{2A\omega r}{\pi}$$

The above discharge is the mean discharge. Hence for double-acting mean discharge (Q) is given by

$$Q = \frac{2A\omega r}{\pi}$$

But the instantaneous discharge (Q_i) to or from the cylinder of the pump will be

$$Q_i = \text{Velocity of piston} \times \text{Area of piston}$$

$$= (\omega r \sin\theta) \times A = A\omega r \sin\theta$$

Hence, rate of flow of liquid into air vessel

$$= Q_i - Q = A\omega r \sin\theta - \frac{2A\omega r}{\pi} = A\omega r\left(\sin\theta - \frac{2}{\pi}\right) \quad ...(20.38)$$

(i) For air vessel fitted to delivery pipe. If equation (20.38) is positive, the liquid is flowing into the air vessel fitted to delivery pipe. If equation (20.38) is negative, then liquid is flowing from the air vessel. And if equation (20.38) is zero then no flow is taking place into or from the air vessel.

(ii) For air vessel fitted to suction pipe. If equation (20.38) is positive, the liquid is flowing from the air vessel fitted to suction pipe. If equation (20.38) is negative, the liquid is flowing into the air vessel.

For no flow of liquid into or from the air vessel, equation (20.38) should be zero.

$$\therefore \quad A\omega r\left(\sin\theta - \frac{2}{\pi}\right) = 0 \text{ or } \sin\theta = \frac{2}{\pi} = 0.6366$$

$$\therefore \quad \theta = 39^\circ\ 32' \text{ or } 140^\circ\ 28'.$$

Problem 20.14 *The cylinder of a single-acting reciprocating pump is 15 cm in diameter and 30 cm in stroke. The pump is running at 30 r.p.m. and discharge water to a height of 12 m. The diameter and length of the delivery pipe are 10 cm and 30 m respectively. If a large air vessel is fitted in the delivery pipe at a distance of 2 m from the centre of the pump, find the pressure head in the cylinder.*

(*i*) *At the beginning of the delivery stroke, and*

(*ii*) *In the middle of the delivery stroke. Take f = .01*

Solution. Given :

Diameter of cylinder, $D = 15 \text{ cm} = 0.15 \text{ m}$

$\therefore$ Area, $A = \frac{\pi}{4} \times .15^2 = 0.01767 \text{ m}^2$

Stroke length, $L = 30 \text{ cm} = 0.30 \text{ m}$

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{0.30}{2} = 0.15 \text{ m}$

Speed of pump, $N = 30$ r.p.m.

$\therefore$ Angular speed $\omega = \frac{2\pi N}{60} = \frac{2 \times \pi \times 30}{60} = \pi \text{ rad/s.}$

Delivery head, $h_d = 12 \text{ m}$

Diameter of delivery pipe, $d_d = 10 \text{ cm} = 0.10 \text{ m}$

$\therefore$ Area, $a_d = (\pi/4)\ (.1)^2 = .007854$

Length of delivery pipe, $l = 30 \text{ m}$

Length of air vessel from the centre of the cylinder,

$$l_d' = 2 \text{ m}$$

$\therefore$ Length of delivery pipe above the air vessel,

$$l_d = l - l_d' = 30 - 2 = 28 \text{ m}$$

Co-efficient of friction, $f = 0.01$.

(*i*) The pressure head in the cylinder at the beginning of the delivery stroke is given by equation (20.25) as

$$= h_d + \frac{l_d'}{g} \times \frac{A}{a_d}\omega^2 r + \frac{4fl_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2$$

$$= 12 + \frac{2}{9.81} \times \frac{.01767}{.007854} \times \pi^2 \times .15 + \frac{4 \times .01 \times 28}{0.1 \times 2 \times 9.81}\left(\frac{.01767}{.007854} \times \frac{\pi \times .15}{\pi}\right)^2$$

$$+ \frac{1}{2 \times 9.81}\left(\frac{.01767}{.007854} \times \frac{\pi \times .15}{\pi}\right)^2$$

$$= 12 + .6709 + 0.065 + .0058 = \mathbf{12.75\ m.\ Ans.}$$

(*ii*) The pressure head in the cylinder in the middle of the delivery stroke is given by equation (20.26) as

$$= h_d + \frac{4f \times l_d'}{d_d \times 2g} \times \left(\frac{A}{a_d}\omega r\right)^2 + \frac{4fl_d}{d_d \times 2g} \times \left(\frac{A}{a_d} \times \frac{\omega r}{\pi}\right)^2 + \frac{1}{2g}\left(\frac{A}{a_d}\frac{\omega r}{\pi}\right)^2$$

$$= 12 + \frac{4 \times .01 \times 2}{0.1 \times 2 \times 9.81} \times \left(\frac{.01767}{.007854} \times \pi \times .15\right)^2 + .065 + .0058$$

$$= 12 + .0458 + .065 + .0058 = \mathbf{12.116\ m.\ Ans.}$$

Problem 20.15 *A single-acting reciprocating pump is to raise a liquid of density 1200 kg per cubic metre through a vertical height of 11.5 metres, from 2.5 metres below pump axis to 9 metres above it. The plunger, which moves with S.H.M., has diameter 125 mm and stroke 225 mm. The suction and delivery pipes are 75 mm diameter and 3.5 metres and 13.5 metres long respectively. There is a large air vessel placed on the delivery pipe near the pump axis. But there is no air vessel on the suction pipe. If separation takes place at 8.829 N/cm^2 below atmospheric pressure, find :*

(*i*) *maximum speed, with which the pump can run without separation taking place, and*

(*ii*) *power required to drive the pump, if f = 0.02.*

Neglect slip for the pump.

Solution. Given :

Liquid density, $\rho = 1200$ kg/m^3

Total vertical height $= 11.5$ m

Suction head, $h_s = 2.5$ m

Delivery head, $h_d = 9$ m

Dia. of plunger, $D = 125$ mm $= 0.125$ m

Area of plunger, $A = \frac{\pi}{4} \times .125^2 = 0.0123$ m^2

Stroke length, $L = 225$ mm $= 0.225$ m

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{.225}{2} = .1125$

Dia. of suction and delivery pipe, $d = 75$ mm $= 0.075$ m

Area of suction and delivery pipe, $a = \frac{\pi}{4}(0.075)^2 = 0.00442$ m^2

Length of suction pipe, $l_s = 3.5$ m

Length of delivery pipe, $l_d = 13.5$ m.

Air vessel is placed on the delivery side only. Hence, the velocity in the delivery pipe will be uniform. And there will be no accelerating head on delivery side.

Separation pressure $= 8.829 \frac{\text{N}}{\text{cm}^2}$ below atmospheric pressure

$= 8.829 \times 10^4 \frac{\text{N}}{\text{m}^2}$ below atmospheric pressure

$\therefore$ Separation pressure head, $h_{sep} = \frac{\text{Separation pressure}}{\rho \times g}$

$= \frac{8.829 \times 10^4}{1200 \times 9.81}$ m below atmosphere

$= 7.5$ m below atmosphere ...(i)

(i) *Maximum speed, with which pump can run without separation taking place.*

Let N = Max. speed with which pump can run without separation taking place.

The separation can take place only at the beginning of suction stroke. As air vessel is not fitted on the suction pipe, there will be accelerating head acting on suction side.

Pressure head at the beginning of suction stroke

$= h_s + h_{as}$ below atmosphere

This pressure should be equal to h_{sep} in the limiting case

$\therefore \quad 7.5 = h_s + h_{as} = 2.5 + h_{as}$

$\therefore \quad h_{as} = 7.5 - 2.5 = 5.0$ m

But 'h_{as}' at the beginning of suction stroke

$$= \frac{l_s}{g} \times \frac{A}{a} \omega^2 r$$

$\therefore$
$$5.0 = \frac{3.5}{9.81} \times \frac{0.0123}{.00442} \times \omega^2 \times .1125$$

$\therefore$
$$\omega = \sqrt{\frac{5.0 \times 9.81 \times .00442}{3.5 \times .0123 \times .1125}} = 6.69 \text{ radians/s.}$$

But
$$\omega = \frac{2\pi N}{60}$$

$\therefore$
$$N = \frac{60 \times \omega}{2\pi} = \frac{60 \times 6.69}{2\pi} = \mathbf{63.88 \text{ r.p.m. Ans.}}$$

$\therefore$ Maximum speed with which the pump can run without separation taking place is 63.88 r.p.m.

(ii) *Power required to drive the pump.*

New discharge (Q) of the single-acting pump is given by equation (20.1) as

$$Q = \frac{ALN}{60} = \frac{0.0123 \times 0.225 \times 63.88}{60} = 0.00294 \text{ m}^3/\text{s}.$$

Velocity of liquid in delivery pipe will be uniform.

$\therefore$ Q = Area of delivery pipe × Velocity = $a \times v$

$\therefore$ $$v = \frac{Q}{a} = \frac{0.00294}{0.00442} = 0.665 \text{ m/s.}$$

$\therefore$ Head loss due to friction in delivery pipe,

$$h_{fd} = \frac{4f \times l_d \times v^2}{d \times 2g} = \frac{4 \times .02 \times 13.5 \times (.665)^2}{.075 \times 2 \times 9.81} = 0.324 \text{ m.}$$

During suction stroke, the value of maximum h_{fs} is given by

$$h_{fs} = \frac{4 \times f \times l_s}{d \times 2g} \times \left(\frac{A}{a}\omega r\right)^2 = \frac{4 \times .02 \times 3.5}{.075 \times 2 \times 9.81}\left(\frac{.0123}{.00442} \times 6.69 \times .1125\right)^2$$

$$= 0.834 \text{ m}$$

Now power required to drive the pump in kW

$$= \frac{\text{Work done/s}}{1000} = \frac{\rho \times g \times Q}{1000} \times \left[h_s + h_d + \frac{2}{3}h_{fs} + h_{fd}\right]$$

$$= \frac{1200 \times 9.81 \times .00294}{1000} \times \left[2.5 + 9.0 + \frac{2}{3} \times .834 + .324\right]$$

$$= \mathbf{0.428 \text{ kW. Ans.}}$$

Problem 20.16 *A double-acting reciprocating piston pump is pumping water (diameter of the piston 250 mm, diameter of piston rod, which is on one side of the piston 50 mm, piston stroke 380 mm). The suction and discharge heads are 4.5 m and 18.6 m respectively. Find the work done by the piston during outward stroke. Would the work done change for the inward stroke?*

Solution. Given :

Dia. of piston, $D = 250 \text{ mm} = 0.25 \text{ m}$

$\therefore$ Area of piston, $A = \frac{\pi}{4} \times D^2 = \frac{\pi}{4} \times 0.25^2 \text{ m}^2 = 0.0491 \text{ m}^2$

Dia. of piston rod, $d = 50 \text{ mm} = 0.05 \text{ m}$

$\therefore$ Area of piston rod, $a = \frac{\pi}{4} \times d^2 = \frac{\pi}{4} \times 0.05^2 = 0.001963 \text{ m}^2$

Stroke length, $L = 380 \text{ mm} = 0.380 \text{ m}$

Suction head, $h_s = 4.5 \text{ m}$

Discharge or delivery head, $h_d = 18.6 \text{ m}$

Find work done during outward stroke

In a double-acting pump for outward stroke, suction side will be towards the piston and delivery side will be towards the piston rod.

Hence, total work done during outward stroke

= Weight of water lifted × height through which water is lifted

+ Weight of water delivered × height through which water is delivered

$= \rho \times g \times Q_1 \times h_s + \rho \times g \times Q_2 \times h_d$

where during outward stroke, $Q_1 = A \times L = 0.0491 \times 0.380 = 0.01865\ m^3$

$Q_2 = (A - a) \times L$

$= (0.0491 - 0.001963) \times 0.380 = 0.01791\ m^3$

and $\rho \times g = 1000 \times 9.81\ N/m^3$

$\therefore$ Total work done during outward stroke

$= (1000 \times 9.81 \times 0.01865 \times 4.5 + 1000 \times 9.81 \times 0.01791 \times 18.6)\ Nm$

$= 9.81 \times 0.01865 \times 4.5 + 9.81 \times 0.01791 \times 18.6\ (kJ)$

$= (0.8233 + 3.268)\ kJ$ ($\because$ J = Nm)

$=$ **4.0913 kJ. Ans.**

For inward stroke, suction side will be towards the piston rod whereas the delivery side will be towards the piston.

$\therefore$ Total work done during inward stroke

$= \rho \times g \times Q_2 \times h_s + \rho \times g \times Q_1 \times h_d = \rho \times g\ (Q_2 \times h_s + Q_1 \times h_d)$

$= 1000 \times 9.81\ (0.01791 \times 4.5 + 0.01865 \times 18.6)\ Nm$

$= 1000 \times 9.81\ (0.0806 + 0.3468)\ J$

$= 9.81\ (0.0806 + 0.3468)\ kJ = 4.192\ kJ$

Hence, work done during inward stroke will be different. **Ans.**

Problem 20.17 *A single-acting reciprocating pump has a plunger diameter of 250 mm and stroke of 450 mm and it is driven with S.H.M. at 60 r.p.m. The length and diameter of delivery pipe are 60 m and 100 mm respectively. Determine the power saved in overcoming friction in the delivery pipe by fitting an air vessel on the delivery side of the pump. Assume friction factor = 0.01.*

Solution. Given :

Dia. of plunger, $D = 250\ mm = 0.25\ m$

$\therefore$ Area, $A = \frac{\pi}{4} \times 0.25^2$

Stroke length, $L = 450\ mm = 0.45\ m$

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{0.45}{2} = 0.225\ m$

Speed, $N = 60$ r.p.m.

$\therefore$ Angular speed, $\omega = 2\pi N/60 = 2\pi \times 60/60 = 2\pi$ rad/s.

Length of delivery pipe, $L = 60\ m$

Dia. of delivery pipe, $d = 100\ mm = 0.1\ m$

$\therefore$ Area of pipe, $a = \frac{\pi}{4} \times 0.1^2$

Friction factor, $f = 0.01$

Power saved is given by,

Power saved $= \rho \times g \times Q \times \left[\frac{2}{3}(h_f)_{\text{without air vessel}} - (h_f)_{\text{with air vessel}}\right]$

where $\rho \times g = 1000 \times 9.81 \frac{\text{N}}{\text{m}^3}$

$$Q = \frac{ALN}{60} = \frac{\pi}{4} \times \frac{0.25^2 \times 0.45 \times 60}{60} = 0.02209 \text{ m}^3/\text{s}$$

Without air vessel, $$h_f = \frac{f^* \times L \times v^2}{d \times 2g} = \frac{f \times L}{d \times 2g} \times \left(\frac{A}{a} \omega \times r\right)^2 \qquad \left[\because \; v = \frac{A}{a} \times \omega \times r\right]$$

$$= \frac{0.01 \times 60}{0.1 \times 2 \times 9.81} \times \frac{\left(\frac{\pi}{4} \times 0.25^2 \times 2\pi \times 0.225\right)^2}{\frac{\pi}{4} \times 0.1^2} = 23.87 \text{ m}$$

With air vessel, $$h_f = \frac{f^* \times L \times \overline{V}^2}{d \times 2g}$$

$$= \frac{f^* \times L}{d \times 2g} \times \left(\frac{A}{a} \times \frac{\omega r}{\pi}\right)^2, \text{ where } \overline{V}^2 = \frac{A}{a} \times \frac{\omega \times r}{\pi}$$

$$= \frac{0.01 \times 60}{0.1 \times 2 \times 9.81} \times \left(\frac{\frac{\pi}{4} \times 0.25^2}{\frac{\pi}{4} \times 0.1^2} \times \frac{2\pi \times 0.225}{\pi}\right)^2 = 2.419 \text{ m.}$$

$\therefore$ Power saved $$= \rho \times g \times Q \times \left[\frac{2}{3}(h_f)_{\text{without air vessel}} - (h_f)_{\text{with air vessel}}\right]$$

$$= 1000 \times 9.81 \times 0.02209 \left[\frac{2}{3} \times 23.87 - 2.419\right] \text{ W}$$

$$= 9.81 \times 0.02209 \left[\frac{2}{3} \times 23.87 - 2.419\right] \text{ kW} = \mathbf{2.924 \text{ kW. Ans.}}$$

Problem 20.18 *A double-acting reciprocating pump runs at 120 r.p.m. When its suction pipe of 100 mm diameter is fitted with an air vessel on its suction side. The diameter of cylinder and stroke are 150 mm and 450 mm respectively. If piston is to be driven with S.H.M., find the rate of flow from or into the air vessel when the crank makes angles of 30° , 90° and 120° with the inner dead centre. Find also the crank angles at which there is no flow into or from the air vessel.*

Solution. Given :

Speed, $N = 120$ r.p.m.

$\therefore$ Angular speed, $$\omega = \frac{2\pi N}{60} = \frac{2\pi \times 120}{60} = 4\pi \text{ rad/s}$$

* Here friction factor is given and hence the formula is $h_f = \frac{f \times L \times V^2}{d \times 2g}$ and not $\frac{4f \times L \times V^2}{d \times 2g}$.

Dia. of suction pipe = 100 mm = 0.1 m

$\therefore$ Area of suction pipe, $a = \frac{\pi}{4}(0.1)^2 = 0.007854 \text{ m}^2$

Dia. of cylinder = 150 mm = 0.15 m

$\therefore$ Area of cylinder, $A = \frac{\pi}{4}(0.15)^2 = 0.01767 \text{ m}^2$

Stroke length, $L = 450 \text{ mm} = 0.45 \text{ m}$

$\therefore$ Crank radius, $r = \frac{L}{2} = \frac{0.45}{2} = 0.225 \text{ m}$

Find rate of flow from or into air vessel when $\theta = 30°$, $90°$ and $120°$

The rate of flow of liquid for double-acting pump into the air vessel is given by equation (20.38).

$\therefore$ Rate of flow of liquid into air vessel

$$= A\omega r\left(\sin\theta - \frac{2}{\pi}\right) = 0.01767 \times 4\pi \times 0.225\left(\sin\theta - \frac{2}{\pi}\right) = 0.04996\left(\sin\theta - \frac{2}{\pi}\right)$$

In this problem, air vessel is fitted to the suction pipe. Hence if the above rate of flow is positive, the liquid will be flowing from the air vessel. And if the above rate of flow is negative, the liquid will be flowing into the air vessel.

(*i*) *For* $\theta = 30°$.

The above rate of flow $= 0.04996\left(\sin 30° - \frac{2}{\pi}\right) = 0.04996(0.5 - 0.6366) =$ **-0.00682 m^3/s. Ans.**

Since the rate of flow is negative, hence the flow is taking place into the air vessel.

(*ii*) *For* $\theta = 90°$.

The rate of flow becomes $= 0.04996\left(\sin 90° - \frac{2}{\pi}\right) = 0.04996(1 - 0.6366) =$ **0.0181 m^3/s. Ans.**

As it is positive, hence rate of flow is taking place from the air vessel.

(*iii*) *For* $\theta = 120°$.

The rate of flow becomes $= 0.04996\left(\sin 120° - \frac{2}{\pi}\right) = 0.04996(0.866 - 0.6366) =$ **0.01146 m^3/s. Ans.**

As it is positive, hence rate of flow is taking place from the air vessel.

(*iv*) *Crank angle at which there is no flow into or from air vessel*

Let θ = angle at which there is no flow. But rate of flow

$$= 0.04996\left(\sin\theta - \frac{2}{\pi}\right)$$

For no flow from or into air vessel

$$0.04996\left(\sin\theta - \frac{2}{\pi}\right) = 0 \text{ or } \sin\theta = \frac{2}{\pi} = 0.6366$$

$\therefore$ $\theta = \sin^{-1} 0.6366 =$ **39° 32′** and **140° 28′. Ans.**

▶ 20.10 COMPARISON BETWEEN CENTRIFUGAL PUMPS AND RECIPROCATING PUMPS

Centrifugal pumps	*Reciprocating pumps*
1. The discharge is continuous and smooth.	1. The discharge is fluctuating and pulsating.
2. It can handle large quantity of liquid.	2. It handles small quantity of liquid only.
3. It can be used for lifting highly viscous liquids.	3. It is used only for lifting pure water or less viscous liquids.
4. It is used for large discharge through smaller heads.	4. It is meant for small discharge and high heads.
5. Cost of centrifugal pump is less as compared to reciprocating pump.	5. Cost of reciprocating pump is approximately four times the cost of centrifugal pump.
6. Centrifugal pump runs at high speed. They can be coupled to electric motor.	6. Reciprocating pump runs at low speed. Speed is limited due to consideration of separation and cavitation.
7. The operation of centrifugal pump is smooth and without much noise. The maintenance cost is low.	7. The operation of reciprocating pump is complicated and with much noise. The maintenance cost is high.
8. Centrifugal pump needs smaller floor area and installation cost is low.	8. Reciprocating pump requires large floor area and installation cost is high.
9. Efficiency is high.	9. Efficiency is low.

HIGHLIGHTS

1. A reciprocating pump consists of a cylinder with a piston, a suction pipe, a delivery pipe, a suction valve and a delivery valve.

2. Discharge through a pump per second is given as

$$Q = \frac{ALN}{60} \quad \text{...For a single-acting}$$

$$= \frac{2ALN}{60} \quad \text{...For a double-acting.}$$

3. Work done by reciprocating pump per second

$$= \frac{\rho g ALN}{60}(h_s + h_d) \quad \text{...For a single-acting}$$

$$= \frac{2\rho g ALN}{60}(h_s + h_d) \quad \text{...For a double-acting.}$$

4. Slip of a pump is defined as the difference between the theoretical discharge and actual discharge of the pump.

5. The pressure head (h_a) due to acceleration in the suction and delivery pipes is given as

$$h_{as} = \frac{l_s}{g} \times \frac{A}{a_s}\omega^2 r \cos\theta \quad \text{...For suction pipe}$$

$$h_{ad} = \frac{l_d}{g} \times \frac{A}{a_d}\omega^2 r \cos\theta \quad \text{... For delivery pipe.}$$

6. The loss of head due to friction in suction and delivery pipes is obtained from

$$h_f = \frac{4fl}{d \times 2g} \times \left(\frac{A}{a}\omega r \sin\theta\right)^2.$$

7. Indicator diagram is a graph between the pressure head in the cylinder and the distance travelled by the piston from inner dead centre for one complete revolution of the crank.
8. Work done by the pump is proportional to the area of the indicator diagram. Area of ideal indicator diagram is the same as the area of indicator diagram due to acceleration in suction and delivery pipes.
9. Work done by the pump per second due to acceleration and friction in suction and delivery pipes

$$= \frac{\rho g A L N}{60}\left(h_s + h_d + \frac{2}{3}h_{fs} + \frac{2}{3}h_{fd}\right) \quad \text{...For a single-acting}$$

$$= \frac{2\rho g A L N}{60}\left(h_s + h_d + \frac{2}{3}h_{fs} + \frac{2}{3}h_{fd}\right) \quad \text{...For a double-acting.}$$

10. Air vessel is used to obtain a continuous supply of water at uniform rate, to save a considerable amount of work and to run the pump at a high speed without separation.
11. Mean velocity ($\overline{V}$) for a single-acting pump is given as

$$\overline{V} = \frac{A}{a}\frac{\omega r}{\pi}.$$

12. Work done by reciprocating with air vessels fitted to suction and delivery pipes

$$\simeq \frac{\rho g A L N}{60}\left[h_s + h_d + h_{fs} + h_{fd}\right].$$

13. Work saved by fitting air vessels in a single-acting reciprocating pump is 84.8% while in a double-acting reciprocating pump, the work saved is 39.2%.

EXERCISE

(A) THEORETICAL PROBLEMS

1. What is a reciprocating pump ? Describe the principle and working of a reciprocating pump with a neat sketch. Why is a reciprocating pump not coupled directly to the motor ? Discuss the reason in detail.
2. Differentiate : (*i*) Between a single-acting and double-acting reciprocating pump, (*ii*) Between a single cylinder and double cylinder reciprocating pump.
3. Define slip, percentage slip and negative slip of a reciprocating pump.
4. How will you classify the reciprocating pumps ?
5. What is the effect of acceleration of the piston on the velocity and acceleration of the water in suction and delivery pipes? Obtain an expression for the pressure head due to acceleration in the suction and delivery pipes.
6. Find an expression for the head lost due to friction in suction and delivery pipes.
7. Define indicator diagram. How will you prove that area of indicator diagram is proportional to the work done by the reciprocating pump?
8. What is the effect of acceleration in suction and delivery pipes on indicator diagram ? Does the area of the indicator diagram change as compared to the area of ideal indicator diagram ?
9. Draw an indicator diagram, considering the effect of acceleration and friction in suction and delivery pipes. Find an expression for the work done per second in case of single-acting reciprocating pump.
10. What is an air vessel ? Describe the function of the air vessel for reciprocating pumps.

11. Show from first principle that the work saved, against friction in the delivery pipe of a single-acting reciprocating pump, by fitting an air vessel is 84.8% while for a double-acting reciprocating pump the work saved is only 39.20%.
12. What is negative slip in a reciprocating pump ? Explain with neat sketches the function of air vessels in a reciprocating pump.
13. Differentiate, with examples between :
    (*i*) Turbines and pumps, (*ii*) Impulse and reaction turbines,
    (*iii*) Radial and axial flow turbines, flow turbines, (*iv*) Inward and outward radial, and
    (*v*) Kaplan and propeller turbines.
14. Explain in brief how and when separation of flow takes place in a reciprocating pump. Discuss the preventive measures usually adopted for effective reduction of separation in such a pump.
15. Why is it that the speed of a reciprocating pump without air vessels is not high ? Explain with sketches.
16. Derive an expression for the head lost due to friction in the delivery pipe of a reciprocating pump with and without an air vessel.

(B) NUMERICAL PROBLEMS

1. A single-acting reciprocating pump running at 30 r.p.m., delivers 0.012 m^3/s of water. The diameter of the piston is 25 cm and stroke length is 50 cm. Determine : (*i*) The theoretical discharge of the pump, (*ii*) Co-efficient of discharge, and (*iii*) Slip and percentage slip of the pump.
   [**Ans.** (*i*) 0.01227 m^3/s, (*ii*) 0.978, (*iii*) .00027 m^2/s and 2.20%]
2. A double-acting reciprocating pump, running at 50 r.p.m. is discharging 900 litres of water per minute. The pump has stroke of 400 mm. The diameter of piston is 250 mm. The delivery and suction heads are 25 m and 4 m respectively. Find the slip of the pump and power required to drive the pump.
   [**Ans.** .0027 m^3/s, 9.3 kW]
3. A single-acting reciprocating pump has a cylinder of a diameter 150 mm and of stroke length 300 mm. The centre of the pump is 4 m above the water surface in the sump. The atmospheric pressure head is 10.3 m of water and pump is running at 40 r.p.m. If the length and diameter of the suction pipe are 5 m and 10 cm respectively, determine the pressure head due to acceleration in the cylinder : (*i*) At the beginning of the suction stroke, and (*ii*) In the middle of suction stroke. [**Ans.** (*i*) 3.018 m, (*ii*) 0]
4. If in Problem 3, the length and diameter of delivery pipe are 35 m and 100 mm respectively and water is delivered by the pump to a tank which is 25 m above the centre of the pump, determine the pressure head in the cylinder : (*i*) At the beginning of the delivery stroke, (*ii*) In the middle of the stroke, and (*iii*) At the end of the delivery stroke. [**Ans.** (*i*) 5.426 m (abs.), (*ii*) 35.3 m (abs.), (*iii*) 14.174 m (abs.)]
5. A single-acting reciprocating pump has piston diameter 15 cm and stroke length 30 cm. The centre of the pump is 5 m above the water level in the sump. The diameter and length of the suction pipe are 10 cm and 8 m respectively. The separation occurs if the absolute pressure head in the cylinder during suction stroke falls below 2.5 m of water. Calculate the maximum speed at which the pump can run without separation. Take atmospheric pressure head = 10.3 m of water. [**Ans.** 30.45 r.p.m.]
6. A single-acting reciprocating pump has a plunger of 100 mm diameter and a stroke length 200 mm. The centre of the pump is 3 m above the water level in the sump and 20 m below the water level in a tank to which water is delivered by the pump. The diameter and length of suction pipe are 50 mm and 5 m while of the delivery pipe are 40 mm and 30 m respectively. Determine the maximum speed at which the pump may be run without separation, if separation occurs at 7.3575 N/cm^2 below the atmospheric pressure. Take atmospheric pressure head = 10.3 m of water. [**Ans.** 36.22 r.p.m.]
7. The diameter and stroke length of a single-acting reciprocating pump are 100 mm and 200 mm respectively. The lengths of suction and delivery pipes are 10 m and 30 m respectively and their diameters are 50 mm. If the pump is running at 30 r.p.m. and suction and delivery heads are 3.5 m and 20 m respectively,

find the pressure head in the cylinder : (*i*) at the beginning of the suction and delivery stroke, (*ii*) in the middle of suction and delivery stroke, and (*iii*) at the end of the suction and delivery stroke. Take atmospheric pressure head = 10.3 m of water and co-efficient of friction = .009 for both pipes.

[**Ans.** (*i*) 2.776 m (abs.), 42.373 m (abs.), (*ii*) 6.22 m, 43.34 m, (*iii*) Not possible, 18.225 m]

8. For Problem 7, find the power required to drive the pump, if the liquid flowing through the pump is water.

[**Ans.** 0.25 kW]

9. The cylinder of a single-acting reciprocating pump is 125 mm in diameter and 250 mm in stroke. The pump is running at 40 r.p.m. and discharge water to a height of 15 m. The diameter and length of the delivery pipe are 100 mm and 30 m respectively. If a large air vessel is fitted in the delivery pipe at a distance of 1.5 m from the centre of the pump, find the pressure head in the cylinder : (*i*) At the beginning of the delivery stroke, and (*ii*) In the middle of the delivery stroke. Take the efficiency of friction = .01.

[**Ans.** (*i*) 15.566 m, (*ii*) 15.07 m]

21 CHAPTER

FLUID SYSTEM

▶ 21.1 INTRODUCTION

Fluid system is defined as the device in which power is transmitted with the help of a fluid which may be liquid (water or oil) or a gas (air) under pressure. Most of these devices are based on the principles of fluid statics and fluid kinematics. In this chapter, the following devices will be discussed:

1. The hydraulic press,
2. The hydraulic accumulator,
3. The hydraulic intensifier,
4. The hydraulic ram,
5. The hydraulic lift,
6. The hydraulic crane,
7. The fluid or hydraulic coupling,
8. The fluid or hydraulic torque converter,
9. The air lift pump, and
10. The gear-wheel pump.

▶ 21.2 THE HYDRAULIC PRESS

The hydraulic press is a device used for lifting heavy weights by the application of a much smaller force. It is based on Pascal's law, which states that the intensity of pressure in a static fluid is transmitted equally in all directions.

The hydraulic press consists of two cylinders of different diameters. One of the cylinder is of large diameter and contains a ram, while the other cylinder is of smaller diameter and contains a plunger as shown in Fig. 21.1. The two cylinders are connected by a pipe. The cylinders and pipe contain a liquid through which pressure is transmitted.

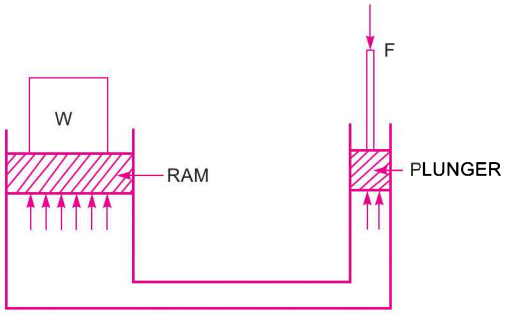


Fig. 21.1 *The hydraulic press.*

When a small force F is applied on the plunger in the downward direction, a pressure is produced on the liquid in contact with the plunger. This pressure is transmitted equally in all directions and acts on the ram in the upward direction as shown in Fig. 21.1. The heavier weight placed on the ram is then lifted up.

Let W = Weight to be lifted,
F = Force applied on the plunger,

A = Area of ram,
a = Area of plunger, and
p = Pressure intensity produced by force F.

$$= \frac{\text{Force } F}{\text{Area of plunger}} = \frac{F}{a}$$

Due to Pascal's law, the above intensity of pressure will be equally transmitted in all directions. Hence, the pressure intensity at the ram will be = $p = \frac{F}{a}$.

But the pressure intensity on ram is also = $\frac{\text{Weight}}{\text{Area of ram}} = \frac{W}{A}$.

Equating the pressure intensity on ram, $\frac{F}{a} = \frac{W}{A}$

$$\therefore \qquad W = \frac{F}{a} \times A. \qquad \text{...(21.1)}$$

21.2.1 Mechanical Advantage. The ratio of weight lifted to the force applied on the plunger is defined as the mechanical advantage. Mathematically, mechanical advantage is written as

$$\text{M. A.} = \frac{W}{F}. \qquad \text{...(21.2)}$$

21.2.2 Leverage of the Hydraulic Press. If a lever is used for applying force on the plunger, then a force F' smaller than F can lift the weight W as shown in Fig. 21.2. The ratio of L/l is called the leverage of the hydraulic press.

Taking moments about Q, $F' \times L = F \times l$

$$\therefore \qquad F = F' \times \frac{L}{l} \qquad \text{...(21.3)}$$

Substituting the value of F in equation (21.1), we get the expression for weight lifted as

$$W = \left(F' \times \frac{L}{l}\right) \times \frac{A}{a} = F' \times \frac{L}{l} \times \frac{A}{a} \qquad \text{...(21.4)}$$

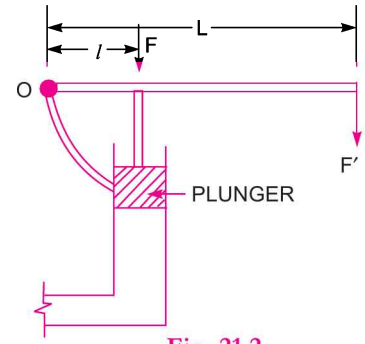


Fig. 21.2

21.2.3 Actual Heavy Hydraulic Press. Based on the nature of the work required, actual hydraulic press is different in shape. But all actual hydraulic press consist of a ram sliding in a cylinder to which high-pressure liquid is forced.

Fig. 21.3 shows one of the actual hydraulic press. It consists of a fixed cylinder in which a ram is sliding. To the lower end of the ram, movable plate is attached. As the ram moves up and down, the movable plate attached to the ram also moves up and down between two fixed plates. When any liquid under high pressure is supplied into the cylinder, the ram

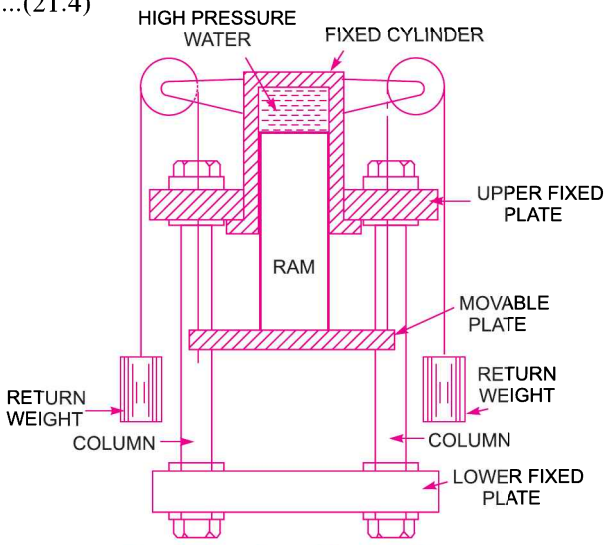


Fig. 21.3 *Actual hydraulic press.*

moves in the downward direction and exerts a force equal to the product of intensity of pressure supplied and area of the ram, on any material placed between the lower fixed plate and the movable plate. Thus the material gets pressed.

To bring back the ram in the upward position, the liquid from the cylinder is taken out. Then by the action of the return weights, the ram along with the movable plate will move up.

Problem 21.1 *A hydraulic press has a ram of 300 mm diameter and a plunger of 45 mm diameter. Find the weight lifted by the hydraulic press when the force applied at the plunger is 50 N.*

Solution. Given :

Diameter of ram, $D = 300 \text{ mm} = 0.30 \text{ m}$

Diameter of plunger, $d = 45 \text{ mm} = 0.045 \text{ m}$

Force on plunger, $F = 50 \text{ N}$

Let weight lifted $= W \text{ N}$

Area of ram, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} (0.30)^2 = 0.07068 \text{ m}^2$

Area of plunger, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} (.045)^2 = .00159 \text{ m}^2$

The weight lifted (W) is given by equation (21.1) as

$$W = \frac{F}{a} \times A = \frac{50 \times .07068}{.00159} = \mathbf{2222.64 \text{ N. Ans.}}$$

Problem 21.2 *A hydraulic press has a ram of 200 mm diameter and a plunger of 30 mm diameter. It is used for lifting a weight of 3 kN. Find the force required at the plunger.*

Solution. Given :

Diameter of ram, $D = 200 \text{ mm} = 0.20 \text{ m}$

$\therefore$ Area of ram, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} \times (0.20)^2 = 0.0314 \text{ m}^2$

Diameter of plunger, $d = 30 \text{ mm} = .03 \text{ m}$

$\therefore$ Area of plunger, $a = \frac{\pi}{4} (.03)^2 = 7.068 \times 10^{-4} \text{ m}^2$

Weight lifted, $W = 3 \text{ kN} = 3 \times 1000 = 3000 \text{ N}.$

Let the force on plunger $= F.$

Using relation given equation (21.1),

$$W = \frac{F}{a} \times A$$

$$F = \frac{W \times a}{A} = \frac{3000 \times 7.068 \times 10^{-4}}{.0314} = \mathbf{67.52 \text{ N. Ans.}}$$

Problem 21.3 *If in the problem 21.2, a lever is used for applying force on the plunger, find the force required at the end of the lever if the ratio l/L is 1/10.*

Solution. Given :

$D = 0.20 \text{ m}, \ A = 0.0314 \text{ m}^2$

$d = 0.03 \text{ m}, \ a = 7.068 \times 10^{-4} \text{ m}^2$

$$W = 3000 \text{ N}, \frac{l}{L} = \frac{1}{10}$$

Let F' = Force required at the end of the lever.

Using equation (21.4), $W = F' \times \frac{L}{l} \times \frac{A}{a}$

$$\therefore \quad F' = W \times \frac{l}{L} \times \frac{a}{A} = 3000 \times \frac{1}{10} \times \frac{7.068 \times 10^{-4}}{0.0314} = \textbf{6.752 N. Ans.}$$

Problem 21.4 *If in the problem 21.1, the stroke of the plunger is 100 mm, find the distance travelled by the weight in 100 strokes. Determine the work done during 100 strokes.*

Solution. The data given in problem 21.1 :

$D = 0.30$ m, $A = 0.07068$ m^2, $d = 0.045$ m, $a = .00159$ m^2

$F = 50$ N and W (calculated) = 2222.64 N

Stroke of plunger = 100 mm = 0.10 m

Number of strokes = 100

Volume of liquid displaced by plunger in one stroke

= Area of plunger × Stroke of plunger

= $a \times 0.10$ m^3 = $.00159 \times 0.10$ = .000159 m^3.

The liquid displaced by plunger will enter the cylinder in which ram is fitted and this liquid will move the ram in the upward direction.

Let the distance moved by the ram or weight in one stroke

= x m

Then volume displaced by ram in one stroke

= Area of ram × x = $A \times x = 0.07068 \times x$ m^3

As volume displaced by plunger and ram is the same,

$$\therefore \quad .000159 = .07068 \times x$$

$$\therefore \quad x = \frac{.000159}{.07068} = .00225 \text{ m}$$

$\therefore$ Distance moved by weight in 100 strokes

= $x \times 100 = .00225 \times 100 =$ **0.225 m. Ans.**

Work done during 100 strokes = Weight lifted × Distance moved

= $W \times 0.225 = 2222.64 \times 0.225$ Nm = **500.094 Nm. Ans**.

Problem 21.5 *A hydraulic press has a ram of 150 mm diameter, plunger of 20 mm diameter. The stroke of the plunger is 200 mm and weight lifted is 800 N. If the distance moved by the weight is 1.0 m in 20 minutes determine :*

(*i*) *The force applied on the plunger,* (*ii*) *Power required to drive the plunger, and*

(*iii*) *Number of strokes performed by the plunger.*

Solution. Given :

Diameter of ram, $D = 150$ mm = 0.15 m

Diameter of plunger, $d = 20$ mm = 0.02 m

Stroke of plunger = 200 mm = 0.20 m

Weight lifted, $W = 800$ N

Distance moved by weight = 1.0 m

Time taken by weight = 20 minutes

Now, area of ram, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} (.15)^2 = 0.01767 \text{ m}^2$

Area of plunger, $a = \frac{\pi}{4} (.02)^2 = .00031416 \text{ m}^2$.

(*i*) Let the force applied on the plunger = F.

Using equation (21.1), we have $W = \frac{F}{a} \times A$

$$\therefore \quad F = \frac{W \times a}{A} = \frac{800 \times .00031416}{.01767} = \textbf{14.22 N. Ans.}$$

(*ii*) Work done by the press per second

$$= \frac{\text{Weight lifted} \times \text{Distance travelled}}{\text{Time}}$$

$$= \frac{800 \times 1.0}{20 \times 60} = 0.6667 \text{ Nm/s}$$

$\therefore$ Power required to drive the plunger

$$= \frac{\text{Work done per sec}}{1000} = \frac{0.6667}{1000} = \textbf{0.000666 kW. Ans.}$$

(*iii*) Volume of liquid displaced by plunger in one stroke

= Area of plunger × Stroke length

= .00031416 × 0.20 = .000062832 m^3

Total volume of liquid displaced in cylinder

= Area of ram × Distance moved by weight

= .01767 × 1.0 = 0.01767 m^3

$\therefore$ Number of strokes performed by plunger or pump

$$= \frac{\text{Total volume of liquid displaced}}{\text{Volume of liquid displaced per stroke}}$$

$$= \frac{0.01767}{.000062832} = 281.22 \simeq \textbf{281. Ans.}$$

▶ 21.3 THE HYDRAULIC ACCUMULATOR

The hydraulic accumulator is a device used for storing the energy of a liquid in the form of pressure energy, which may be supplied for any sudden or intermittent requirement. In case of hydraulic lift or the hydraulic crane, a large amount of energy is required when lift or crane is moving upward. This energy is supplied from hydraulic accumulator. But when the lift is moving in the downward direction, no large external energy is required and at that time, the energy from the pump is stored in the accumulator.

Fig. 21.4 shows a hydraulic accumulator which consists of a fixed vertical cylinder containing a sliding ram. A heavy weight is placed on the ram. The inlet of the cylinder is connected to the pump, which continuously supplies water under pressure to the cylinder. The outlet of the cylinder is connected to the machine (which may be lift or crane etc.)

The ram is at the lowermost position in the beginning. The pump supplies water under pressure continuously. If the water under pressure is not required by the machine (lift or crane), the water under pressure will be stored in the cylinder. This will raise the ram on which a heavy weight is placed. When the ram is at the uppermost position, the cylinder is full of water and accumulator has stored the maximum amount of pressure energy. When the machine (lift or crane) requires a large amount of energy, the hydraulic accumulator will supply this energy and ram will move in the downward direction.

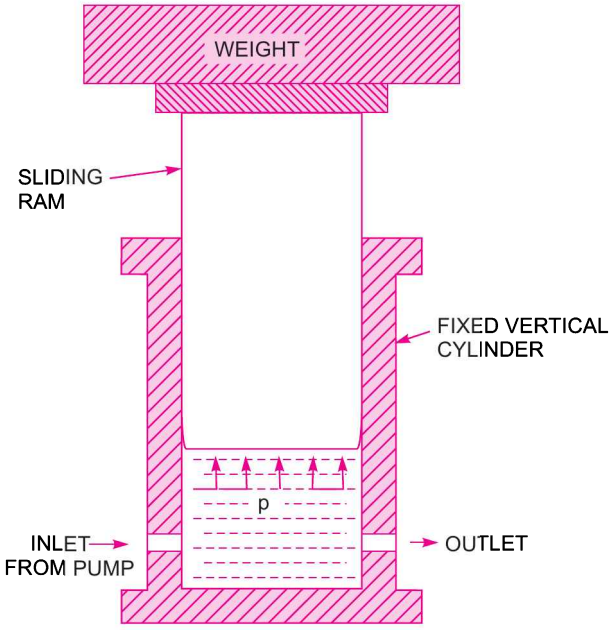


Fig. 21.4 *The hydraulic accumulator.*

21.3.1 Capacity of Hydraulic Accumulator. It is defined as the maximum amount of hydraulic energy stored in the accumulator. The expression for the capacity of accumulator is obtained as :

Let A = Area of the sliding ram,

L = Stroke or lift of the ram,

p = Intensity of water pressure supplied by the pump, and

W = Weight placed on the ram (including the weight of ram),

W = Intensity of pressure × Area of ram

Then $= p \times A$

The work done in lifting the ram $= W \times$ Lift of ram $= WL$

$= p \times A \times L$ $(W = p \times A)$

The work done in lifting the ram is also the energy stored in the accumulator. And energy stored is equal to the capacity of the accumulator.

$\therefore$ Capacity of accumulator = Work done in lifting the ram

$= p \times A \times L$ ...(21.5)

But $A \times L$ = Volume of accumulator

$\therefore$ Capacity of accumulator $= p \times$ Volume of accumulator. ...(21.6)

Problem 21.6 *Determine the length of stroke for an accumulator having a displacement of 115 litres. The diameter of the plunger is 350 mm.*

Solution. Given :

Displacement = 115 litres = 0.115 m^3

or Volume of accumulator = 0.115 m^3

Dia. of plunger, D = 350 mm = 0.35 m

$\therefore$ Area of plunger, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} \times .35^2$ m^2

But volume of accumulator $= A \times L$

where L = length of stroke.

$\therefore$ Volume $= \frac{\pi}{4} D^2 \times L$ or $0.115 = \frac{\pi}{4} \times (0.35)^2 \times L$

$\therefore$ $L = \frac{0.115 \times 4}{\pi \times 0.35^2}$ = **1.195 m. Ans.**

Problem 21.7 *The water is supplied at a pressure of 14 N/cm^2 to an accumulator, having a ram of diameter 1.5 m. If the total lift of the ram is 8 m, determine :*

(*i*) *The capacity of the accumulator, and*

(*ii*) *Total weight placed on the ram (including the weight of ram).*

Solution. Given :

Supply pressure, p = 14 N/cm^2 = 14×10^4 N/m^2

Dia. of ram, D = 1.5 m

$\therefore$ Area of ram, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} (1.5)^2 = 1.767$ m^2

Lift of ram, L = 8 m.

(*i*) Capacity of accumulator is given by equation (21.5) as

Capacity $= p \times A \times L = 14 \times 10^4 \times 1.767 \times 8$

$= 1979.04 \times 10^3$ Nm = **1979.04 kNm. Ans.**

(*ii*) Total weight (W), placed on the ram is given by

$W = p \times A = 14 \times 10^4 \times 1.767$ N = **247380 N. Ans.**

Problem 21.8 *The total weight (including the self-weight of ram) placed on the sliding ram of a hydraulic accumulator is 40 kN. The diameter of the ram is 500 mm. If the frictional resistance against the movement of the ram is 5% of the total weight, determine the intensity of pressure of water when :*

(*i*) *The ram is moving up with a uniform velocity, and*

(*ii*) *The ram is moving down with uniform velocity.*

Solution. Given :

Total weight, W = 40 kN = 40 × 1000 = 40000 N

Dia. of ram, D = 500 mm = 0.50 m

$\therefore$ Area of ram, $A = \frac{\pi}{4} D^2 = \frac{\pi}{4} (.50)^2 = 0.1963 \text{ m}^2$

Frictional resistance against the movement of ram = 5% of total weight = $\frac{5}{100} \times 40000 = 2000$ N.

(*i*) *Intensity of pressure of water when ram is moving up with a uniform velocity.* When ram is moving up, the frictional resistance is acting opposite to the direction of movement of the ram, *i.e.*, frictional resistance is acting in the downward direction. Weight is also acting in the downward direction.

$\therefore$ Total force on the ram = Total weight + Frictional resistance
= 40000 + 2000 = 42000 N.

$\therefore$ Pressure intensity (p) $= \frac{\text{Total force on ram}}{\text{Area of ram}} = \frac{42000}{0.1963}$
$= 213958 \text{ N/m}^2 =$ **21.3958 N/cm^2. Ans.**

(*ii*) *Intensity of pressure when ram is moving down with a uniform velocity.* In this case, the frictional resistance is acting in the upward direction.

$\therefore$ Total force on the ram = Total weight – Frictional resistance
= 40000 – 2000 = 38000 N

$\therefore$ Pressure intensity (p) $= \frac{\text{Total force}}{\text{Area}} = \frac{38000}{0.1963}$
$= 193581 \text{ N/m}^2 =$ **19.3581 N/cm^2. Ans.**

Problem 21.9 *If in the problem 21.8, the stroke of the ram is 10 m and the ram falls through the full stroke in 4 minutes steadily, find the work done by the accumulator per second. If the pump, connected to the inlet of the accumulator, supplies .01 m^3/s at the same time, determine the work supplied by the pump per second and also power delivered by the accumulator to the hydraulic machine, connected at the outlet of the hydraulic accumulator, when ram is moving downwards.*

Solution. The data from problem 21.8, when ram is moving downward :

Total weight = 40000 N
Frictional resistance = 2000 N
Total force on ram = Total weight – Frictional resistance
= 40000 – 2000 = 38000 N
Area, $A = 0.1963 \text{ m}^2$
Pressure intensity of water when ram is moving downward
$p = 193581 \text{ N/m}^2$
Stroke of ram, $L = 10$ m
Time taken by ram to fall through full stroke, $t = 4$ minutes $= 4 \times 60 = 240$ s
Discharge supplied $Q = .01 \text{ m}^3/\text{s}$

Distance moved by ram in one second $= \frac{\text{Stroke of ram}}{\text{Time}} = \frac{L}{t} = \frac{10}{240} = \frac{1}{24}$ m/s.

(*i*) Work done by accumulator per second
= Total force on ram × Distance moved by ram per sec
$= 38000 \times \frac{1}{24} =$ **1583.33 Nm. Ans.**

(*ii*) Work supplied by the pump per sec = Weight of water supplied by pump per second × Head of supply pressure

$$= \rho g \times Q \times H = 1000 \times 9.81 \times .01 \times H \text{ Nm}$$

where H = Pressure head of water supplied $= \dfrac{p}{\rho \times g} = \dfrac{193581}{1000 \times 9.81} = 19.733$ m

$\therefore$ Work supplied by pump per sec = $1000 \times 9.81 \times 0.01 \times 19.733$ = **1935.81 Nm. Ans.**

(*iii*) Power delivered by the accumulator to the hydraulic machine connected at the outlet of accumulator

$$= \frac{1}{1000} \text{ (Work done by accumulator per second + Work supplied by pump per second)}$$

$$= \frac{1}{1000} (1583.33 + 1935.81) = \textbf{3.519 kW. Ans.}$$

Problem 21.10 *An accumulator is loaded with 40 kN weight. The ram has a diameter of 30 cm and stroke of 6 m. Its friction may be taken as 5%. It takes two min. to fall through its full stroke. Find the total work supplied and power delivered to the hydraulic appliance by the accumulator, when 7.5 lit/s is being delivered by a pump, while the accumulator descends with the stated velocity.*

Solution. Given :

Total weight = 40 kN = $40 \times 1000 = 40000$ N

Dia. of ram, $D = 30$ cm $= 0.3$ m

$\therefore$ Area of ram, $A = \dfrac{\pi}{4} (.3)^2 = 0.07068$ m^2

Stroke of ram, $L = 6$ m

Friction = 5%

$\therefore$ Net load on accumulator (when it descends)

$$= 40000 \times 0.95 = 38000 \text{ N}$$

Time taken by ram to fall through full stroke, $t = 2$ min $= 2 \times 60 = 120$ sec

$\therefore$ Distance moved by ram per sec $= \dfrac{L}{t} = \dfrac{6}{120} = \dfrac{1}{20}$ m/s

Water supplied by pump $= 7.5$ lit/s $= \dfrac{7.5}{1000}$ m^3/s $= .0075$ m^3/s

Work supplied by accumulator per second

= Net load on ram × Distance moved by ram per sec

$$= 38000 \times \frac{1}{20} = 1900 \text{ Nm/s}$$

Intensity of pressure of water, $p = \dfrac{\text{Net load}}{\text{Area}} = \dfrac{38000}{0.07068} = 542857 \dfrac{\text{N}}{\text{m}^2}$

Head due to pressure, $H = \dfrac{p}{\rho g} = \dfrac{542857}{1000 \times 9.81} = 55.337$ m

Work supplied by pump per second = Weight of water supplied per second × Head of supplied pressure

$$= \rho g \times Q \times H = 1000 \times 9.81 \times .0075 \times 55.337 = 4071.35 \text{ Nm/s}$$

$\therefore$ Total work supplied per second to hydraulic machine

= Work supplied by accumulator and by pump

= 1900 + 4071.35 Nm/s = **5971.35 Nm/s. Ans.**

Power delivered to the hydraulic machine

$$= \frac{\text{Total work supplied per sec}}{1000} = \frac{5971.35}{1000} = \textbf{5.9713 kW. Ans.}$$

Problem 21.11 *An accumulator has a ram of diameter 250 mm and a lift of 8 m. The total weight on accumulator is 70 kN. The packing friction is 5% of the load on the ram. Find the power delivered to the machine if ram falls through the full height in 100 sec and at the same time the pumps are delivering 0.028 m^3/s through the accumulator.*

Solution. Given :

Dia. of ram, $D = 250 \text{ mm} = 0.25 \text{ m}$

$\therefore$ Area of ram, $A = \frac{\pi}{4}(.25)^2 = \frac{\pi}{64} \text{ m}^2$

Lift of ram, $L = 8 \text{ m}$

Total weight $= 70 \text{ kN} = 70 \times 1000 = 70000 \text{ N}$

Packing friction = 5% of 70000 $= \frac{5}{100} \times 70000 = 3500 \text{ N}$

$\therefore$ Net load on accumulator, when the ram is moving downwards

$= 70000 - 3500 = 66500 \text{ N}$

Time taken by ram to fall through 8 m, $t = 100$ sec

Water supplied by pump, $Q = 0.028 \text{ m}^3/\text{s}$.

When ram is moving downwards, the pressure intensity (p) is given by,

$$p = \frac{\text{Net load}}{\text{Area}} = \frac{66500}{\left(\frac{\pi}{64}\right)} = \frac{66500 \times 64}{\pi} \text{ N/m}^2$$

Head corresponding to the above pressure intensity,

$$h = \frac{p}{\rho g} = \frac{66500 \times 4}{1000 \times 9.81 \times \pi} = 138.09 \text{ m of water.}$$

Power delivered by pump $= \frac{\rho g . Q . H}{1000} = \frac{1000 \times 9.81 \times 0.028 \times 138.09}{1000} = 37.931 \text{ kW}$

Power supplied by accumulator $= \frac{\text{Net load on ram} \times \text{Lift}}{1000 \times \text{Time}} = \frac{66500 \times 8}{1000 \times 100} = 5.32 \text{ kW.}$

$\therefore$ Total power $= 37.931 + 5.32 = \textbf{43.251 kW. Ans.}$

21.3.2 Differential Hydraulic Accumulator.

It is a device in which the liquid is stored at a high pressure by a comparatively small load on the ram. It consists of a fixed vertical cylinder of small diameter as shown in Fig. 21.5. The fixed vertical cylinder is surrounded by closely fitting brass bush, which is surrounded by an inverted moving cylinder, having circular projected collar at the base on which weights are placed.

The liquid from the pump is supplied to the fixed vertical cylinder. The liquid moves up through the small diameter of fixed vertical cylinder and then enters the inverted cylinder. The water exerts an upward pressure force on the internal annular area of the inverted moving cylinder, which is loaded at the base. The internal annular area of the inverted moving cylinder is equal to the sectional area of the brass bush. When the inverted moving cylinder moves up, the hydraulic energy is stored in the accumulator.

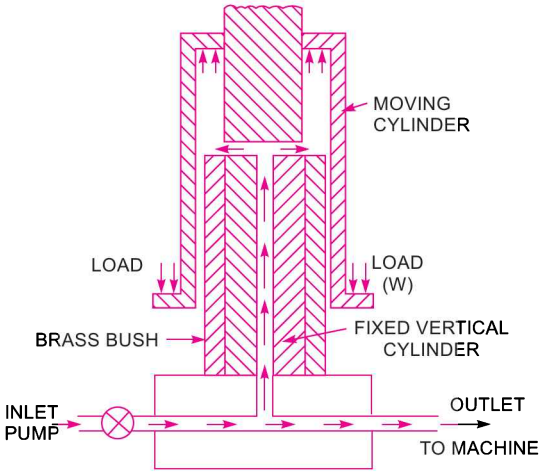


Fig. 21.5 *Differential hydraulic accumulator.*

Let p = Intensity of pressure of liquid supplied by pump,

a = Area of brass-bush,

L = Vertical lift of the moving cylinder,

W = Total weight placed on the moving cylinder including the weight of cylinder.

Then $W = p \times a$

$\therefore$ $$P = \frac{W}{a} \qquad ...(21.7)$$

From equation (21.7), it is clear that pressure intensity can be increased with a small load W, by making area 'a' small.

Now total energy stored in the accumulator = Total weight × Vertical lift

$$= W \times L \text{ Nm.} \qquad ...[21.7\ (a)]$$

▶ 21.4 THE HYDRAULIC INTENSIFIER

The device, which is used to increase the intensity of pressure of water by means of hydraulic energy available from a large amount of water at a low pressure, is called the hydraulic intensifier. Such a device is needed when the hydraulic machines such as hydraulic press requires water at very high pressure which cannot be obtained from the main supply directly.

A hydraulic intensifier consists of fixed ram through which the water, under a high pressure, flows to the machine. A hollow inverted sliding cylinder, containing water under high pressure, is mounted over the fixed ram. The inverted sliding cylinder is surrounded by another fixed inverted cylinder which contains water from the main supply at a low pressure as shown in Fig. 21.6

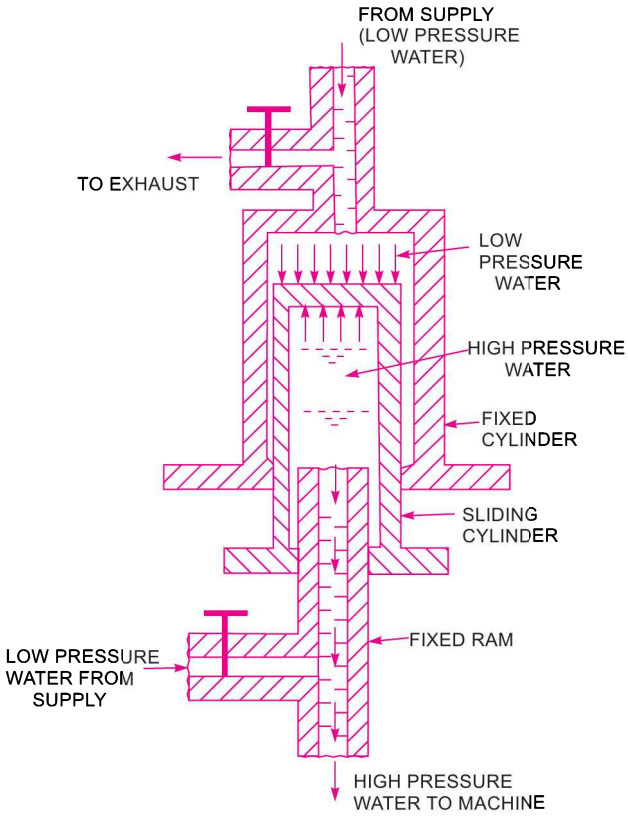


Fig. 21.6 *The hydraulic intensifier.*

A large quantity of water at low pressure from supply enters the inverted fixed cylinder. The weight of this water pressure the sliding cylinder in the downward direction. The water in the sliding cylinder gets compressed due to the downward movement of the sliding cylinder and its pressure is thus increased. The high pressure water is forced out of the sliding cylinder through the fixed ram, to the machine as shown in Fig. 21.6.

Let p = Intensity of pressure of water from supply to the fixed cylinder (low pressure water),

A = External area of the sliding cylinder,

a = Area of the end of the fixed ram, and

p^* = Intensity of the pressure of water in the sliding cylinder (high pressure water).

The force exerted by low pressure water on the sliding cylinder in the downward direction

$$= p \times A.$$

The force exerted by the high pressure water on the sliding cylinder in the upward direction

$$= p^* \times a.$$

Equating the upward and downward forces,

$$p \times A = p^* \times a.$$

$$p^* = \frac{p \times A}{a}. \qquad \text{...(21.8)}$$

Problem 21.12 *The diameters of fixed ram and fixed cylinder of an intensifier are 8 cm and 20 cm respectively. If the pressure of the water supplied to the fixed cylinder is 300 N/cm^2, find the pressure of the water flowing through the fixed ram.*

Solution. Given :

Dia. of fixed ram, $d = 8$ cm

$\therefore$ Area of fixed ram, $a = \frac{\pi}{4} d^2 = \frac{\pi}{4} \times 8^2 = 16\ \pi\ \text{cm}^2$

Dia. of fixed cylinder, $D = 20$ m

$\therefore$ Area of fixed cylinder, $A = \frac{\pi}{4} \times 20^2 = 100\ \pi\ \text{cm}^2$

Intensity of supply pressure, $p = 300\ \text{N/cm}^2$

Let the intensity of pressure of water flowing through fixed ram $= p^*$

Using equation (21.8), $p^* = \frac{p \times A}{a} = \frac{300 \times 100\pi}{16\pi} = \mathbf{1875\ N/cm^2.\ Ans.}$

Problem 21.13 *The pressure intensity of water supplied to an intensifier is 20 N/cm^2 while the pressure intensity of water leaving the intensifier is 100 N /cm^2. The external diameter of the sliding cylinder is 20 cm. Find the diameter of the fixed ram of the intensifier.*

Solution. Given :

Supply pressure, $p = 20\ \text{N/cm}^2$

Intensity of pressure leaving the intensifier, $p^* = 100\ \text{N/cm}^2$

External dia. of sliding cylinder, $D = 20$ cm

$\therefore$ Area of sliding cylinder, $A = \frac{\pi}{4} \times 20^2 = 100\ \pi\ \text{cm}^2$

Let the dia. of the fixed ram $= d$

$\therefore$ Area of the fixed ram, $a = \frac{\pi}{4} d^2$

Using equation (21.8), $p^* = \frac{p \times A}{a}$

$$100 = \frac{20 \times 100\pi}{\frac{\pi}{4} d^2} = \frac{20 \times 100 \times 4}{d^2}$$

$\therefore$ $$d = \sqrt{\frac{20 \times 100 \times 4}{100}} = \sqrt{80} = \mathbf{8.94\ cm.\ Ans.}$$

▶ 21.5 THE HYDRAULIC RAM

The hydraulic ram is a pump which raises water without any external power for its operation. When large quantity of water is available at a small height, a small quantity of water can be raised to a greater height with the help of hydraulic ram. It works on the principle of water hammer.

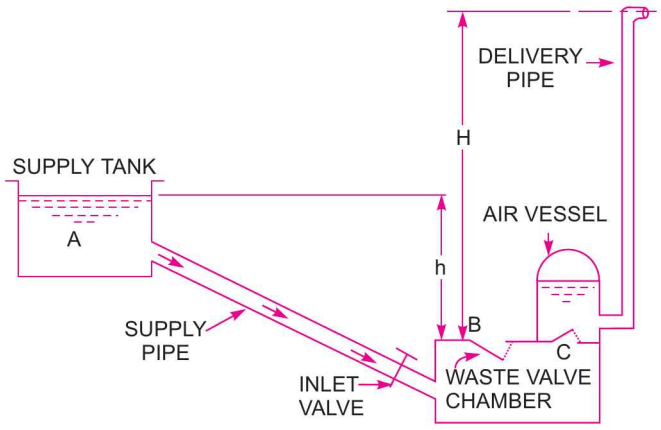


Fig. 21.7 *The hydraulic ram.*

Fig. 21.7 shows the main components of the hydraulic ram. When the inlet valve fitted to the supply pipe is opened, water starts flowing from the supply tank to the chamber, which has two valves at *B* and *C*. The valve *B* is called waste valve and valve *C* is called the delivery valve. The valve *C* is fitted to an air vessel. As the water is coming into the chamber from supply tank, the level of water rises in the chamber and waste valve *B* starts moving upward. *A* stage comes, when the waste valve *B* suddenly closes. This sudden closure of waste valve creates high pressure inside the chamber. This high pressure force opens the delivery valve *C*. The water from chamber enters the air vessel and compresses the air inside the air vessel. This compressed air exerts force on the water in the air vessel and small quantity of water is raised to a greater height as shown in Fig. 21.7.

When the water in the chamber loses its momentum, the waste valve *B* opens in the downward direction and the flow of water from supply tank starts flowing to the chamber and the cycle will be repeated.

Let W = Weight of water flowing per second into chamber,

w = Weight of water raised per second,

h = Height of water in supply tank above the chamber,

H = Height of water raised from the chamber.

The energy supplied by the supply tank to ram

= Weight of water supplied × Height of supply water

$= W \times h$ ...(*i*)

Energy delivered by the ram = Weight of water raised × Height through which water is raised

$= w \times H$ ...(*ii*)

∴ Efficiency of the hydraulic ram,

$$\eta = \frac{\text{Energy delivered by the ram}}{\text{Energy supplied to the ram}} = \frac{w \times H}{W \times h} \qquad ...(21.9)$$

The above expression of efficiency was given by D' Aubuisson and hence known as *D' Aubuisson's efficiency*.

Rankine gave another form of the above efficiency. According to him, the weight of water (w) is raised to a height of ($H - h$) and not H. The water is initially at a height of h from the ram and hence the water is only raised to a height equal to ($H - h$). Hence according to Rankine :

Energy delivered by the ram $= w \times (H - h)$

Energy supplied $= (W - w)\, h$

$\therefore$ Efficiency, $$\eta = \frac{w \times (H - h)}{(W - w) \times h} \qquad ...(21.10)$$

Equation (21.10) is known as *Rankine's efficiency.*

The above two efficiencies, in terms of discharge is written as,

D' Aubuisson's $$\eta = \frac{q \times H}{Q \times h} \qquad ...(21.11)$$

Rankine's $$\eta = \frac{q(H - h)}{(Q - q) \times h} \qquad ...(21.12)$$

where q = Discharge of delivery pipe,

Q = Discharge through supply pipe.

Problem 21.14 *The water is supplied at the rate of 0.02 m^3 per second from a height of 3 m to a hydraulic ram, which raises 0.002 m^3/s to a height of 20 m from the ram. Determine D' Aubuisson's and Rankine's efficiencies of the hydraulic ram.*

Solution. Given :

Discharge through supply pipe, $Q = 0.02$ m^3/s

Supply head, $h = 3$ m

Discharge raised, $q = 0.002$ m^3/s

Height of water raised from hydraulic ram, $H = 20$ m

Using equation (21.11),

D' Aubuisson's $$\eta = \frac{q \times H}{Q \times h} = \frac{.002 \times 20}{.02 \times 3} = .6667 = \mathbf{66.67\%.\ Ans.}$$

Rankine's efficiency is given by equation (21.12) as

Rankine's $$\eta = \frac{q(H - h)}{(Q - q) \times h}$$

$$= \frac{0.002 \times (20 - 3)}{(0.020 - .0002) \times 3} = \frac{0.002 \times 17}{.018 \times 3} = 0.6296 = \mathbf{62.96\%.\ Ans.}$$

Problem 21.15 *The water is supplied at the rate of 3000 litres per minute from a height of 4 m to a hydraulic ram, which raises 300 litres/minute to a height of 30 m from the ram. The length and diameter of the delivery pipe is 100 m and 70 mm respectively. Calculate the efficiency of the hydraulic ram if the co-efficiency of friction f = .009.*

Solution. Given :

Discharge supplied, $Q = 3000$ litres/minute

$$= \frac{3000}{60} \text{ lit/s} = \frac{3000}{60 \times 1000} \text{m}^3\text{/s} = 0.05 \text{ m}^3\text{/s}$$

Supply head, $h = 4$ m

Discharge raised, $q = 300$ lit/min $= \dfrac{0.3}{60} = .005$ m^3/s ($\because$ 300 lit = 0.3 m^3)

Height of water raised from hydraulic ram, $H = 30$ m

Length of delivery pipe, $L = 100$ m

Dia. of delivery pipe, $d = 70$ mm $= .07$ m

Co-efficient of friction, $f = .009$

Head lost due to friction in delivery pipe is

$$h_f = \frac{4fLV^2}{d \times 2g} = \frac{4 \times .009 \times 100 \times V^2}{.07 \times 2 \times 9.81} \quad ...(i)$$

But V = Velocity of water in delivery pipe

$$= \frac{\text{Discharge in delivery pipe}}{\text{Area}}$$

$$= \frac{q}{\frac{\pi}{4}d^2} = \frac{0.005}{\frac{\pi}{4} \times (.07)^2} = 1.299 \simeq 1.3 \text{ m/s.}$$

Substituting this value of V in equation (i), we get

$$h_f = \frac{4 \times .009 \times 100 \times (1.3)}{.07 \times 2 \times 9.81} = 4.43 \text{ m}$$

$\therefore$ Effective head developed by the ram

$$= H + h_f = 30 + 4.43 = 34.43 \text{ m}$$

D' Aubuisson's efficiency is given by equation (21.11), as

$$\eta = \frac{q \times \text{Effective head}}{Q \times h} \quad \text{(Here } H \text{ = Effective head)}$$

$$= \frac{.005 \times 34.43}{0.05 \times 4} = 0.8607 = \mathbf{86.07\%. \text{ Ans.}}$$

Rankine's efficiency is given by equation (21.12) as

$$\eta = \frac{q(\text{Effective head} - h)}{(Q - q) \times h}$$

$$= \frac{.005\,(34.43 - 4.0)}{(.05 - .005) \times 4.0} = \frac{.005 \times 30.43}{.045 \times 4.0} = 0.8453 = \mathbf{84.53\%. \text{ Ans.}}$$

▶ 21.6 THE HYDRAULIC LIFT

The hydraulic lift is a device used for carrying passenger or goods from one floor to another in multi-storeyed building. The hydraulic lifts are of two types, namely,

1. Direct acting hydraulic lift, and 2. Suspended hydraulic lift.

21.6.1 Direct Acting Hydraulic Lift. It consists of a ram, sliding in fixed cylinder as shown in Fig. 21.8. At the top of the sliding ram, a cage (on which the persons may stand or goods may be placed) is fitted. The liquid under pressure flows into the fixed cylinder. This liquid exerts force on the sliding ram, which moves vertically up and thus raises the cage to the required height.

The cage is moved in the downward direction, by removing the liquid from the fixed cylinder.

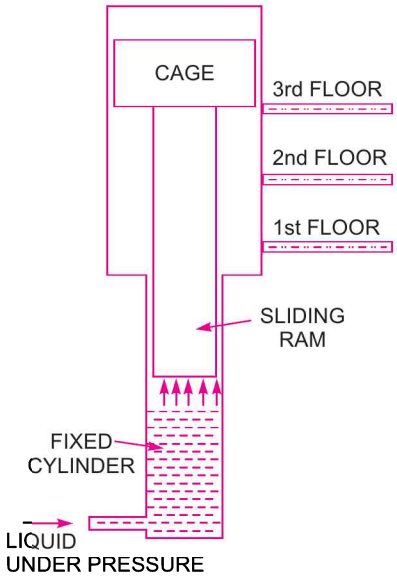


Fig. 21.8 *Suspended hydraulic lift.*

21.6.2 Suspended Hydraulic Lift. Fig. 21.9 shows the suspended hydraulic lift. It is a modified form of the direct acting hydraulic lift. It consists of a cage (on which persons may stand or goods may be placed) which is suspended from a wire rope. A jigger, consisting of a fixed cylinder, a sliding ram and a set of two pulley blocks, is provided at the foot of the hole of the cage. One of the pulley block is movable and the other is a fixed one. The end of the sliding ram is connected to the movable pulley block. A wire rope, one end of which is fixed at *A* and the other end is taken round all the pulleys of the movable and fixed blocks and finally over the guide pulleys as shown in Fig. 21.9. The cage is suspended from the other end of the rope. The raising or lowering of the cage of the lift is done by the jigger as explained below.

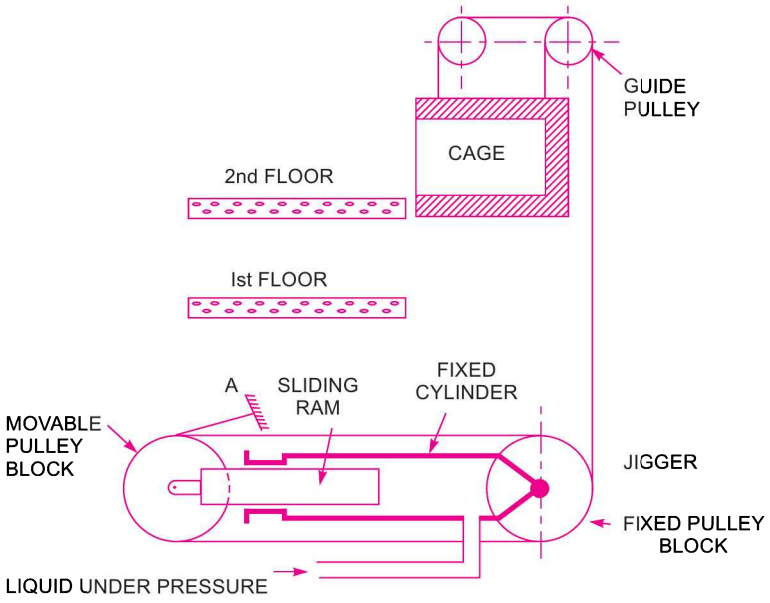


Fig. 21.9 *Suspended hydraulic lift.*

When water under high pressure is admitted into the fixed cylinder of the jigger, the sliding ram is forced to move towards left. As one end of the sliding ram is connected to the movable pulley block and hence the movable pulley block moves towards the left, thus increasing the distance between two

pulley blocks. The wire rope connected to the cage is pulled and the cage is lifted. For lowering the cage, water from the fixed cylinder is taken out. The sliding ram moves towards right and hence movable pulley blocks also moves towards right. This decreases the distance between two pulley blocks and the cage is lowered due to increased length of the rope.

Problem 21.16 *A hydraulic lift is required to lift a load of 8 kN through a height of 10 metres, once in every 80 seconds. The speed of the lift is 0.5 m per second. Determine :*

(i) Power required to drive the lift, (ii) Working period of lift in seconds, and

(iii) Idle period of the lift in seconds.

Solution. Given :

Load lifted, $W = 8 \text{ kN} = 8 \times 1000 = 8000 \text{ N}$

Height, $H = 10 \text{ m}$

Time for one operation, $t = 80 \text{ s}$

Speed of lift, $v = 0.5 \text{ m/s}$.

(*i*) Work done in lifting the load in 80 seconds

$$= W \times H = 8000 \times 10 = 80000 \text{ Nm}$$

$\therefore$ Work done/s $= \dfrac{80000}{80} = 1000 \text{ Nm/s}$

$\therefore$ Power required to drive the lift $= \dfrac{1}{1000} \times \text{Work done/s} = \dfrac{1}{1000} \times 1000 =$ **1.0 kW. Ans.**

(*ii*) Working period of the lift $= \dfrac{\text{Height of the lift}}{\text{Velocity of lift}} = \dfrac{10}{0.50} =$ **20 sec. Ans.**

(*iii*) Idle period of the lift = Total time – Working period of lift

$= 80 - 20 =$ **60 sec. Ans.**

Problem 21.17 *A hydraulic lift is required to lift a load of 12 kN through a height of 10 m, once in every 1.75 minutes. The speed of the lift is 0.75 m/s. During working stroke of the lift, water from accumulator and the pump at a pressure of 400 N/cm^2 is supplied to the lift. If the efficiency of the pump is 8% and that of lift is 75%, find the power required to drive the pump and the minimum capacity of the accumulator. Neglect friction losses in the pipe.*

Solution. Given :

Load lifted, $W = 12 \text{ kN} = 12 \times 1000 = 12000 \text{ N}$

Height, $H = 10 \text{ m}$

Total time for one operation, $t = 1.75 \text{ min} = 1.75 \times 60 = 105 \text{ s}$

Speed of lift, $v = 0.75 \text{ m/s}$

Water pressure from accumulator and pump,

$$p = 400 \text{ N/cm}^2 = 400 \times 10^4 \text{ N/m}^2$$

Efficiency of pump, $\eta_p = 80\% = 0.80$

Efficiency of lift, $\eta_l = 75\% = 0.75$

Work done by water (supplied from accumulator and pump) in raising lift per second

$$= \text{Load lifted} \times \text{Distance travelled per s}$$

$$= W \times \text{Velocity of lift}$$

$$= W \times v = 12000 \times 0.75 = 9000 \text{ Nm/s}$$

$\therefore$ Useful power $= \dfrac{\text{Work done per second}}{1000} = \dfrac{9000}{1000} = 9 \text{ kW}$

$\therefore$ Actual power supplied to lift $= \dfrac{\text{Useful horse power}}{\eta_l} = \dfrac{9.0}{0.75} = 12$ kW.

The power 12 kW has been supplied by the pump and by accumulator to the lift.

Let P_1 = Output of the pump in kW

Then $(12 - P_1)$ = Output of the accumulator in kW ...(*i*)

Now, working period of the lift $= \dfrac{\text{Height of lift}}{\text{Velocity of lift}} = \dfrac{H}{v} = \dfrac{10}{0.75} = 13.33$ s

Idle period of lift = Total time – Working period of lift

= 105 – 13.33 = 91.67 s

Thus during idle period of lift, the energy will be stored in the accumulator and during working period of lift of 13.33 s, the energy will be supplied by the accumulator to the lift.

$\therefore$ Energy stored during idle period in accumulator

= Output of pump × Idle period

$= (P_1 \times 1000) \times 91.67$ Nm/s ($\because$ 1 kW = 1000 Nm/s) ...(*ii*)

The above energy is supplied by accumulator in 13.33 seconds.

$\therefore$ Energy supplied by accumulator per second

$$= \frac{(P_1 \times 1000) \times 91.67}{13.33} \text{ Nm/s}$$

$\therefore$ Power supplied by accumulator $= \dfrac{1}{1000}$ [Energy supplied by accumulator per second]

$$= \frac{1}{1000}\left[\frac{P_1 \times 1000 \times 91.67}{13.33}\right] = 6.877\, P_1.$$

But from equation (*i*), power supplied by accumulator is also

$= (12 - P_1)$

$\therefore$ Equating the two values of power supplied by accumulator,

$6.877\, P_1 = 12 - P_1$ or $6.877\, P_1 + P_1 = 12$ or $7.877\, P_1 = 12$

$\therefore$ $P_1 = \dfrac{12}{7.877} = 1.523.$

But we have assumed P_1 as the output of the pump.

$\therefore$ Input to the pump $= \dfrac{\text{Output}}{\text{Efficiency of pump}} = \dfrac{P_1}{\eta_p} = \dfrac{1.523}{0.80} = 1.90375$ kW.

(*i*) $\therefore$ Power required to drive the pump = **1.90375 kW. Ans.**

(*ii*) Minimum capacity of the accumulator.

From equation (*ii*), the energy stored in the accumulator

$= P_1 \times 1000 \times 91.67$ Nm

$= 1.523 \times 1000 \times 91.67$ Nm ($\because$ $P_1 = 1.523$)

= 139613.4 Nm.

The capacity of the accumulator means the amount of hydraulic energy stored in the accumulator.

$\therefore$ Minimum capacity of accumulator = Energy stored = **139613.4 Nm. Ans.**

▶ 21.7 THE HYDRAULIC CRANE

Hydraulic crane is a device, used for raising or transferring heavy loads. It is widely used in workshops, warehouses and dock sidings.

A hydraulic crane consists of a mast, tie, jib, guide pulley and a jigger. The jib and tie are attached to the mast. The jib can be raised or lowered in order to decrease or increase the radius of action of the crane. The mast along with the jib can revolve about a vertical axis and thus the load attached to the rope can be transferred to any place within the area of the crane's action. The jigger, which consists of a movable ram sliding in a fixed cylinder, is used for lifting or lowering the heavy loads. One end of the ram is in contact with water and the other end is connected to set of movable pulley block. Another pulley block, called the fixed pulley block is attached to the fixed cylinder. The pulley block, attached to the ram, moves up and down while the pulley block, attached to the fixed cylinder, is not having any movement.

A wire rope, one end of which is fixed to a movable pulley (which is attached to the sliding ram) is taken round all the pulleys of the two sets of the pulleys and finally passes over the guide pulley, attached to the jib as shown in Fig. 21.10. The other end of the rope is provided with a hook, for suspending the load.

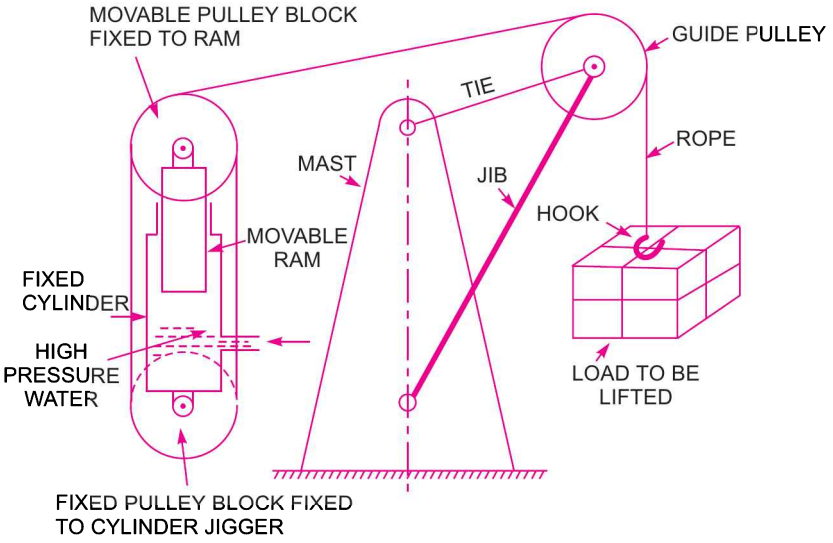


Fig. 21.10 *The hydraulic crane.*

For lifting the load by the crane, the water under high pressure is admitted into the cylinder of the jigger. This water forces the sliding ram to move vertically up. Due to the movement of the ram in the vertically up direction, the movable pulley block attached to the ram also moves upward. This increases the distance between two pulley blocks and hence the wire passing over the guide pulley is pulled by the jigger. This raises the load attached to the hook.

Problem 21.18 *Find the efficiency of a hydraulic crane, which is supplied 300 litres of water under a pressure of 60 N/cm^2 for lifting a weight of 12 kN through a height of 11 m.*

Solution. Given :

Water supplied, $Q = 300 \text{ litres} = 0.30 \text{ m}^3$

Pressure, $p = 60 \text{ N/cm}^2 = 60 \times 10^4 \frac{\text{N}}{\text{m}^2}$

Weight lifted, $W = 12 \text{ kN} = 12 \times 1000 = 12000 \text{ N}$

Height, $h = 11 \text{ m}$

Output of the crane = Weight lifted × Height through which weight is lifted

$= W \times h = 12000 \times 11 \text{ Nm}$

Input of the crane = Energy supplied by the water

= Work done by water on the ram

= Force on ram × Distance moved by ram

= Pressure × Area of ram × Stroke of ram

$= p \times A \times L$

$= 60 \times 10^4 \times$ Volume displaced

$= 60 \times 10^4 \times 0.30$ $(\because \quad A \times L = Q)$

$= 18 \times 10^4 \text{ Nm}$

$\therefore$ Efficiency of the crane $= \dfrac{\text{Output}}{\text{Input}} = \dfrac{12000 \times 11}{18 \times 10^4} = 0.7333 =$ **73.33%. Ans.**

Problem 21.19 *The efficiency of a hydraulic crane, which is supplied water under a pressure of 70 N/cm^2 for lifting a weight through a height of 10 m, is 60%. If the diameter of the ram is 150 mm and velocity ratio is 6, find*

(*i*) *the weight lifted by the crane, and*

(*ii*) *the volume of water required in litres to lift the weight.*

Solution. Given :

Efficiency, $\eta = 60\% = 0.60$

Pressure of water, $p = 70 \text{ N/cm}^2 = 70 \times 10^4 \text{ N/m}^2$

Height through which weight is lifted, $h = 10 \text{ m}$

Diameter of the ram, $D = 150 \text{ mm} = 0.15 \text{ m}$

$\therefore$ Area of ram, $A = \dfrac{\pi}{4} (.15)^2 = 0.01767 \text{ m}^2$

Velocity ratio = 6

Pressure force on ram, P = Pressure × Area of ram

$= p \times A = 70 \times 10^4 \times 0.01767 = 12369 \text{ N}.$

(*i*) We know efficiency of the hydraulic crane is given as

$$\eta = \frac{\text{Output}}{\text{Input}} = \frac{\text{Weight} \times \text{Distance moved by weight}}{\text{Force} \times \text{Distance moved by force}}$$

or $$0.60 = \frac{W \times \text{Distance moved by weight}}{P \times \text{Distance moved by force}}$$

But $\dfrac{\text{Distance moved by weight}}{\text{Distance moved by force}} = \text{Velocity ratio} = 6$

$\therefore$ $$0.60 = \frac{W}{P} \times 6 = \frac{W}{12369} \times 6$$

$\therefore$ $$W = \frac{0.60 \times 12369}{6} = \textbf{1236.9 N. Ans.}$$

(*ii*) Volume of water required to lift the weight :

$$\text{Velocity ratio} = \frac{\text{Distance moved by weight}}{\text{Distance moved by force on ram}}$$

or $$6 = \frac{h}{\text{Stroke of ram}} \quad (\because \text{ Distance moved by ram = Stroke of ram})$$

$\therefore$ Stroke of ram, $L = \frac{h}{6} = \frac{10}{6} = 1.667$ m

$\therefore$ Volume of water = Area of ram × Stroke of ram = $A \times L = 0.01767 \times 1.667$

$= 0.02945 \text{ m}^3 = 0.02945 \times 1000 =$ **29.45 litres. Ans.**

Problem 21.20 *A hydraulic crane is lifting a weight of 12000 N through a height of 12 m with a speed of 18 m per minute once in every two minutes. The efficiency of the hydraulic crane is 65% and it is working under a pressure of 500 N/cm^2 of water. The crane is fed from an accumulator to which water is supplied by a pump. Find :*

(*i*) *the capacity of the cylinder of the jigger in litres,*

(*ii*) *the capacity of the accumulator in litres, and*

(*iii*) *minimum power required for the pump.*

Solution. Given :

Weight lifted, $W = 12000$ N

Height, $h = 12$ m

Speed of weight, $V = 18$ m/min

No. of times the weight if lifted = Once in every two minutes

Efficiency, $\eta = 65\% = 0.65$

Pressure of water, $p = 500 \text{ N/cm}^2 = 500 \times 10^4 \text{ N/m}^2$

(*i*) Output of the crane = Weight lifted × Height

$= W \times h = 12000 \times 12 = 144000$ Nm.

Input of the crane = Work done by water on ram

= Force on ram × Distance moved by ram

$= p \times A \times L$

$= p \times$ Volume of cylinder $(\because \ A \times L =$ Volume of cylinder)

$= 500 \times 10^4 \times$ Volume of cylinder

$\therefore \quad \eta = \frac{\text{Output}}{\text{Input}} = \frac{144000}{500 \times 10^4 \times \text{volume of cylinder}}$

$\therefore$ Volume of cylinder $= \frac{144000}{500 \times 10^4 \times \eta} = \frac{144000}{500 \times 10^4 \times 0.65} = 0.0443 \text{ m}^3 = 44.3$ litres.

$\therefore$ Capacity of the cylinder of the jigger = **44.3 litres. Ans.**

(*ii*) Input of the crane $= p \times$ Volume of cylinder $= 500 \times 10^4 \times 0.0443 = 221500$ Nm.

This input is given to the crane once in every two minutes.

$\therefore$ Input to crane per min. $= \frac{221500}{2} = 110750$ Nm.

The weight 12000 N is lifted to a height of 12 m with a speed of 18 m/min.

Time required to lift the weight through height of 12 m $= \frac{\text{Height}}{\text{Speed}} = \frac{12}{18} = \frac{2}{3}$ min.

$\therefore$ Work done by the pump during lifting

= Work done per min. × Time required to lift the weight

$= 110750 \times \frac{2}{3} = 73833.33$ Nm

$\therefore$ Energy supplied by accumulator = Total input energy to the crane – Work done during lifting

$= 221500 - 73833.33 = 147666.67$ Nm ...(*i*)

But energy supplied by accumulator = Force on the ram of accumulator × Lift of ram

$= p \times A \times H = p \times$ Capacity of accumulator

$= 500 \times 10^4 \times$ Capacity of accumulator ...(*ii*)

Equating the two values given by equations (*i*) and (*ii*),

$147666.67 = 500 \times 10^4 \times$ Capacity of accumulator

$\therefore$ Capacity of accumulator $= \frac{147666.67}{500 \times 10^4} = 0.0295 \text{ m}^3 =$ **29.5 litres. Ans.**

(*iii*) Minimum power required for the pump

$= \frac{\text{Work input per minute}}{1000 \times 60} = \frac{110750}{1000 \times 60} =$ **1.846 kW. Ans.**

▶ 21.8 THE FLUID OR HYDRAULIC COUPLING

The fluid or hydraulic coupling is a device used for transmitting power from driving shaft to driven shaft with the help of fluid (generally oil). There is no mechanical connection between the two shafts. It consists of a radial pump impeller mounted on a driving shaft *A* and a radial flow reaction turbine mounted on the driven shaft *B*. Both the impeller and runner are identical in shape and they together form a casing which is completely enclosed and filled with oil.

In the beginning, both the shafts *A* and *B* are at rest. When the driving shaft *A* is rotated, the oil starts moving from the inner radius to the outer radius of the pump impeller as shown in Fig. 21.11. The pressure energy and kinetic energy of the oil increases at the outer radius of the pump impeller. This oil of increased energy enters the runner of the reaction turbine at the outer radius of the turbine runner and flows inwardly to the inner radius of the turbine runner. The oil, while flowing through the runner, transfers its energy to the blades of the runner and makes the runner to rotate. The oil, from the runner then flows back into the pump impeller, thus having a continuous circulation.

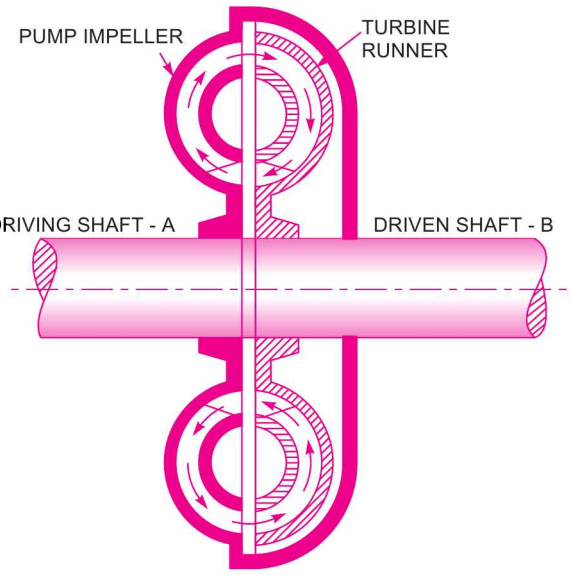


Fig. 21.11 *The hydraulic coupling.*

The power is transmitted hydraulically from the driving shaft to driven shaft and the driven shaft is free from engine vibrations. The speed of the driven shaft B is always less than the speed of the shaft A, by about 2 per cent. The efficiency of the power transmission by hydraulic coupling is about 98%. This is derived as given below.

$$\text{Efficiency of a fluid coupling} = \frac{\text{Power output}}{\text{Power input}}$$

or
$$\eta = \frac{\text{Power transmitted to shaft } B}{\text{Power available at shaft } A} \qquad ...(i)$$

But power at any shaft $= \dfrac{2\pi NT}{60{,}000} \propto NT \propto \text{Speed} \times \text{Torque}$

Let N_A = Speed of shaft A, ...(*ii*)

T_A = Torque at the shaft A,

N_B = Speed of shaft B,

T_B = Torque transmitted to shaft B.

From equation (*ii*), we have

Power available to shaft $A \propto$ (Speed of shaft A) $\times$ Torque of A

$$\propto N_A \times T_A.$$

Similarly, power transmitted to shaft $B \propto N_B \times T_B$.

Substituting these values of powers in equation (*ii*),

$$\eta = \frac{N_B \times T_B}{N_A \times T_A}$$

But $T_A = T_B$ ($\because$ Torque transmitted is same)

$$\therefore \quad \eta = \frac{N_B}{N_A} \qquad ...(21.13)$$

Slip of fluid coupling is defined as the ratio of the difference of the speeds of the driving and driven shaft to the speed of the driving sharp. Mathematically,

Slip, $$S = \frac{N_A - N_B}{N_A} = 1 - \frac{N_B}{N_A} = 1 - \eta \qquad \left(\because \frac{N_B}{N_A} = \eta\right) \qquad ...(21.14)$$

▶ 21.9 THE HYDRAULIC TORQUE CONVERTER

The hydraulic torque converter is a device used for transmitting increased torque at the driven shaft. The torque transmitted at the driven shaft may be more or less than the torque available at the driving shaft. The torque at the driven shaft may be increased about five times the torque available at the driving shaft with an efficiency of about 90%.

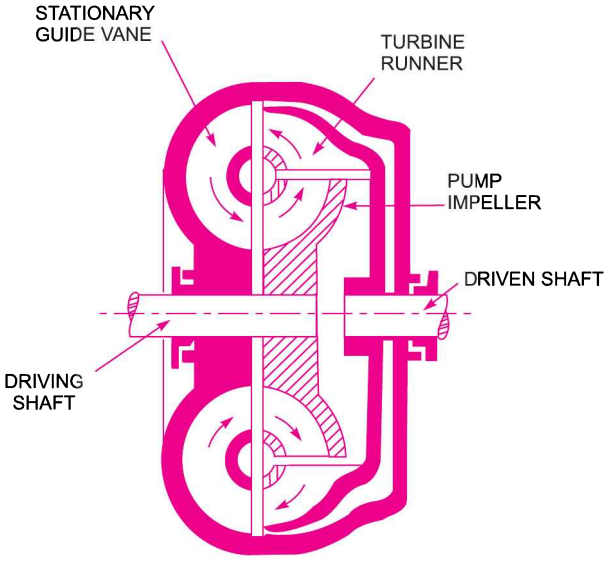


Fig. 21.12 *Fluid of hydraulic torque converter.*

As mentioned in Art. 21.8, the power at any shaft is proportional to the product of torque and speed of the shaft. Hence, if the torque at the driven shaft is to be increased, the corresponding value of the speed at the same shaft should be decreased. The speed of the driven shaft is decreased by decreasing the velocity of oil, which is allowed to flow from the pump impeller to the turbine runner and then through stationary guide vanes as shown in Fig. 21.12. Due to the decrease in speed at the driven shaft, the torque increases.

▶ 21.10 THE AIR LIFT PUMP

The air lift pump is a device which is used for lifting water from a well or sump by using compressed air. The compressed air is made to mix with the water. The density of the mixture of air and water is reduced. The density of this mixture is much less than that of pure water. Hence a very small column of pure water can balance a very long column of air water mixture. This is the principle on which the air lift pump works.

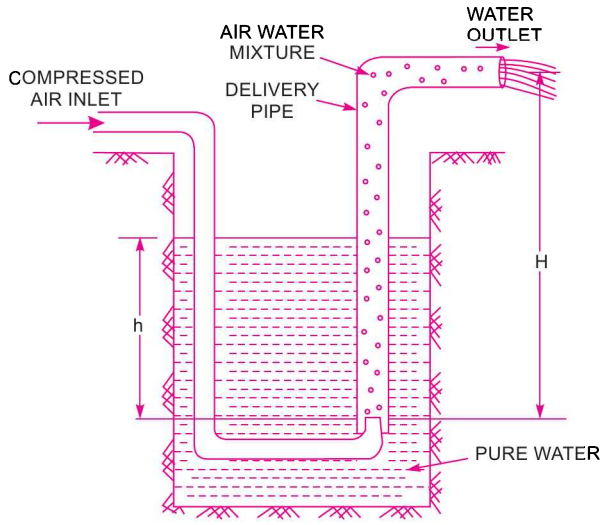


Fig. 21.13 *Air lift pump.*

Fig. 21.13 shows the air lift pump. The compressed air is introduced through one or more nozzles at the foot of the delivery pipe, which is fixed in the well from which water is to be lifted. In the delivery pipe, a mixture of air and water is formed. The density of this air water mixture becomes very less as compared to the density of pure water. Hence, a small column of pure water will balance a very long column of air water mixture. This air water mixture will be discharged out of the delivery pipe. The flow will continue as long as there is supply of compressed air.

Let h = Height of static water level above the tip of the nozzle,

H = Height to which water is lifted above the tip of the nozzle.

The $(H - h)$ is known as the useful lift. The best results are obtained if the useful lift $(H - h)$ is less than the height of static water (h) above the tip of the nozzle. Hence for best results, $(H - h)$ should be less than h.

The ratio $\left(\frac{h}{H-h}\right)$ generally varies from 4 to 1.

When $h = 30$ m, the ratio $\left(\frac{h}{H-h}\right)$ is about 4.

When $h = 90$ m, the ratio $\left(\frac{h}{H-h}\right)$ is 1.

For $h = 30$ m, $\frac{h}{(H-h)} = 4$ or $\frac{30}{(H-30)} = 4$ or $30 = 4H - 120$

or $30 + 120 = 4H$ or $H = \frac{150}{4} = 37.5$ m.

The air lift pump is not having any moving parts below water level and hence there are no chances of suspended solid particles damaging the pump. This is the main advantage of this pump. Also this pump can raise more water through a bore hole of given diameter than any other pump. But the efficiency of this pump is low as out of the energy expended in compressing the air, only 20 to 40% energy appears in the form of useful water horse-power.

▶ 21.11 THE GEAR-WHEEL PUMP

The gear pump is a rotary pump in which two gears mesh to provide the pumping action. This type of pump is mostly used for cooling water and pressure oil to be supplied for lubrication to motors, turbines, machine tools etc. Although the gear pump is a rotating machinery, yet its action on liquid to be pumped is not dynamic and it merely displaces the liquid from one side to the other. The flow of liquid to be pumped is continuous and uniform.

Fig. 21.14 shows the gear pump, which consists of two identical intermeshing gears working in a fine clearance inside a casing. One of the gear is keyed to a driving shaft. The other gear revolves due to driving gear. The space between teeth and the casing is filled with oil. The oil is carried round between the gears from the suction pipe to the delivery pipe.

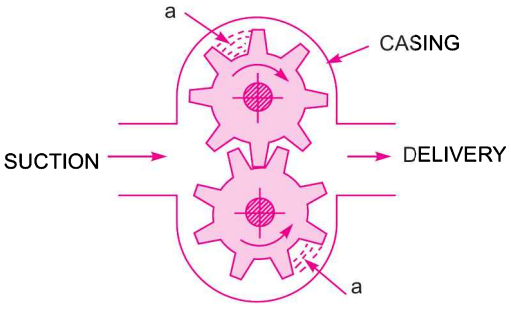


Fig. 21.14 *Gear-wheel pump.*

The mechanical contact between the gears does not allow the flow from inlet to outlet directly. The outer radial tips of the gears and sides of the gears form a part off moving oil.

The oil pushed into the delivery pipe, cannot back into the suction pipe due to the meshing of the gears. The theoretical oil pumped per second is obtained as :

Let N = Speed of rotating gear in r.p.m.,
a = Area enclosed between two successive teeth and casing,
n = Total number of teeth in each gear,
L = Axial length of teeth.

Volume of oil discharged per revolution = $2 \times a \times L \times N$ m^3

$\therefore$ Discharge/s = Volume of oil per revolution $\times$ No. of revolution in one second

$$= 2aLn \times \frac{N}{60} \text{ m}^3$$

The actual discharge will be less than the theoretical discharge.

Now, volumetric efficiency $= \dfrac{\text{Actual discharge}}{\text{Theoretical discharge}}$.

HIGHLIGHTS

1. The devices, based on the principles of fluid statics and fluid kinematics for the transmission of power with the help of a fluid (oil or air), are called fluid or hydraulic devices.
2. A device, which is used for lifting heavy weights by the application of a much smaller force is known as hydraulic press.
3. The device, used for storing the energy of a liquid in the form of pressure energy which may be supplied for any sudden or intermittent requirement, is known as hydraulic accumulator.
4. Capacity of the hydraulic accumulator is given as
$$= p \times A \times L$$
where p = Liquid pressure supplied by pump, A = Area of the sliding ram, L = Stroke or lift of the ram.
5. Differential hydraulic accumulator is a device in which liquid is stored at a high pressure by comparatively small load on the ram.
6. Hydraulic intensifier is a device, in which the pressure intensity of a liquid is increased by means of hydraulic energy available from a large amount of liquid at a low pressure. The increased pressure intensity (p^*) is given by the relation,
$$p^* = \frac{p \times A}{a}$$
where p = Low pressure intensity of liquid, A = External area of the sliding ram,
a = Area of the end of the fixed ram.
7. Hydraulic ram is a pump which raises water without any external power (electricity etc.) for its operation. There are two efficiencies of a hydraulic ram namely D' Aubuisson's efficiency and Rankine efficiency.
They are given by the relations, D' Aubuisson's $\eta = \dfrac{wh}{Wh}$ or $\dfrac{qH}{Qh}$

Rankine's $\eta = \dfrac{w \times (H-h)}{(W-w) \times h}$ or $\dfrac{q \times (H-h)}{(Q-q) \times h}$

where w = Weight of water raised per sec, W = Weight of water flowing per sec into chamber,

h = Height of water above chamber in supply tank, H = Height of water raised above chamber.

8. Hydraulic lift is a device used for carrying persons or goods from one floor to another floor in a multi-storeyed building. They are of two types namely direct acting hydraulic lifts and suspended hydraulic lifts.
9. Hydraulic crane is a device used for raising or transferring heavy weights.
10. Fluid coupling is a device, in which power is transmitted from driving shaft to driven shaft without any change of torque while torque converter is a device in which arrangement is provided for getting increased or decreased torque at the driven shaft.

EXERCISE

(A) THEORETICAL PROBLEMS

1. Explain the term,'Hydraulic devices'. Name any five hydraulic devices.
2. Draw a neat sketch and explain the principle and working of a hydraulic press.
3. Define the term, the hydraulic accumulator. Obtain an expression for the capacity of a hydraulic accumulator. Differentiate between hydraulic accumulator and differential accumulator.
4. What is a hydraulic intensifier ? Explain its principle and working.
5. Differentiate between a hydraulic ram and a centrifugal pump. Obtain an expression for the efficiencies of the hydraulic ram.
6. Explain with the help of a neat sketch, the principle and working of the following hydraulic devices :
   (*a*) Hydraulic lift, (*b*) Hydraulic crane,
   (*c*) Hydraulic coupling, and (*d*) Hydraulic torque converter.
7. What is a difference between a fluid coupling and fluid torque converter? Explain the torque converter with a sketch.
8. Explain with neat sketch, the working of air lift pump. Mention its advantages.
9. How does a torque converter differ from a fluid coupling ? Explain the working principle of any one of them.

(B) NUMERICAL PROBLEMS

1. A hydraulic press has a ram of 300 mm diameter and a plunger of 50 mm diameter. Find the weight lifted by the hydraulic press when the force applied at the plunger is 40 N. [**Ans.** 1440 N]
2. A hydraulic press has a ram of 150 mm diameter and plunger of 30 mm. The stroke of the plunger is 250 mm and weight lifted is 600 N. If the distance moved by the weight is 1.20 m in 20 minutes, determine : (*a*) the force applied on the plunger, (*b*) power required to drive the plunger, and (*c*) number of strokes performed by the plunger. [**Ans.** (*a*) 24 N, (*b*) 0.0006 kW, (*c*) 120]
3. The water is supplied at a pressure of 15 N/cm^2 to an accumulator, having a ram of diameter 2.0 m. If the total lift of the ram is 10 m, determine :
   (*a*) the capacity of the accumulator, and
   (*b*) total weight placed on the ram (including the weight of the ram).
   [**Ans.** (*a*) 4712.4 kNm, (*b*) 471240 N]

4. The diameters of the fixed ram and fixed cylinder of an intensifier are 100 mm and 250 mm respectively. If the pressure of the water supplied to the fixed cylinder is 25 N/cm^2, find the pressure of the water flowing through the fixed ram. [**Ans.** 156.25 N/cm^2]

5. The water is supplied at the rate of 30 litres per second from a height of 4 m to a hydraulic ram, which raises 3 litres per second to a height of 18 m from the ram. Determine D' Aubuisson's and Rankine's efficiencies of the hydraulic ram. [**Ans**. 45%, 38.8%]

6. A hydraulic lift is required to lift a load of 98.1 kN through a height of 12 m, once in every 100 seconds. The speed of the lift is 600 mm/s. Determine :

(*a*) power required to drive the lift, (*b*) working period of lift in seconds, and (*c*) idle period of the lift in seconds. [**Ans.** (*a*) 11.772 kW, (*b*) 20 sec, (*c*) 80 sec]

7. Find the efficiency of a hydraulic crane, which is supplied 400 litres of water under a pressure of 490.5 N/cm^2 for lifting a weight of 98.1 kN through a height of 10 m. [**Ans**. 50%]

8. In a hydraulic coupling, the speeds of the driving and driven shafts are 800 r.p.m. and 780 r.p.m. respectively. Find :

(*a*) the efficiency of the hydraulic coupling, and (*b*) the slip of the coupling. [**Ans.** (*a*) 97.5%, (*b*) 2.5%]

OBJECTIVE TYPE QUESTIONS

Tick mark (√) the most appropriate statement of the multiple choice answers :

1. An ideal fluid is defined as the fluid which
- (*a*) is compressible
- (*b*) is incompressible
- (*c*) is incompressible and non-viscous (inviscid)
- (*d*) has negligible surface tension.

2. Newton's law of viscosity states that
- (*a*) shear stress is directly proportional to the velocity
- (*b*) shear stress is directly proportional to velocity gradient
- (*c*) shear stress is directly proportional to shear strain
- (*d*) shear stress is directly proportional to the viscosity.

3. A Newtonian fluid is defined as the fluid which
- (*a*) is incompressible and non-viscous
- (*b*) obeys Newton's law of viscosity
- (*c*) is highly viscous
- (*d*) is compressible and non-viscous.

4. Kinematic viscosity is defined as equal to
- (*a*) dynamic viscosity × density
- (*b*) dynamic viscosity/density
- (*c*) dynamic viscosity × pressure
- (*d*) pressure × density.

5. Dynamic viscosity (μ) has the dimensions as
- (*a*) MLT^{-2} (*b*) $ML^{-1}T^{-1}$
- (*c*) $ML^{-1}T^{-2}$ (*d*) $M^{-1}L^{-1}T^{-1}$

6. Poise is the unit of
- (*a*) mass density
- (*b*) kinematic viscosity
- (*c*) viscosity
- (*d*) velocity gradient.

7. The increase of temperature
- (*a*) increases the viscosity of a liquid
- (*b*) decreases the viscosity of a liquid
- (*c*) decreases the viscosity of a gas
- (*d*) increases the viscosity of a gas.

8. Stoke is the unit of
- (*a*) surface tension
- (*b*) viscosity
- (*c*) kinematic viscosity
- (*d*) none of the above.

9. The dividing factor for converting one poise into MKS unit of dynamic viscosity is
- (*a*) 9.81 (*b*) 98.1
- (*c*) 981 (*d*) 0.981.

10. Surface tension has the units of
- (*a*) force per unit area
- (*b*) force per unit length
- (*c*) force per unit volume
- (*d*) none of the above.

11. The gases are considered incompressible when Mach number
- (*a*) is equal to 1.0 (*b*) is equal to 0.50
- (*c*) is more than 0.3 (*d*) is less than 0.2.

12. Pascal's law states that pressure at a point is equal in all directions
- (*a*) in a liquid at rest
- (*b*) in a fluid at rest
- (*c*) in a laminar flow
- (*d*) in a turbulent flow.

13. The hydrostatic law states that rate of increase of pressure in a vertical direction is equal to
- (*a*) density of the fluid
- (*b*) specific weight of the fluid
- (*c*) weight of the fluid
- (*d*) none of the above.

14. Fluid statics deals with
- (*a*) viscous and pressure forces
- (*b*) viscous and gravity forces
- (*c*) gravity and pressure forces
- (*d*) surface tension and gravity forces.

15. Gauge pressure at a point is equal to
- (*a*) absolute pressure plus atmospheric pressure
- (*b*) absolute pressure minus atmospheric pressure
- (*c*) vacuum pressure plus absolute pressure
- (*d*) none of the above.

16. Atmospheric pressure held in terms of water column is
- (*a*) 7.5 m (*b*) 8.5 m
- (*c*) 9.81 m (*d*) 10.30 m.

17. The hydrostatic pressure on a plane surface is equal to
- (*a*) $wA\bar{h}$ (*b*) $wA\bar{h}\sin^2\theta$.
- (*c*) $\frac{1}{2}wA\bar{h}$ (*d*) $wA\bar{h}\sin\theta$.

where A = Area of plane surface, and
h = Depth of centroid of the plane area below the liquid-free surface.

18. Centre of pressure of a plane surface immersed in a liquid is
(*a*) above the centre of gravity of the plane surface
(*b*) at the centre of gravity of the plane surface
(*c*) below the centre of gravity of the plane surface
(*d*) none of the above.

19. The resultant hydrostatic force acts through a point known as
(*a*) centre of gravity
(*b*) centre of buoyancy
(*c*) centre of pressure
(*d*) none of the above.

20. For a submerged curved surface, the vertical component of the hydrostatic force is
(*a*) mass of the liquid supported by the curved surface
(*b*) weight of the liquid supported by the curved surface
(*c*) the force on the projected area of the curved surface on vertical plane
(*d*) none of the above.

21. For a floating body, the buoyant force passes through the
(*a*) centre of gravity of the body
(*b*) centre of gravity of the submerged part of the body
(*c*) metacentre of the body
(*d*) centroid of the liquid displaced by the body.

22. The condition of stable equilibrium for a floating body is
(*a*) the metacentre M coincides with the centre of gravity G
(*b*) the metacentre M is below centre of gravity G
(*c*) the metacentre M is above centre of gravity G
(*d*) the centre of buoyancy B is above centre of gravity G.

23. A submerged body will be in stable equilibrium if
(*a*) the centre of buoyancy B is below the centre of gravity G
(*b*) the centre of buoyancy B coincides with G
(*c*) the centre of buoyancy B is above the metacentre M
(*d*) the centre of buoyancy B is above G.

24. The metacentric height of a floating body is
(*a*) the distance between metacentre and centre of buoyancy
(*b*) the distance between the centre of buoyancy and centre of gravity
(*c*) the distance between metacentre and centre of gravity
(*d*) none of the above.

25. The necessary condition for the flow to be steady is that
(*a*) the velocity does not change from place to place
(*b*) the velocity is constant at a point with respect to time
(*c*) the velocity changes at a point with respect to time
(*d*) none of the above.

26. The necessary condition for the flow to be uniform is that
(*a*) the velocity is constant at a point with respect to time
(*b*) the velocity is constant in the flow field with respect to space
(*c*) the velocity changes at a point with respect to time
(*d*) none of the above.

27. The flow in pipe is laminar if
(*a*) Reynolds number is equal to 2500
(*b*) Reynolds number is equal to 4000
(*c*) Reynolds number is more than 2500
(*d*) none of the above.

28. A stream line is a line
(*a*) which is along the path of a particle
(*b*) which is always parallel to the main direction of flow
(*c*) across which there is no flow
(*d*) on which tangent drawn at any point gives the direction of velocity.

29. Continuity equation can take the form
(*a*) $A_1 V_1 = A_2 V_2$
(*b*) $\rho_1 A_1 = \rho_2 A_2$
(*c*) $\rho_1 A_1 V_1 = \rho_2 A_2 V_2$
(*d*) $p_1 A_1 V_1 = p_2 A_2 V_2$.

30. Pitot-tube is used for measurement of
(*a*) pressure
(*b*) flow
(*c*) velocity at a point
(*d*) discharge.

31. Bernoulli's theorem deals with the law of conservation of
(*a*) mass (*b*) momentum
(*c*) energy (*d*) none of the above.

32. Continuity equation deals with the law of conservation of
(*a*) mass (*b*) momentum
(*c*) energy (*d*) none of the above.

33. Irrotational flow means
(*a*) the fluid does not rotated while moving
(*b*) the fluid moves in straight lines
(*c*) the net rotation of fluid particles about their mass centre is zero
(*d*) none of the above.

34. The velocity components in x and y directions in terms of velocity potential (ϕ) are
(*a*) $u = -\frac{\partial \phi}{\partial x}, v = \frac{\partial \phi}{\partial y}$
(*b*) $u = \frac{\partial \phi}{\partial y}, v = \frac{\partial \phi}{\partial x}$
(*c*) $u = -\frac{\partial \phi}{\partial x}, v = -\frac{\partial \phi}{\partial y}$
(*d*) $u = -\frac{\partial \phi}{\partial x}, v = -\frac{\partial \phi}{\partial y}$.

35. The velocity components in x and y directions in terms of stream function (ψ) are
(*a*) $u = \frac{\partial \psi}{\partial x}, v = \frac{\partial \psi}{\partial y}$
(*b*) $u = -\frac{\partial \psi}{\partial x}, v = \frac{\partial \psi}{\partial y}$
(*c*) $u = \frac{\partial \psi}{\partial y}, v = \frac{\partial \psi}{\partial x}$
(*d*) $u = -\frac{\partial \psi}{\partial y}, v = \frac{\partial \psi}{\partial x}$.

36. The relation between tangential velocity (V) and radius (r) is given by
(*a*) $V \times r$ = Constant for forced vortex
(*b*) V/r = Constant for forced vortex
(*c*) $V \times r$ = Constant for free vortex
(*d*) V/r = Constant for free vortex.

37. The pressure variation along the radial direction for vortex flow along a horizontal plane is given as
(*a*) $\frac{\partial p}{\partial r} = -\rho \frac{V^2}{r}$ (*b*) $\frac{\partial p}{\partial r} = \rho \frac{V}{r^2}$
(*c*) $\frac{\partial p}{\partial r} = \rho \frac{V^2}{r}$ (*d*) none of the above.

38. For a forced vortex flow, the height of paraboloid formed is equal to
(*a*) $\frac{p}{\rho g} + \frac{V^2}{2g}$ (*b*) $\frac{V^2}{2g}$
(*c*) $\frac{V^2}{r^2 \times 2g}$ (*d*) $\frac{\omega r^2}{2g}$.

39. Bernoulli's equation is derived making assumptions that
(*a*) the flow is uniform and incompressible
(*b*) the flow is non-viscous, uniform and steady
(*c*) the flow is steady, non-viscous, incompressible and irrotational
(*d*) none of the above.

40. The Bernoulli's equation can take the form
(*a*) $\frac{p_1}{\rho_1} + \frac{V_1^2}{2g} + Z_1 = \frac{p_2}{\rho_2} + \frac{V_2^2}{2g} + Z_2$
(*b*) $\frac{p_1}{\rho_1 g} + \frac{V_1^2}{2} + Z_1 = \frac{p_2}{\rho_2 g} + \frac{V_2^2}{2} + Z_2$
(*c*) $\frac{p_1}{\rho_1 g} + \frac{V_1^2}{2g} + gZ_1 = \frac{p_2}{\rho_2 g} + \frac{V_2^2}{2g} + gZ_2$
(*d*) $\frac{p_1}{\rho_1 g} + \frac{V_1^2}{2g} + Z_1 = \frac{p_2}{\rho_2 g} + \frac{V_2^2}{2g} + Z_2$.

41. The flow rate through a circular pipe is measured by
(*a*) Pitot-tube (*b*) Venturimeter
(*c*) Orifice-meter (*d*) None of the above.

42. The range for co-efficient of discharge (C_d) for a venturimeter is
(*a*) 0.6 to 0.7 (*b*) 0.7 to 0.8
(*c*) 0.8 to 0.9 (*d*) 0.95 to 0.99.

43. The co-efficient of velocity (C_v) for an orifice is
(*a*) $C_v = \sqrt{\frac{4x^2}{yH}}$ (*b*) $C_v = \frac{2x}{\sqrt{4yH}}$
(*c*) $C_v = \sqrt{\frac{x^2}{4yH}}$ (*d*) none of the above.

44. The co-efficient of discharge (C_d) in terms of C_v and C_c is
(*a*) $C_d = \frac{C_v}{C_c}$ (*b*) $C_d = C_v \times C_c$
(*c*) $C_d = \frac{C_c}{C_v}$ (*d*) none of the above.

45. An orifice is known as large orifice when the head of liquid from the centre of orifice is

(*a*) more than 10 times the depth of orifice
(*b*) less than 10 times the depth of orifice
(*c*) less than 5 times the depth of orifice
(*d*) none of the above.

46. Which mouthpiece is having maximum co-efficient of discharge
(*a*) external mouthpiece
(*b*) convergent-divergent mouthpiece
(*c*) internal mouthpiece
(*d*) none of the above.

47. The co-efficient of discharge (C_d)
(*a*) for an orifice is more than that for a mouthpiece
(*b*) for internal mouthpiece is more than that for external mouthpiece
(*c*) for a mouthpiece is more than that for an orifice
(*d*) none of the above.

48. A flow is said to be laminar when
(*a*) the fluid particles moves in a zig-zag way
(*b*) the Reynolds number is high
(*c*) the fluid particles move in layers parallel to the boundary
(*d*) none of the above.

49. For the laminar flow through a circular pipe
(*a*) the maximum velocity = 1.5 times the average velocity
(*b*) the maximum velocity = 2.0 times the average velocity
(*c*) the maximum velocity = 2.5 times the average velocity
(*d*) none of the above.

50. The loss of pressure head for the laminar flow through pipes varies
(*a*) as the square of velocity
(*b*) directly as the velocity
(*c*) as the inverse of the velocity
(*d*) none of the above.

51. For the laminar flow through a pipe, the shear stress over the cross-section
(*a*) varies inversely as the distance from the centre of the pipe
(*b*) varies directly as the distance from the surface of the pipe
(*c*) varies directly as the distance from the centre of the pipe
(*d*) remains constant over the cross-section.

52. For the laminar flow between two parallel plates
(*a*) the maximum velocity = 2.0 times the average velocity
(*b*) the maximum velocity = 2.5 times the average velocity
(*c*) the maximum velocity = 1.33 times the average velocity
(*d*) none of the above.

53. The value of the kinetic energy correction factor (α) of the viscous flow through a circular pipe is
(*a*) 1.33 (*b*) 1.50
(*c*) 2.0 (*d*) 1.25.

54. The value of the momentum correction factor (β) for the viscous flow through a circular pipe is
(*a*) 1.33 (*b*) 1.50
(*c*) 2.0 (*d*) 1.25.

55. The pressure drop per unit length of a pipe for laminar flow is
(*a*) equal to $\frac{12\mu\bar{U}L}{\rho g D^2}$ (*b*) equal to $\frac{12\mu\bar{U}}{\rho g D^2}$
(*c*) equal to $\frac{32\mu\bar{U}L}{\rho g D^2}$ (*d*) none of the above.

56. For viscous flow between two parallel plates, the pressure drop per unit length is equal to
(*a*) $\frac{12\mu\bar{U}L}{\rho g D^2}$ (*b*) $\frac{12\mu\bar{U}L}{D^2}$
(*c*) $\frac{32\mu\bar{U}L}{D^2}$ (*d*) $\frac{12\mu\bar{U}}{D^2}$.

57. The velocity distribution in laminar flow through a circular pipe follow the
(*a*) parabolic law (*b*) linear law
(*c*) logarithmic law (*d*) none of the above.

58. A boundary is known as hydrodynamically smooth if
(*a*) $\frac{k}{\delta'} = 0.3$ (*b*) $\frac{k}{\delta'} > 0.3$
(*c*) $\frac{k}{\delta'} < 0.25$ (*d*) $\frac{k}{\delta'} = 6.0$
where k = Average height of the irregularities from the boundary
and δ' = Thickness of laminar sub-layer.

59. The co-efficient of friction for laminar flow through a circular pipe is given by
(*a*) $f = \frac{0.0791}{(R_e)^{1/4}}$ (*b*) $f = \frac{16}{R_e}$
(*c*) $f = \frac{64}{R_e}$ (*d*) none of the above.

60. The loss of head due to sudden expansion of a pipe is given by

(*a*) $h_L = \frac{V_1^2 - V_2^2}{2g}$ (*b*) $h_L = \frac{0.5\, V_1^2}{2g}$

(*c*) $h_L = \frac{(V_1 - V_2)^2}{2g}$ (*d*) none of the above.

61. The loss of head due to sudden contraction of a pipe is equal to

(*a*) $\left(\frac{1}{C_c} - 1\right)^2 \frac{V_2}{2g}$ (*b*) $\left(1 - \frac{1}{C_c}\right)^2 \frac{V_2}{2g}$

(*c*) $\frac{1}{C_c}\left(1 - \frac{V_2^2}{2g}\right)$ (*d*) none of the above.

62. Hydraulic gradient line (H.G.L.) represents the sum of
(*a*) pressure head and kinetic head
(*b*) kinetic head and datum head
(*c*) pressure head, kinetic head and datum head
(*d*) pressure head and datum head.

63. Total energy line (T.E.L.) represents the sum of
(*a*) pressure head and kinetic head
(*b*) kinetic head and datum head
(*c*) pressure head and datum head
(*d*) pressure head, kinetic head and datum head.

64. When the pipes are connected in series, the total rate of flow
(*a*) is equal to the sum of the rate of flow in each pipe
(*b*) is equal to the reciprocal of the sum of the rate of flow in each pipe
(*c*) is the same as flowing through each pipe
(*d*) none of the above.

65. Power transmitted through pipes, will be maximum when
(*a*) head lost due to friction = $\frac{1}{2}$ total head at inlet of the pipe
(*b*) head lost due to friction = $\frac{1}{4}$ total head at inlet of the pipe
(*c*) head lost due to friction = total head at the inlet of the pipe
(*d*) head lost due to friction = $\frac{1}{3}$ total head at the inlet of the pipe.

66. The valve closure is said to be gradual if the time required to close the valve

(*a*) $t = \frac{2L}{C}$ (*b*) $t \le \frac{2L}{C}$

(*c*) $t < \frac{4L}{C}$ (*d*) $t > \frac{2L}{C}$

where L = Length of pipe, C = Velocity of pressure wave.

67. The velocity of pressure wave in terms of bulk modulus (K) and density (ρ) is given by

(*a*) $C = \sqrt{\frac{\rho}{K}}$ (*b*) $C = \sqrt{K\rho}$

(*c*) $C = \sqrt{\frac{K}{\rho}}$ (*d*) none of the above.

68. Reynold's number is defined as the
(*a*) ratio of inertia force to gravity force
(*b*) ratio of viscous force to gravity force
(*c*) ratio of viscous force to elastic force
(*d*) ratio of inertia force to viscous force.

69. Froude's number is defined as the ratio of
(*a*) inertia force to viscous force
(*b*) inertia force to gravity force
(*c*) inertia force to elastic force
(*d*) inertia force to pressure force.

70. Mach number is defined as the ratio of
(*a*) inertia force to viscous force
(*b*) viscous force to surface tension force
(*c*) viscous force to elastic force
(*d*) inertia force to elastic force.

71. Euler's number is the ratio of
(*a*) inertia force to pressure force
(*b*) inertia force to elastic force
(*c*) inertia force to gravity force
(*d*) none of the above.

72. Models are known undistorted model if
(*a*) the prototype and model are having different scale ratios
(*b*) the prototype and model are having same scale ratio
(*c*) model and prototype are kinematically similar
(*d*) none of the above.

73. Geometric similarity between model and prototype means
(*a*) the similarity discharge
(*b*) the similarity of linear dimensions
(*c*) the similarity of motion
(*d*) the similarity of forces.

74. Kinematic similarity between model and prototype means
(*a*) the similarity of forces
(*b*) the similarity of shape
(*c*) the similarity of motion
(*d*) the similarity of discharge.

75. Dynamic similarity between model and prototype means
(*a*) the similarity of forces
(*b*) the similarity of motion
(*c*) the similarity of shape
(*d*) none of the above.

76. Reynolds number is expressed as

(*a*) $R_e = \frac{\rho \mu L}{V}$ (*b*) $R_e = \frac{V \mu L}{\rho}$

(*c*) $R_e = \frac{\rho V L}{\mu}$ (*d*) $R_e = \frac{V \times d}{\nu}$.

77. Froude's number (F_e) is given by

(*a*) $F_e = V\sqrt{\frac{L}{g}}$ (*b*) $F_e = V\sqrt{\frac{g}{L}}$

(*c*) $F_e = \frac{V}{\sqrt{Lg}}$ (*d*) none of the above.

78. Mach number (M) is given by

(*a*) $M = \frac{C}{V}$ (*b*) $M = V \times C$

(*c*) $M = \frac{V}{C}$ (*d*) none of the above.

79. Boundary layer on a flat plate is called laminar boundary layer if
(*a*) Reynold number is less than 2000
(*b*) Reynold number is less than 4000
(*c*) Reynold number is less than 5×10^5
(*d*) None of the above.

80. Boundary layer thickness (δ) is the distance from the surface of the solid body in the direction perpendicular to flow, where the velocity of fluid is equal to
(*a*) free-stream velocity
(*b*) 0.9 times the free-stream velocity
(*c*) 0.99 times the free-stream velocity
(*d*) none of the above.

81. Displacement thickness (δ^*) is given by

(*a*) $\delta^* = \int_0^{\delta}\left(1 - \frac{U}{u}\right)dy$

(*b*) $\delta^* = \int_0^{\delta} \frac{u}{U}\left(1 - \frac{u}{U}\right)dy$

(*c*) $\delta^* = \int_0^{\delta} \frac{u}{U}\left(1 - \frac{u^2}{U^2}\right)dy$

(*d*) none of the above.

82. Momentum thickness (θ) is given by

(*a*) $\theta = \int_0^{\delta} \frac{u}{U}\left(1 - \frac{u}{U}\right)dy$

(*b*) $\theta = \int_0^{\delta} \left(1 - \frac{u}{U}\right)dy$

(*c*) $\theta = \int_0^{\delta} \frac{u}{U}\left(1 - \frac{u^2}{U^2}\right)dy$

(*d*) none of the above.

83. Energy thickness (δ^{**}) is equal to

(*a*) $\int_0^{\delta} \frac{u}{U}\left[1 - \frac{u}{U}\right]$

(*b*) $\int_0^{\delta} \frac{u}{U}\left(1 - \frac{u^2}{U^2}\right)dy$

(*c*) $\int_0^{\delta} \frac{u}{U}\left(1 - \frac{u}{U}\right)^2$

(*d*) none of the above.

84. Von-Karman momentum integral equation is given as

(*a*) $\frac{\tau_0}{\frac{1}{2}\rho U^2} = \frac{\partial \theta}{\partial x}$ (*b*) $\frac{\tau_0}{\rho U^2} = \frac{\partial \theta}{\partial x}$

(*c*) $\frac{\tau_0}{2\rho U^2} = \frac{\partial \theta}{\partial x}$ (*d*) none of the above.

85. The boundary layer separation takes place if
(*a*) pressure gradient is zero
(*b*) pressure gradient is positive
(*c*) pressure gradient is negative
(*d*) none of the above.

86. The condition for boundary layer separation is

(*a*) $\left(\frac{\partial u}{\partial y}\right)_{y=0} = +\text{ve}$ (*b*) $\left(\frac{\partial u}{\partial y}\right)_{y=0} = -\text{ve}$

(*c*) $\left(\frac{\partial u}{\partial y}\right)_{y=0} = 0$ (*d*) none of the above.

87. The boundary layer flow will be attached to the surface if

(*a*) $\left(\frac{\partial u}{\partial y}\right)_{y=0} = 0$ (*b*) $\left(\frac{\partial u}{\partial y}\right)_{y=0} = +\text{ve}$

(*c*) $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ = –ve (*d*) none of the above.

88. The condition for detached flow is

(*a*) $\left(\frac{\partial u}{\partial y}\right)_{y=0} = 0$ (*b*) $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ = +ve

(*c*) $\left(\frac{\partial u}{\partial y}\right)_{y=0}$ = –ve (*d*) none of the above.

89. Drag is defined as the force exerted by a flowing fluid on a solid body
(*a*) in the direction of flow
(*b*) perpendicular to the direction of flow
(*c*) in the direction which is at an angle of 45° to the direction of flow
(*d*) none of the above.

90. Lift force is defined as the force exerted by a flowing fluid on a solid body
(*a*) in the direction of flow
(*b*) perpendicular to the direction of flow
(*c*) at an angle of 45° to the direction of flow
(*d*) none of the above.

91. Drag force is expressed mathematically, as

(*a*) $F_D = \frac{1}{2}\rho U^2 \times C_D \times A$

(*b*) $F_D = \rho U^2 \times C_D \times A$

(*c*) $F_D = 2\rho U^2 \times C_D \times A$

(*d*) none of the above.

92. Lift force (F_L) is expressed mathematically, as

(*a*) $F_L = \frac{1}{2}\rho U^2 \times C_L$

(*b*) $F_L = \frac{1}{2}\rho U^2 \times C_L \times A$

(*c*) $F_L = 2\rho U^2 \times C_L \times A$

(*d*) $F_L = \rho U^2 \times C_L \times A$.

93. Total drag on a body is the sum of
(*a*) pressure drag and velocity drag
(*b*) pressure drag and friction drag
(*c*) friction drag and velocity drag
(*d*) none of the above.

94. A body is called stream lined body when it is placed in a flow and the surface of the body
(*a*) coincides with streamlines
(*b*) does not coincide with the streamlines
(*c*) is perpendicular to the streamlines
(*d*) none of the above.

95. A body is called bluff body if the surface of the body
(*a*) coincides with the streamlines
(*b*) does not coincide with the streamlines
(*c*) is very smooth
(*d*) none of the above.

96. The drag on a sphere (F_D) for Reynold's number less than 0.2 is given by
(*a*) $F_D = 5\pi\mu DU$ (*b*) $F_D = 3\pi\mu DU$
(*c*) $F_D = 2\ \pi\mu DU$ (*d*) $F_D = \pi\mu DU$.

97. The skin friction drag on a sphere (for Reynold's number less than 0.2) is equal to
(*a*) one-third of the total drag
(*b*) half of the total drag
(*c*) two-third of the total drag
(*d*) none of the above.

98. The pressure drag on a sphere (for Reynold's number less than 0.2) is equal to
(*a*) one-third of the total drag
(*b*) half of the total drag
(*c*) two-third of the total drag
(*d*) none of the above.

99. Terminal velocity of a falling body is equal to
(*a*) a maximum velocity with which body will fall
(*b*) the maximum constant velocity with which body will fall
(*c*) half of the maximum velocity
(*d*) none of the above.

100. When a falling body has attained terminal velocity, the weight of the body is equal to
(*a*) drag force minus buoyant force
(*b*) buoyant force minus drag force
(*c*) drag force plus the buoyant force
(*d*) none of the above.

101. The tangential velocity of ideal fluid at any point on the surface of the cylinder is given by

(*a*) $u_\theta = \frac{1}{2}\ U \sin\theta$ (*b*) $u_\theta = U \sin\theta$

(*c*) $u_\theta = 2U \sin\theta$ (*d*) none of the above.

102. The lift force (F_L) produced on a rotating circular cylinder in a uniform flow is given by

(*a*) $F_L = \frac{LU\Gamma}{\rho}$ (*b*) $F_L = \rho LU\Gamma$

(*c*) $F_L = \frac{\rho U\Gamma}{\rho}$ (*d*) $F_L = \frac{\rho LU}{\Gamma}$

where L = Length of the cylinder, U = Free-stream velocity, Γ = Circulation.

103. The lift co-efficient (C_L) for a rotating cylinder in a uniform flow is given by

(*a*) $C_L = \dfrac{\Gamma U}{R}$ (*b*) $C_L = \dfrac{\Gamma R}{U}$

(*c*) $C_L = \dfrac{\Gamma}{RU}$ (*d*) $C_L = \dfrac{RU}{\Gamma}$.

104. Kinematic viscosity (ν) is equal to

(*a*) $\mu \times \rho$ (*b*) $\dfrac{\mu}{\rho}$

(*c*) $\dfrac{\rho}{\mu}$ (*d*) none of the above.

105. Compressibility is equal to

(*a*) $\dfrac{\left(\dfrac{dV}{V}\right)}{dp}$ (*b*) $\dfrac{dp}{-\left(\dfrac{dV}{V}\right)}$

(*c*) $\dfrac{dp}{d\rho}$ (*d*) $\sqrt{\dfrac{dp}{d\rho}}$.

106. Hydrostatic law of pressure is given as

(*a*) $\dfrac{\partial p}{\partial z} = \rho g$ (*b*) $\dfrac{\partial p}{\partial z} = 0$

(*c*) $\dfrac{\partial p}{\partial z} = z$ (*d*) $\dfrac{\partial p}{\partial z}$ = constant.

107. Four curves are shown in Fig. 1 with velocity gradient $\left(\dfrac{\partial u}{\partial y}\right)$ along x-axis and viscous shear stress (τ) along y-axis. Curve A corresponds to

(*a*) ideal fluid
(*b*) newtonian fluid
(*c*) non-newtonian fluid
(*d*) ideal solid.

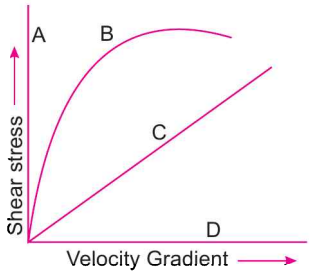


Fig. 1

108. Curve B in Fig.1 corresponds to

(*a*) ideal fluid
(*b*) newtonian fluid
(*c*) non-newtonian fluid
(*d*) ideal solid.

109. Curve C in Fig. 1 corresponds to

(*a*) ideal fluid
(*b*) newtonian fluid
(*c*) non-newtonian fluid
(*d*) ideal solid.

110. Curve D in Fig. 1 corresponds to

(*a*) ideal fluid
(*b*) newtonian fluid
(*c*) non-newtonian fluid
(*d*) ideal solid.

111. The relation between surface tension (σ) and difference of pressure (Δp) between the inside and outside of a liquid droplet is given as

(*a*) $\Delta p = \dfrac{\sigma}{4d}$ (*b*) $\Delta p = \dfrac{\sigma}{2d}$

(*c*) $\Delta p = \dfrac{4\sigma}{d}$ (*d*) $\Delta p = \dfrac{\sigma}{d}$.

112. For a soap bubble, the surface tension (σ) and difference of pressure (Δp) are related as

(*a*) $\Delta p = \dfrac{\sigma}{4d}$ (*b*) $\Delta p = \dfrac{\sigma}{2d}$

(*c*) $\Delta p = \dfrac{4\sigma}{d}$ (*d*) $\Delta p = \dfrac{8\sigma}{d}$.

113. For a liquid jet, the surface tension (σ) and difference of pressure (Δp) are related as

(*a*) $\Delta p = \dfrac{\sigma}{4d}$ (*b*) $\Delta p = \dfrac{\sigma}{2d}$

(*c*) $\Delta p = \dfrac{4\sigma}{d}$ (*d*) $\Delta p = \dfrac{2\sigma}{d}$.

114. The capillary rise or fall of a liquid is given by

(*a*) $h = \dfrac{\sigma \cos\theta}{4\rho g d}$ (*b*) $h = \dfrac{4\sigma \cos\theta}{\rho g d}$

(*c*) $h = \dfrac{8\sigma \cos\theta}{\rho g d}$ (*d*) none of the above.

115. Manometer is a device used for measuring

(*a*) velocity at a point in fluid
(*b*) pressure at a point in a fluid
(*c*) discharge of a fluid
(*d*) none of the above.

116. Differential manometers are used for measuring

(*a*) velocity at a point in a fluid
(*b*) pressure at a point in a fluid
(*c*) difference of pressure between two points
(*d*) none of the above.

117. The pressure at a height Z in a static compressible fluid undergoing isothermal compression is given by

(*a*) $p = p_0 e^{-\frac{gR}{ZT}}$ (*b*) $p = p_0 e^{-\frac{gT}{RZ}}$

(*c*) $p = p_0 e^{-\frac{RT}{gZ}}$ (*d*) $p = p_0 e^{-\frac{gT}{RT}}$

where p_0 = Pressure at ground level, R = Gas constant, T = Absolute temperature.

118. The pressure at a height Z in a static compressible fluid undergoing adiabatic compression is given by

(*a*) $p = p_0\left[1-\frac{\gamma-1}{\gamma}\frac{RT_0}{gZ}\right]^{\frac{\gamma}{\gamma-1}}$

(*b*) $p = p_0\left[1-\frac{\gamma}{\gamma-1}\frac{RT_0}{gZ}\right]^{\frac{\gamma}{\gamma-1}}$

(*c*) $p = p_0\left[1-\frac{\gamma-1}{\gamma}\frac{gZ}{RT_0}\right]^{\frac{\gamma}{\gamma-1}}$

(*d*) none of the above.

119. The temperature at a height Z in a static compressible fluid undergoing adiabatic compression is given as

(*a*) $T = T_0\left[1-\frac{\gamma-1}{\gamma}\frac{RT_0}{gZ}\right]$

(*b*) $T = T_0\left[1-\frac{\gamma-1}{\gamma}\frac{gZ}{RT_0}\right]$

(*c*) $T = T_0\left[1-\frac{\gamma}{\gamma-1}\frac{RT_0}{gZ}\right]$

(*d*) none of the above.

120. Temperature lapse-rate is given by

(*a*) $L = -\frac{R}{g}\left[\frac{\gamma-1}{\gamma}\right]$ (*b*) $L = -\frac{R}{g}\left[\frac{\gamma}{\gamma-1}\right]$

(*c*) $L = -\frac{g}{R}\left[\frac{\gamma-1}{\gamma}\right]$ (*d*) none of the above.

121. When the fluid is at rest, the shear stress is

(*a*) maximum (*b*) zero

(*c*) unpredictable (*d*) none of the above.

122. The depth of centre of pressure of an inclined immersed surface from free surface of liquid is equal to

(*a*) $\frac{I_G}{A\bar{h}}+\bar{h}$ (*b*) $\frac{I_G A\sin^2\theta}{\bar{h}}+\bar{h}$

(*c*) $\frac{I_G\sin^2\theta}{A\bar{h}}+\bar{h}$ (*d*) $\frac{I_G\bar{h}}{A\sin^2\theta}+\bar{h}.$

123. The depth of centre of pressure of a vertical immersed surface from free surface of liquid is equal to

(*a*) $\frac{I_G}{A\bar{h}}+\bar{h}$ (*b*) $\frac{I_G A}{\bar{h}}+\bar{h}$

(*c*) $\frac{I_G\bar{h}}{\bar{h}}+\bar{h}$ (*d*) $\frac{A\bar{h}}{I_G}+\bar{h}.$

124. The centre of pressure for a plane vertical surface lies at a depth of

(*a*) half the height of the immersed surface

(*b*) one-third the height of the immersed surface

(*c*) two-third the height of the immersed surface

(*d*) none of the above.

125. The inlet length of a venturimeter

(*a*) is equal to the outlet length

(*b*) is more than the outlet length

(*c*) is less than the outlet length

(*d*) none of the above.

126. Flow of a fluid in a pipe takes place from

(*a*) higher level to lower level

(*b*) higher pressure to lower pressure

(*c*) higher energy to lower energy

(*d*) none of the above.

127. The point, through which the buoyant force is acting, is called

(*a*) centre of pressure

(*b*) centre of gravity

(*c*) centre of buoyancy

(*d*) none of the above.

128. The point, through which the weight is acting, is called

(*a*) centre of pressure

(*b*) centre of gravity

(*c*) centre of buoyancy

(*d*) none of the above.

129. The point, about which a floating body starts oscillating when the body is tilted, is called

(*a*) centre of pressure

(*b*) centre of buoyancy

(*c*) centre of gravity

(*d*) metacentre.

130. The metacentric height (GM) is given by

(a) $GM = BG - \frac{I}{V}$ (b) $GM = \frac{V}{I} - BG$

(c) $GM = \frac{I}{V} - BG$ (d) none of the above.

131. For a floating body, if the metacentre is above the centre of gravity, the equilibrium is called

(a) stable (b) unstable

(c) neutral (d) none of the above.

132. For a floating body, if the metacentre is below the centre of gravity, the equilibrium is called

(a) stable (b) unstable

(c) neutral (d) none of the above.

133. For a floating body, if the metacentre coincides with the centre of gravity, the equilibrium is called

(a) stable (b) unstable

(c) neutral (d) none of the above.

134. For a floating body, if centre of buoyancy is above the centre of gravity, the equilibrium is called

(a) stable (b) unstable

(c) neutral (d) none of the above.

135. For a sub-merged body, if the centre of buoyancy is above the centre of gravity, the equilibrium is called

(a) stable (b) unstable

(c) neutral (d) none of the above.

136. For a sub-merged body, if the centre of buoyancy is below the centre of gravity, the equilibrium is called

(a) stable (b) unstable

(c) neutral (d) none of the above.

137. For a sub-merged body, if the centre of buoyancy coincides with the centre of gravity, the equilibrium is called

(a) stable (b) unstable

(c) neutral (d) none of the above.

138. For a sub-merged body, if the metacentre is below the centre of gravity, the equilibrium is called

(a) stable (b) unstable

(c) neutral (d) none of the above.

139. The metacentric height (GM) experimentally is given as

(a) $GM = \frac{W \tan\theta}{wx}$ (b) $GM = \frac{w \tan\theta}{W \times x}$

(c) $GM = \frac{wx}{W \tan\theta}$ (d) $GM = \frac{Wx}{w \tan\theta}$

where w = Movable weight, W = Weight of floating body including w, θ = Angle of tilt.

140. The time period of oscillation of a floating body is given by

(a) $T = 2\pi\sqrt{\frac{GM \times g}{k^2}}$ (b) $T = 2\pi\sqrt{\frac{k^2}{GM \times g}}$

(c) $T = 2\pi\sqrt{\frac{GM}{gk^2}}$ (d) $T = 2\pi\sqrt{\frac{gk^2}{GM}}$.

where k = Radius of gyration, GM = Metacentric height, and T = Time period.

141. If the velocity, pressure, density etc., do not change at a point with respect to time, flow is called

(a) uniform (b) incompressible

(c) non-uniform (d) steady.

142. If the velocity, pressure, density, etc., change at a point with respect to time, the flow is called

(a) uniform (b) compressible

(c) unsteady (d) incompressible.

143. If the velocity in a fluid flow does not change with respect to length of direction of flow, it is called

(a) steady flow

(b) uniform flow

(c) incompressible flow

(d) rotational flow.

144. If the velocity in a fluid flow changes with respect to length of direction of flow, it is called

(a) unsteady flow

(b) compressible flow

(c) irrotational flow

(d) none of the above.

145. If the density of a fluid is constant from point to point in a flow region, it is called

(a) steady flow

(b) incompressible flow

(c) uniform flow

(d) rotational flow.

146. If the density of a fluid changes from point to point in a flow region, it is called

(a) steady flow (b) unsteady flow

(c) non-uniform flow (d) compressible flow.

147. If the fluid particles move in straight lines and all the lines are parallel to the surface, the flow is called

(a) steady (b) uniform

(c) compressible (d) laminar.

148. If the fluid particles move in a zig-zag way, the flow is called

(a) unsteady (b) non-uniform

(c) turbulent (d) incompressible.

149. The acceleration of a fluid particle in the direction of x is given by

(*a*) $A_x = u\dfrac{\partial u}{\partial x} + v\dfrac{\partial v}{\partial y} + w\dfrac{\partial w}{\partial z} + \dfrac{\partial u}{\partial t}$

(*b*) $A_x = u\dfrac{\partial u}{\partial x} + u\dfrac{\partial v}{\partial y} + u\dfrac{\partial w}{\partial z} + \dfrac{\partial u}{\partial t}$

(*c*) $A_x = u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} + w\dfrac{\partial u}{\partial z} + \dfrac{\partial u}{\partial t}$

(*d*) none of the above.

150. The local acceleration in the direction of x is given by

(*a*) $u\dfrac{\partial u}{\partial x} + \dfrac{\partial u}{\partial t}$ (*b*) $\dfrac{\partial u}{\partial t}$

(*c*) $u\dfrac{\partial u}{\partial x}$ (*d*) none of the above.

151. The convective acceleration in the direction of x is given by

(*a*) $u\dfrac{\partial u}{\partial y} + v\dfrac{\partial v}{\partial y} + w\dfrac{\partial w}{\partial z}$

(*b*) $u\dfrac{\partial u}{\partial x} + u\dfrac{\partial u}{\partial y} + u\dfrac{\partial u}{\partial z}$

(*c*) $u\dfrac{\partial u}{\partial x} + u\dfrac{\partial v}{\partial y} + u\dfrac{\partial w}{\partial z}$

(*d*) $u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} + w\dfrac{\partial u}{\partial z}$.

152. Shear strain rate is given by

(*a*) $\dfrac{1}{2}\left(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\right)$ (*b*) $\dfrac{1}{2}\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}$

(*c*) $\dfrac{1}{2}\left(\dfrac{\partial v}{\partial x} + \dfrac{\partial u}{\partial y}\right)$ (*d*) $\dfrac{1}{2}\dfrac{\partial v}{\partial x} + \dfrac{\partial u}{\partial y}$.

153. For a two-dimensional fluid element in x-y plane, the rotational component is given as

(*a*) $\omega_z = \dfrac{1}{2}\left(\dfrac{\partial v}{\partial x} + \dfrac{\partial u}{\partial y}\right)$

(*b*) $\omega_z = \dfrac{1}{2}\left(\dfrac{\partial u}{\partial x} - \dfrac{\partial v}{\partial y}\right)$

(*c*) $\omega_z = \dfrac{1}{2}\left(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\right)$

(*d*) $\omega_z = \dfrac{1}{2}\left(\dfrac{\partial v}{\partial x} - \dfrac{\partial u}{\partial y}\right)$.

154. Vorticity is given by

(*a*) two times the rotation
(*b*) 1.5 times the rotation
(*c*) three times the rotation
(*d*) equal to the rotation.

155. Study of fluid motion with the forces causing the flow is known as

(*a*) kinematics of fluid flow
(*b*) dynamics of fluid flow
(*c*) statics of fluid flow
(*d*) none of the above.

156. Study of fluid motion without considering the forces, causing the flow, is known as

(*a*) kinematics of fluid flow
(*b*) dynamics of fluid flow
(*c*) statics of fluid flow
(*d*) none of the above.

157. Study of fluid at rest is known as

(*a*) kinematics (*b*) dynamics
(*c*) statics (*d*) none of the above.

158. The term $V^2/2g$ is known as

(*a*) kinetic energy
(*b*) pressure energy
(*c*) kinetic energy per unit weight
(*d*) none of the above.

159. The terms $p/\rho g$ is known as

(*a*) kinetic energy per unit weight
(*b*) pressure energy
(*c*) pressure energy per unit weight
(*d*) none of the above.

160. The term Z is known as

(*a*) potential energy
(*b*) pressure energy
(*c*) potential energy per unit weight
(*d*) none of the above.

161. The discharge through a venturimeter is given as

(*a*) $Q = \dfrac{A_1^2 A_2^2}{\sqrt{A_1^2 - A_2^2}} \times \sqrt{2gh}$

(*b*) $Q = \dfrac{A_1 A_2}{\sqrt{2A_1^2 - A_2^2}} \times \sqrt{2gh}$

(*c*) $Q = \dfrac{A_1 A_2}{\sqrt{A_1^2 - A_2^2}} \times \sqrt{2gh}$

(*d*) none of the above.

162. The difference of pressure head (h) measured by mercury-oil differential manometer is given as

(*a*) $h = x\left[1 - \frac{S_g}{S_o}\right]$ (*b*) $h = x\left[S_g - S_o\right]$

(*c*) $h = x\left[S_o - S_g\right]$ (*d*) $h = x\left[\frac{S_g}{S_o} - 1\right]$

where x = Difference of mercury level, S_g = Specific gravity of mercury and S_o = Specific gravity of oil.

163. The difference of pressure head (h) measured by a differential manometer containing lighter liquid is

(*a*) $h = x\left[1 - \frac{S_l}{S_o}\right]$ (*b*) $h = x\left[\frac{S_l}{S_o} - 1\right]$

(*c*) $h = x\left[S_o - S_l\right]$ (*d*) none of the above.

where S_l = Specific gravity of lighter liquid in manometer.

S_o = Specific gravity of fluid flowing

x = Difference of lighter liquid levels in differential manometer.

164. Pitot-tube is used to measure
(*a*) discharge
(*b*) average velocity
(*c*) velocity at a point
(*d*) pressure at a point.

165. Venturimeter is used to measure
(*a*) discharge
(*b*) average velocity
(*c*) velocity at a point
(*d*) pressure at a point.

166. Orifice-meter is used to measure
(*a*) discharge
(*b*) average velocity
(*c*) velocity at a point
(*d*) pressure at a point.

167. For a sub-merged curved surface, the horizontal component of force due to static liquid is equal to
(*a*) weight of liquid supported by the curved surface
(*b*) force on a projection of the curved surface on a vertical plane
(*c*) area of curved surface × pressure at the centroid of the submerged area
(*d*) none of the above.

168. For a sub-merged curved surface, the vertical component of force due to static liquid is equal to
(*a*) weight of the liquid supported by curved surface
(*b*) force on a projection of the curved surface on a vertical plane
(*c*) area of curved surface × pressure at the centroid of the sub-merged area
(*d*) none of the above.

169. An oil of specific gravity 0.7 and pressure 0.14 kgf/cm^2 will have the height of oil as
(*a*) 70 cm of oil (*b*) 2 m of oil
(*c*) 20 cm of oil (*d*) 80 cm of oil.

170. The difference in pressure head, measured by a mercury water differential manometer for a 20 cm difference of mercury level will be
(*a*) 2.72 m (*b*) 2.52 m
(*c*) 2.0 m (*d*) 0.2 m.

171. The difference in pressure head, measured by a mercury-oil differential manometer for a 20 cm difference of mercury level will be (sp. gravity of oil = 0.8)
(*a*) 2.72 m of oil (*b*) 2.52 m of oil
(*c*) 3.20 m of oil (*d*) 2.0 m of oil.

172. The rate of flow through a venturimeter varies as
(*a*) H (*b*) $\sqrt{H}$
(*c*) $H^{3/2}$ (*d*) $H^{5/2}$.

173. The rate of flow through a V-notch varies as
(*a*) H (*b*) $\sqrt{H}$
(*c*) $H^{3/2}$ (*d*) $H^{5/2}$.

174. Orifices are used to measure
(*a*) velocity (*b*) pressure
(*c*) rate of flow (*d*) none of the above.

175. Mouthpieces are used to measure
(*a*) velocity (*b*) pressure
(*c*) viscosity (*d*) rate of flow.

176. The ratio of actual velocity of a jet of water at vena-contracta to the theoretical velocity, is known as
(*a*) co-efficient of discharge
(*b*) co-efficient of velocity
(*c*) co-efficient of contraction
(*d*) co-efficient of viscosity.

177. The ratio of actual discharge of a jet of water to its theoretical discharge is known as
(*a*) co-efficient of discharge
(*b*) co-efficient of velocity
(*c*) co-efficient of contraction
(*d*) co-efficient of viscosity.

178. The ratio of the area of the jet of water at vena-contracta to the area of orifice, is known as
(*a*) co-efficient of discharge
(*b*) co-efficient of velocity
(*c*) co-efficient of contraction
(*d*) co-efficient of viscosity.

179. The discharge through a large rectangular orifice is
(*a*) $\frac{2}{3}C_d \times b \times \sqrt{2g}(\sqrt{H_2} - \sqrt{H_1})$
(*b*) $\frac{8}{15}C_d \times b \times \sqrt{2g}(H_2^{3/2} - H_1^{3/2})$
(*c*) $\frac{2}{3}C_d \times b \times \sqrt{2g}(H_2^{3/2} - H_1^{3/2})$
(*d*) none of the above.
where b = Width of orifice, H_1 = Height of liquid above top edge of the orifice,
H_2 = Height of liquid above bottom edge of orifice.

180. The discharge through fully sub-merged orifice is
(*a*) $C_d \times b \times (H_2 - H_1) \times \sqrt{2g} \times H^{3/2}$
(*b*) $C_d \times b \times (H_2 - H_1) \times \sqrt{2gH}$
(*c*) $C_d \times b \times (H_2^{3/2} - H_1^{3/2}) \times \sqrt{2gH}$
(*d*) none of the above
where H = Difference of liquid level on both sides of the orifice,
H_1 = Height of liquid above top edge orifice on upstream side,
H_2 = Height of liquid above bottom edge of orifice on upstream side.

181. Notch is a device used for measuring
(*a*) rate of flow through pipes
(*b*) rate of flow through a small channel
(*c*) velocity through a pipe
(*d*) velocity through a small channel.

182. The discharge through a rectangular notch is given by
(*a*) $Q = \frac{2}{3}C_d \times L \times H^{5/2}$
(*b*) $Q = \frac{2}{3}C_d \times L \times H^{3/2}$
(*c*) $Q = \frac{8}{15}C_d \times L \times H^{5/2}$
(*d*) $\frac{8}{15}Q = \frac{8}{15}C_d \times L \times H^{3/2}$.

183. The discharge through a triangular notch is given by
(*a*) $Q = \frac{2}{3}C_d \times \tan\frac{\theta}{2} \times \sqrt{2gH}$
(*b*) $Q = \frac{2}{3}C_d \times \tan\frac{\theta}{2} \times \sqrt{2g} \times H^{3/2}$
(*c*) $Q = \frac{2}{15}C_d \times \tan\frac{\theta}{2} \times \sqrt{2g}\ H^{5/2}$
(*d*) none of the above.
where θ = Total angle of triangular notch,
H = Head over notch.

184. The discharge through a trapezoidal notch is given as
(*a*) $Q = \frac{2}{3}C_{d_1} \times L \times H^{3/2} + \frac{8}{15} \times C_{d_2} \times \tan\theta/2 \times \sqrt{2g} \times H^{3/2}$
(*b*) $Q = \frac{2}{3}C_{d_1} \times L \times H^{5/2} + \frac{8}{15} \times C_{d_2} \times \tan\theta/2 \times \sqrt{2g}H^{3/2}$
(*c*) $Q = \frac{2}{3}C_{d_1} \times L \times H^{3/2} + \frac{8}{15} \times C_{d_2} \times \tan\theta/2 \times \sqrt{2g}H^{5/2}$
(*d*) none of the above
where $\theta/2$ = Slope of the side of the trapezoidal notch.

185. The error in discharge due to the error in the measurement of head over a rectangular notch is given by
(*a*) $\frac{dQ}{Q} = \frac{5}{2}\frac{dH}{H}$ (*b*) $\frac{dQ}{Q} = \frac{3}{2}\frac{dH}{H}$
(*c*) $\frac{dQ}{Q} = \frac{7}{2}\frac{dH}{H}$ (*d*) $\frac{dQ}{Q} = \frac{1}{2}\frac{dH}{H}$.

186. The error in discharge due to the error in the measurement of head over a triangular notch is given by
(*a*) $\frac{dQ}{Q} = \frac{5}{2}\frac{dH}{H}$ (*b*) $\frac{dQ}{Q} = \frac{3}{2}\frac{dH}{H}$
(*c*) $\frac{dQ}{Q} = \frac{7}{2}\frac{dH}{H}$ (*d*) $\frac{dQ}{Q} = \frac{1}{2}\frac{dH}{H}$.

187. The velocity with which the water approaches a notch is called
(*a*) velocity of flow
(*b*) velocity of approach
(*c*) velocity of whirl
(*d*) none of the above.

188. The discharge over a rectangular notch considering velocity of approach is given as
(*a*) $Q = \frac{2}{3} C_d L \sqrt{2g}\,(H^{3/2} - h_a^{3/2})$
(*b*) $Q = \frac{2}{3} C_d L \sqrt{2g}\,(H - h_a)^{3/2}$
(*c*) $Q = \frac{2}{3} C_d L \sqrt{2g}\,[(H + h_a)^{3/2} - h_a^{3/2}]$
(*d*) none of the above.
where H = Head over notch, and h_a = Head due to velocity of approach.

189. The velocity of approach (V_a) is given by
(*a*) $V_a = \frac{\text{Discharge over notch}}{\text{Area of notch}}$
(*b*) $V_a = \frac{\text{Discharge over notch}}{\text{Area of channel}}$
(*c*) $V_a = \frac{\text{Discharge over notch}}{\text{Head over notch} \times \text{Width of channel}}$
(*d*) none of the above.

190. Francis's formula for a rectangular weir with end contraction suppressed is given as
(*a*) $Q = 1.84\, L\, H^{5/2}$ (*b*) $Q = \frac{2}{3} L \times H^{3/2}$
(*c*) $Q = 1.84\, L\, H^{3/2}$ (*d*) $Q = \frac{2}{3} L \times H^{5/2}$.

191. Francis's formula for a rectangular weir for two end contractions is given by
(*a*) $Q = 1.84[L - 0.2H]H^{5/2}$
(*b*) $Q = 1.84[L - 0.2H]H^{3/2}$
(*c*) $Q = 1.84[L - 0.2H]H^{5/2}$
(*d*) none of the above.

192. Bazin's formula for discharge over a rectangular weir without velocity of approach is given by
(*a*) $Q = mL \times \sqrt{2g}\, H^{5/2}$
(*b*) $Q = mL \times \sqrt{2g} \times H^{3/2}$
(*c*) $Q = m \times L \times \sqrt{2gH}$
(*d*) none of the above.
where $m = 0.405 + \frac{0.003}{H}$ and H = Head over weir.

193. Cipolletti weir is a trapezoidal weir having side slope of
(*a*) 1 horizontal to 2 vertical
(*b*) 4 horizontal to 1 vertical
(*c*) 1 horizontal to 4 vertical
(*d*) 1 horizontal to 3 vertical.

194. The co-efficient of friction in terms of shear stress is given by
(*a*) $f = \frac{2\rho V^2}{\tau_0}$ (*b*) $f = \frac{2\tau_0}{\rho V^2}$
(*c*) $f = \frac{\tau_0}{2\rho V^2}$ (*d*) $f = \frac{\rho V^2}{2\tau_0}$

195. When the pipes are connected in parallel, the total loss of head
(*a*) is equal to the sum of the loss of head in each pipe
(*b*) is same as in each pipe
(*c*) is equal to the reciprocal of the sum of loss of head in each pipe
(*d*) none of the above.

196. L_1, L_2, L_3 and the length of three pipes, connected in series. If d_1, d_2 and d_3 are their diameters, then the equivalent size of the pipe is given by
(*a*) $\frac{L}{d^5} = \frac{L_1}{d_1^5} + \frac{L_2}{d_2^5} + \frac{L_3}{d_3^5}$
(*b*) $\frac{d^5}{L} = \frac{d_1^5}{L_1} + \frac{d_2^5}{L_2} + \frac{d_3^5}{L_3}$
(*c*) $Ld^5 = L_1 d_2^5 + L_2 d_2^5 + L_3 d_3^5$
(*d*) none of the above.

197. The power transmitted through pipe is given by
(*a*) $\frac{\rho g \times Q \times H}{1000}$
(*b*) $\frac{\rho g \times Q \times h_f}{1000}$
(*c*) $\frac{\rho g \times Q \times (H - h_f)}{4500}$
(*d*) $\frac{\rho g \times Q \times (H - h_f)}{1000}$
where H = Total head at the inlet of pipe, h_f = Head lost due to friction in pipe and Q = discharge per second.

198. Efficiency of power transmission through pipe is given by

(*a*) $\frac{H - h_f}{H}$ (*b*) $\frac{H}{H + h_f}$

(*c*) $\frac{H - h_f}{H + h_f}$ (*d*) none of the above

where H = Total head at inlet, h_f = Head lost due to friction.

199. Maximum efficiency of power transmission through pipe is

(*a*) 50% (*b*) 66.67%

(*c*) 75% (*d*) 100%.

200. For a viscous flow through circular pipes, certain curves are shown in Fig. 2, curve *A* is for

(*a*) shear stress distribution

(*b*) velocity distribution

(*c*) pressure distribution

(*d*) none of the above.

201. Curve *B* in Fig. 2 is for

(*a*) shear stress distribution

(*b*) velocity distribution

(*c*) pressure distribution

(*d*) none of the above.

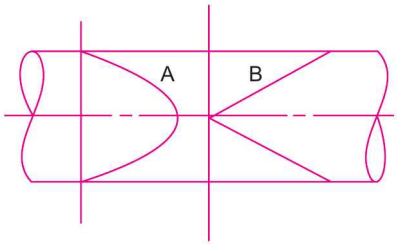


Fig. 2

202. Fig. 3 shows four curves for velocity distribution across a section for Reynolds number equal to 1000, 4000, 6000 and 10000. Curve *A* corresponds to Reynolds number equal to

(*a*) 1000 (*b*) 4000

(*c*) 6000 (*d*) 10000.

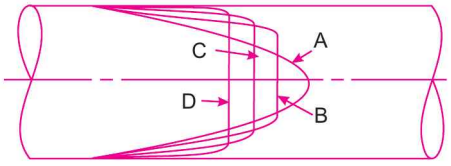


Fig. 3

203. Curve *B* in Fig. 3 corresponds to Reynolds number

(*a*) 1000 (*b*) 4000

(*c*) 6000 (*d*) 10000.

204. Curve *C* in Fig. 3 corresponds to the Reynolds number

(*a*) 1000 (*b*) 4000

(*c*) 6000 (*d*) 10000.

205. Curve *D* in Fig. 3 corresponds to the Reynold number

(*a*) 1000 (*b*) 4000

(*c*) 6000 (*d*) 10000.

206. The shear stress distribution across a section of a circular pipe, having viscous flow is given by

(*a*) $\tau = \frac{\partial p}{\partial x} r^2$ (*b*) $\tau = \frac{\partial p}{\partial x} \frac{r}{2}$

(*c*) $\tau = -\frac{\partial p}{\partial x} \frac{r}{2}$ (*d*) $\tau = -\frac{\partial p}{\partial x} \times 2r.$

207. The velocity distribution across a section of a circular pipe having viscous flow is given by

(*a*) $u = U_{max}\left[1 - \left(\frac{r}{R}\right)^2\right]$

(*b*) $u = U_{max}\,[R^2 - r^2]$

(*c*) $u = U_{max}\left[1 - \frac{r}{R}\right]^2$

(*d*) none of the above.

208. The velocity distribution across a section of two fixed parallel plates having viscous flow is given by

(*a*) $u = \frac{1}{2\mu}\left(-\frac{\partial p}{\partial x}\right)(t^2 - y^2)$

(*b*) $u = \frac{1}{2\mu}\frac{\partial p}{\partial x}\left[t_y - y^2\right]$

(*c*) $u = \frac{1}{2\mu}\frac{\partial p}{\partial x}\,[y - ty]$

(*d*) $u = -\frac{1}{2\mu}\frac{\partial p}{\partial x}[t - y^2]$

where t = Distance between two plates and y is measured from the lower plate.

209. The shear stress distribution across a section of two fixed parallel plates having viscous flow is given by

(*a*) $\tau = -\frac{1}{2}\frac{\partial p}{\partial x}[t^2 - y^2]$

(*b*) $\tau = -\frac{1}{2}\frac{\partial p}{\partial x}[t - 2y]$

(*c*) $\tau = -\frac{1}{2}\frac{\partial p}{\partial x}[ty - y^2]$

(*d*) $\tau = \frac{1}{2}\frac{\partial p}{\partial x}[y - ty]$

where t = Distance between two parallel plates and y is measured from the plate.

210. The ratio of inertia force to viscous force is known as
(*a*) Reynolds number (*b*) Froude number
(*c*) Mach number (*d*) Euler number.

211. The square root of the ratio of inertia force to gravity force is called
(*a*) Reynolds number (*b*) Froude number
(*c*) Mach number (*d*) Euler number.

212. The square root of the ratio of inertia force to force due to compressibility is known as
(*a*) Reynolds number (*b*) Froude number
(*c*) Mach number (*d*) Euler number.

213. The square root of the ratio of inertia force to pressure force is known as
(*a*) Reynolds number (*b*) Froude number
(*c*) Mach number (*d*) Euler's number.

214. Model analysis of pipes flow are based on
(*a*) Reynolds number (*b*) Froude number
(*c*) Mach number (*d*) Euler number.

215. Model analysis of free surface flows are based on
(*a*) Reynolds number (*b*) Froude number
(*c*) Mach number (*d*) Euler number.

216. Model analysis of aeroplanes and projectile moving at supersonic speed based on
(*a*) Reynolds number (*b*) Froude number
(*c*) Mach number (*d*) Euler number.

217. The boundary-layer takes place
(*a*) for ideal fluids
(*b*) for pipe-flow only
(*c*) for real fluids
(*d*) for flow over flat plate only.

218. The boundary layer is called turbulent boundary layer if
(*a*) Reynolds number is more than 2000
(*b*) Reynolds number is more than 4000
(*c*) Reynolds number is more than 5×10^5
(*d*) none of the above.

219. Laminar sub-layer exists in
(*a*) laminar boundary layer region
(*b*) turbulent boundary layer region
(*c*) transition zone
(*d*) none of the above.

220. The thickness of laminar boundary layer at a distance x from the leading edge over a flat plate varies as
(*a*) $x^{4/5}$ (*b*) $x^{1/2}$
(*c*) $x^{1/5}$ (*d*) $x^{3/5}$.

221. The thickness of turbulent boundary layer at a distance x from the leading edge over a flat plate varies as
(*a*) $x^{4/5}$ (*b*) $x^{1/2}$
(*c*) $x^{1/5}$ (*d*) $x^{3/5}$.

222. The separation of boundary layer takes place in case of
(*a*) negative pressure gradient
(*b*) positive pressure gradient
(*c*) zero pressure gradient
(*d*) none of the above.

223. The velocity profile for turbulent boundary layer is

(*a*) $\frac{u}{U} = \sin\left(\frac{\pi}{2}\frac{y}{\delta}\right)$

(*b*) $\frac{u}{U} = \left(\frac{y}{\delta}\right)^{4/7}$

(*c*) $\frac{u}{U} = 2\left(\frac{y}{\delta}\right) - \left(\frac{y}{\delta}\right)^2$

(*d*) $\frac{u}{U} = \frac{3}{2}\left(\frac{y}{\delta}\right) - \frac{1}{2}\left(\frac{y}{\delta}\right)^3$.

224. The drag force exerted by a fluid on a body immersed in the fluid is due to
(*a*) pressure and viscous force
(*b*) pressure and gravity force
(*c*) pressure and turbulence force
(*d*) none of the above.

225. For supersonic flow, if the area of flow increases then
(*a*) velocity decreases
(*b*) velocity increases
(*c*) velocity is constant
(*d*) none of the above.

226. The area velocity relationship for compressible fluids is

(*a*) $\frac{dA}{A} = \frac{dV}{A}[1 - M^2]$ (*b*) $\frac{dA}{A} = \frac{dV}{V}[M^2 - 1]$

(*c*) $\frac{dA}{A} = \frac{dV}{V}[1 - V^2]$ (*d*) $\frac{dA}{A} = \frac{dV}{V}[C^2 - 1]$.

227. The flow in open channel is laminar if the Reynolds number is
(*a*) 2000 (*b*) less than 2000
(*c*) less than 500 (*d*) none of the above.

228. The flow in open channel is turbulent if the Reynolds number is
(*a*) 2000 (*b*) more than 2000
(*c*) more than 4000 (*d*) 4000.

229. If the Froude number in open channel flow is less than 1.0, the flow is called
(*a*) critical flow
(*b*) super-critical flow
(*c*) sub-critical flow
(*d*) none of the above.

230. If the Froude number in open channel flow is equal to 1.0, the flow is called
(*a*) critical flow (*b*) streaming flow
(*c*) shooting (*d*) none of the above.

231. If the Froude number in open channel flow is more than 1.0, the flow is called
(*a*) critical flow (*b*) streaming flow
(*c*) shooting flow (*d*) none of the above.

232. Chezy's formula is given as
(*a*) $V = i\sqrt{mC}$ (*b*) $V = C\sqrt{mi}$
(*c*) $V = m\sqrt{Ci}$ (*d*) none of the above.

233. The discharge through a rectangular channel is maximum when
(*a*) $m = \frac{d}{3}$ (*b*) $m = \frac{d}{2}$
(*c*) $m = 2d$ (*d*) $m = \frac{3d}{2}$
where m = Hydraulic mean depth, d = Depth of flow.

234. The discharge through a trapezoidal channel is maximum when
(*a*) half of top width = sloping side
(*b*) top width = half of sloping side
(*c*) top width = 1.5 × sloping side
(*d*) none of the above.

235. The maximum velocity through a circular channel takes place when depth of flow is equal to
(*a*) 0.95 times the diameter
(*b*) 0.5 times the diameter
(*c*) 0.81 times the diameter
(*d*) 0.5 times the diameter.

236. The maximum discharge through a circular channel takes place when depth of flow is equal to
(*a*) 0.95 times the diameter
(*b*) 0.3 times the diameter
(*c*) 0.81 times the diameter
(*d*) 0.5 times the diameter.

237. Specific energy of a flowing fluid per unit weight is equal to
(*a*) $\frac{p}{w} + \frac{V^2}{2g}$ (*b*) $\frac{p}{w} + h$
(*c*) $\frac{V^2}{2g} + h$ (*d*) $\frac{p}{w} + \frac{V^2}{2g} + h$.

238. The depth of flow after hydraulic jump is
(*a*) $d_2 = \frac{d_1}{2}[\sqrt{1 + 8(F_e)_1^2} - 1]$
(*b*) $d_2 = \frac{d_1}{2}[1 + \sqrt{8(F_e)_1^2} - 1]$
(*c*) $d_2 = \frac{d_1}{2} + \sqrt{\frac{d_1^2}{4} + 8(F_e)_1}$
(*d*) none of the above.

239. The depth of flow at which specific energy is minimum is called
(*a*) normal depth (*b*) critical depth
(*c*) alternate depth (*d*) none of the above.

240. The critical depth (h_c) is given by
(*a*) $\left(\frac{q^2}{g}\right)^{1/2}$ (*b*) $\left(\frac{q}{g}\right)^{1/3}$
(*c*) $\left(\frac{q^2}{g}\right)^{1/3}$ (*d*) $\left(\frac{q^2}{g}\right)^{2/3}$
where q = Rate of flow per unit width of channel.

241. For a circular channel, the wetted perimeter is given by
(*a*) $\frac{R\theta}{2}$ (*b*) $3R\theta$
(*c*) $2R\theta$ (*d*) $R\theta$
where R = Radius of circular channel and θ = half the angle subtended by the water surface at the centre.

242. For a circular channel the area of flow is given by
(*a*) $R^2\left(2\theta - \frac{\sin 2\theta}{2}\right)$ (*b*) $R^2\left(\theta - \frac{\sin 2\theta}{2}\right)$
(*c*) $R^2(\theta - \sin 2\theta)$ (*d*) none of the above
where θ = Half the angle subtended by water surface at the centre, and R = Radius of circular channel.

243. The hydraulic mean depth is given by

(*a*) $\frac{P}{A}$ (*b*) $\frac{P^2}{A}$

(*c*) $\frac{A}{P}$ (*d*) $\sqrt{\frac{A}{P}}$

where A = Area , and P = Wetted perimeter.

244. A most economical section is one which for a given cross-sectional area, slope of bed (i) and co-efficient of resistance has

(*a*) maximum wetted perimeter
(*b*) maximum discharge
(*c*) maximum depth of flow
(*d*) none of the above.

245. Specific speed of a turbine is defined as the speed of the turbine which

(*a*) produces unit power at unit head
(*b*) produces unit horse power at unit discharge
(*c*) delivers unit discharge at unit head
(*d*) delivers unit discharge at unit power.

246. A pump is defined as a device which converts

(*a*) Hydraulic energy into mechanical energy
(*b*) Mechanical energy into hydraulic energy
(*c*) Kinetic energy into mechanical energy
(*d*) None of the above.

247. A turbine is a device which converts

(*a*) Hydraulic energy into mechanical energy
(*b*) Mechanical energy into hydraulic energy
(*c*) Kinetic energy into mechanical energy
(*d*) Electrical energy into mechanical energy.

248. The force exerted by a jet of water on a stationary vertical plate in the direction of jet is given by

(*a*) $F_x = \rho AV^2 \sin^2 \theta$

(*b*) $F_x = \rho AV^2 \ [1 + \cos \theta]$

(*c*) $F_x = \rho AV^2$

(*d*) none of the above.

249. The force exerted by a jet of water on a stationary inclined plate in the direction of jet is given by

(*a*) $F_x = \rho AV^2$

(*b*) $F_x = \rho AV^2 \sin^2 \theta$

(*c*) $F_x = \rho AV^2 \ [1 + \cos \theta]$

(*d*) $F_x = \rho AV^2 \ [1 + \sin \theta]$.

250. The force exerted by a jet of water on a stationary curved plate in the direction of jet is equal to

(*a*) ρAV^2

(*b*) $\rho AV^2 \sin^2 \theta$

(*c*) $\rho AV^2 \ (1 + \cos \theta)$

(*d*) $\rho AV^2 \ [1 + \sin \theta]$.

251. The force exerted by a jet of water having velocity V on a vertical plate, moving with a velocity u is given by

(*a*) $F_x = \rho A(V - u)^2 \sin^2 \theta$
(*b*) $F_x = \rho A(V - u)^2$
(*c*) $F_x = \rho A(V - u)^2 \ [1 + \cos \theta]$
(*d*) None of the above.

252. The force exerted by a jet of water having velocity V on a series of vertical plates moving with velocity u is given by

(*a*) $F_x = \rho AV^2$ (*b*) $F_x = \rho A \ (V - u)^2$
(*c*) $F_x = \rho AVu$ (*d*) None of the above.

253. Efficiency of the jet of water having velocity V striking a series of vertical plates moving with a velocity u is given by

(*a*) $\eta = \frac{2V \ (V - u)}{u^2}$ (*b*) $\eta = \frac{2u(V - u)}{V^2}$

(*c*) $\eta = \frac{u^2}{V^2 \ (V - u)}$ (*d*) None of the above.

254. Efficiency, of the jet of water having velocity V and striking a series of vertical plates moving with a velocity u, is maximum when

(*a*) $u = 2V$ (*b*) $u = \frac{V}{2}$

(*c*) $u = \frac{3V}{2}$ (*d*) $u = \frac{4V}{3}$.

255. Maximum efficiency of a series of vertical plates is

(*a*) 66.67% (*b*) 33.33%
(*c*) 50% (*d*) 80%.

256. For a series of curved radial vanes, the work done per second per unit weight is equal to

(*a*) $\frac{1}{g} V_{W_1} u_1 + V_{W_2} u_2$ (*b*) $\frac{1}{g}[V_1 u_1 + V_2 u_2]$

(*c*) $\frac{1}{g}[V_{W_1} u_1 \pm V_{W_2} u_2]$ (*d*) none of the above.

257. The net head (H) on the turbine is given by

(*a*) H = Gross Head + Head lost due to friction
(*b*) H = Gross Head – Head lost due to friction

(*c*) H = Gross Head + $\frac{V^2}{2g}$ – Head lost due to friction.

258. Hydraulic efficiency of a turbine is defined as the ratio of
(*a*) Power available at the inlet of turbine to power given by water to the runner
(*b*) Power at the shaft of the turbine to power given by water to the runner
(*c*) Power at the shaft of the turbine to the power at the inlet of turbine
(*d*) None of the above.

259. Mechanical efficiency of a turbine is the ratio of
(*a*) Power at the inlet to the power at the shaft of turbine
(*b*) Power at the shaft to the power given to the runner
(*c*) Power at the shaft to the power at the inlet of turbine
(*d*) None of the above.

260. The overall efficiency of a turbine is the ratio of
(*a*) Power at the inlet of turbine to the power at the shaft
(*b*) Power at the shaft to the power given to the runner
(*c*) Power at the shaft to the power at the inlet of turbine
(*d*) None of the above.

261. The relation between hydraulic efficiency (η_h), mechanical efficiency (η_m) and overall efficiency (η_o) is
(*a*) $\eta_h = \eta_o \times \eta_m$ (*b*) $\eta_o = \eta_h \times \eta_m$
(*c*) $\eta_o = \frac{\eta_m}{\eta_h}$ (*d*) none of the above.

262. A turbine is called impulse if at the inlet of the turbine
(*a*) total energy is only kinetic energy
(*b*) total energy is only pressure energy
(*c*) total energy is the sum of kinetic energy and pressure energy
(*d*) none of the above.

263. A turbine is called reaction turbine if at the inlet of the turbine the total energy is
(*a*) kinematic energy only
(*b*) kinetic energy and pressure energy
(*c*) pressure energy only
(*d*) none of the above.

264. Which of the following statement is correct?
(*a*) Pelton wheel is a reaction turbine
(*b*) Pelton wheel is a radial flow turbine
(*c*) Pelton wheel is an impulse turbine
(*d*) None of the above.

265. Francis turbine is
(*a*) an impulse turbine
(*b*) a radial flow impulse turbine
(*c*) an axial flow turbine
(*d*) a reaction radial flow turbine.

266. Kaplan turbine is
(*a*) an impulse turbine
(*b*) a radial flow impulse turbine
(*c*) an axial flow reaction turbine
(*d*) a radial flow reaction turbine.

267. Jet ratio (m) is defined as the ratio of
(*a*) diameter of jet of water to diameter of Pelton wheel
(*b*) velocity of vane to the velocity of jet of water
(*c*) velocity of flow to the velocity of jet of water
(*d*) diameter of Pelton wheel to diameter of the jet of water.

268. Flow ratio is defined as the ratio of
(*a*) Velocity of flow at inlet to the velocity given by $\sqrt{2gH}$
(*b*) Velocity of runner at inlet to the velocity of flow at inlet
(*c*) Velocity of runner to the velocity given by $\sqrt{2gH}$
(*d*) None of the above.

269. Speed ratio is given by
(*a*) $\frac{u}{\sqrt{2gH}}$ (*b*) $\frac{V_f}{\sqrt{2gH}}$
(*c*) $\frac{\sqrt{2gH}}{V_f}$ (*d*) $\frac{V_w}{\sqrt{2gH}}$.

270. The speed ratio for Pelton wheel varies from
(*a*) 0.45 to 0.50 (*b*) 0.6 to 0.7
(*c*) 0.3 to 0.4 (*d*) 0.8 to 0.9.

271. The discharge through Pelton Turbine is given by
(*a*) $Q = \pi DB\, V_f$
(*b*) $Q = \frac{\pi}{4} d^2 \times \sqrt{2gH}$
(*c*) $Q = \frac{\pi}{4}\left[D_o^2 - D_b^2\right] \times V_f$
(*d*) None of the above.

272. The discharge through Francis Turbine is given by
(*a*) $Q = \pi DB\, V_f$
(*b*) $Q = \frac{\pi}{4} d^2 \times \sqrt{2gH}$
(*c*) $Q = \frac{\pi}{4}\left[D_o^2 - D_b^2\right]$
(*d*) None of the above.

273. The discharge through Kaplan turbine is given by
(*a*) $Q = \pi DB\, V_f$ (*b*) $Q = \frac{\pi}{4} d^2 \times \sqrt{2gH}$
(*c*) $Q = \frac{\pi}{4}\left[D_o^2 - D_b^2\right]$ (*d*) $Q = 0.9\, \pi DB\, V_f$.

274. Draft tube is used for discharging water from the exit of
(*a*) an impulse turbine (*b*) a Francis turbine
(*c*) a Kaplan turbine (*d*) a Pelton wheel.

275. Specific speed of a turbine is defined as the speed at which the turbine runs when
(*a*) working under unit head and discharging one litre per second
(*b*) working under unit head and develops unit horse power
(*c*) develops unit horse power and discharges one litre per second
(*d*) none of the above.

276. The specific speed (N_s) of a turbine is given by
(*a*) $N_s = \frac{N\sqrt{P}}{H^{3/4}}$ (*b*) $N_s = \frac{N\sqrt{Q}}{H^{3/4}}$
(*c*) $N_s = \frac{N\sqrt{P}}{H^{5/4}}$ (*d*) $N_s = \frac{NP^{5/4}}{\sqrt{H}}$.

277. Unit speed is the speed of a turbine when it is working
(*a*) under unit head and develops unit power
(*b*) under unit head and discharge one m^3/sec
(*c*) under unit head
(*d*) none of the above.

278. Unit discharge is the discharge of a turbine when
(*a*) the head on turbine is unity and it develops unit power
(*b*) the head on turbine is unity and it moves at unit speed
(*c*) the head on the turbine is unity
(*d*) none of the above.

279. Unit power is the power developed by a turbine when
(*a*) head on turbine is unity and discharge is also unity
(*b*) head = one metre and speed is unity
(*c*) head on turbine is unity
(*d*) none of the above.

280. The unit speed (N_u) is given by the expression
(*a*) $N_u = \frac{N}{H^{3/2}}$ (*b*) $N_u = \frac{N}{H^{3/4}}$
(*c*) $N_u = \frac{N}{\sqrt{H}}$ (*d*) $N_u = \frac{N}{H^{5/4}}$.

281. The unit discharge (Q_u) is given by the expression
(*a*) $Q_u = \frac{Q}{\sqrt{H}}$ (*b*) $Q_u = \frac{Q}{H^{3/2}}$
(*c*) $Q_u = \frac{Q}{H^{3/4}}$ (*d*) $Q_u = \frac{Q}{H^{5/4}}$.

282. Unit power (P_u) is given by the expression
(*a*) $P_u = \frac{P}{\sqrt{H}}$ (*b*) $P_u = \frac{P}{H^{3/2}}$
(*c*) $P_u = \frac{P}{H^{3/4}}$ (*d*) $P_u = \frac{P}{H^{5/4}}$.

283. The unit discharge (Q_u) and unit speed (N_u) curves for different turbines are shown in Fig. 4. Curve *A* is for
(*a*) Francis Turbine
(*b*) Kaplan Turbine
(*c*) Pelton Turbine
(*d*) Propeller Turbine.

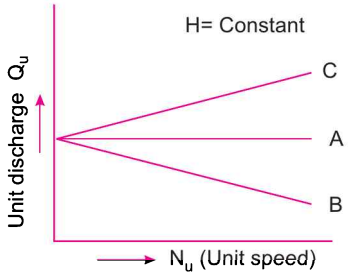


Fig. 4

284. Curve *B* in Fig. 4 is for
(*a*) Francis Turbine
(*b*) Kaplan Turbine
(*c*) Pelton Turbine
(*d*) Propeller Turbine.

285. Curve *D* in Fig. 4 is for
(*a*) Francis Turbine
(*b*) Kaplan Turbine
(*c*) Pelton Turbine
(*d*) Propeller Turbine.

286. Tick mark the correct statement
(*a*) curves at constant speed are called main characteristic curves
(*b*) curves at constant head are called main characteristic curves
(*c*) curves at constant efficiency are called operating characteristic curves
(*d*) curves at constant efficiency are called main characteristic curves.

287. Main characteristic curves of a turbine means
(*a*) curves at constant speed
(*b*) curves at constant efficiency
(*c*) curves at constant head
(*d*) none of the above.

288. Operating characteristic curves of a turbine means
(*a*) curves drawn at constant speed
(*b*) curves drawn at constant efficiency
(*c*) curves drawn at constant head
(*d*) none of the above.

289. Muschel curves means
(*a*) curves at constant head
(*b*) curves at constant speed
(*c*) curves at constant efficiency
(*d*) none of the above.

290. Governing of a turbine means
(*a*) the head is kept constant under all condition of working
(*b*) the speed is kept constant under all conditions
(*c*) the discharge is kept constant under all conditions
(*d*) none of the above.

291. The work done by impeller of a centrifugal pump on water per second per unit weight of water is given by
(*a*) $\frac{1}{g}V_{W_1}u_1$ (*b*) $\frac{1}{g}V_{W_2}u_2$
(*c*) $\frac{1}{g}(V_{W_2}u_2 - V_{W_1}u_1)$ (*d*) none of the above.

292. The manometer head (H_m) of a centrifugal pump is given by
(*a*) Pressure head at outlet of pump – pressure head at inlet
(*b*) Total head at inlet – total head at outlet
(*c*) Total head at outlet – total head at inlet
(*d*) None of the above.

293. The manometric efficiency (η_{man}) of a centrifugal pump is given by
(*a*) $\frac{H_m}{gV_{W_2}u_2}$ (*b*) $\frac{gH_m}{V_{W_2}u_2}$
(*c*) $\frac{V_{W_2}u_2}{gH_m}$ (*d*) $\frac{g \times V_{W_2}u_2}{H_m}$.

294. Mechanical efficiency (η_{mech}) of a centrifugal pump is given by
(*a*) (Power at the impeller)/S.H.P.
(*b*) S.H.P./Power at the impeller
(*c*) Power possessed by water/power at the impeller
(*d*) Power possessed by water/S.H.P.

295. To produce a high head by multistage centrifugal pumps, the impellers are connected
(*a*) in parallel
(*b*) in series
(*c*) in parallel and in series both
(*d*) none of the above.

296. To discharge a large quantity of liquid by multi-stage centrifugal pump, the impellers are connected
(*a*) in parallel
(*b*) in series
(*c*) in parallel and in series
(*d*) none of the above.

297. Specific speed of a pump is the speed at which a pump runs when
(*a*) head developed is unity and discharge is one cubic metre
(*b*) head developed is unity and shaft horse power is also unity
(*c*) discharge is one cubic metre and shaft horse power is unity
(*d*) none of the above.

298. The specific speed (N_s) of a pump is given by the expression
(*a*) $N_s = \frac{N\sqrt{Q}}{H_m^{5/4}}$ (*b*) $N_s = \frac{N\sqrt{P}}{H_m^{3/4}}$
(*c*) $N_s = \frac{N\sqrt{Q}}{H_m^{3/4}}$ (*d*) $N_s = \frac{N\sqrt{P}}{H_m^{5/4}}$.

299. The operating characteristic curves of a centrifugal pump are shown in Fig. 5, curve *A* is for
(*a*) Head (*b*) Efficiency
(*c*) Power (*d*) None of the above.

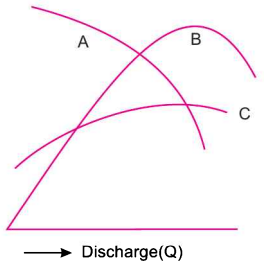


Fig. 5

300. Curve *B* in Fig. 5 is for
(*a*) Head (*b*) Efficiency
(*c*) Power (*d*) None of the above.

301. Curve *C* in Fig. 5 is for
(*a*) Head (*b*) Efficiency
(*c*) Power (*d*) None of the above.

302. Cavitation will take place if the pressure of the flowing fluid at any point is
(*a*) more than vapour pressure of the fluid
(*b*) equal to vapour pressure of the fluid
(*c*) is less than vapour pressure of the fluid
(*d*) none of the above.

303. Cavitation can take place in case of
(*a*) Pelton Wheel
(*b*) Francis Turbine
(*c*) Reciprocating pump
(*d*) Centrifugal pump.

304. Which of the following statement is correct?
(*a*) Centrifugal pump convert mechanical energy into hydraulic energy by sucking liquid into chamber
(*b*) Reciprocating pumps convert mechanical energy into hydraulic energy by means of centrifugal force.
(*c*) Centrifugal pumps convert mechanical energy into hydraulic energy by means of centrifugal force
(*d*) Reciprocating pumps convert hydraulic energy into mechanical energy.

305. The discharge through a single-acting reciprocating pump is
(*a*) $Q = \frac{ALN}{60}$ (*b*) $Q = \frac{2ALN}{60}$
(*c*) $Q = ALN$ (*d*) $Q = 2ALN$.

306. The pressure head due to acceleration (h_a) in reciprocating pump is given by
(*a*) $h_a = \frac{l}{g} \times \frac{a}{A} \times \omega^2 r \sin\theta$
(*b*) $h_a = \frac{l}{g} \times \frac{A}{a} \times \omega^2 r \sin\theta$
(*c*) $h_a = \frac{l}{g} \times \frac{A}{a} \omega^2 r \cos\theta$
(*d*) $h_a = \frac{A}{a} \omega^2 r \sin\theta$
where A = Area of cylinder, a = Area of pipe and r = radius of crank.

307. Indicator diagram shows for one complete revolution of crank the
(*a*) Variation of kinetic head in the cylinder
(*b*) Variation of pressure head in the cylinder
(*c*) Variation of kinetic and pressure head in the cylinder
(*d*) None of the above.

308. Air vessel in a reciprocating pump is used
(*a*) to obtain a continuous supply of water at uniform rate
(*b*) to reduce suction head
(*c*) to increase the delivery head
(*d*) none of the above.

309. The work saved by fitting an air vessel to a single-acting reciprocating pump is
(*a*) 39.2% (*b*) 84.4%
(*c*) 48.8% (*d*) 92.3%.

310. The work saved by fitting an air vessel to a double acting reciprocating pump is
(*a*) 39.2% (*b*) 84.8%
(*c*) 48.8% (*d*) 92.3%.

311. The pressure, at which separation takes place, is known as separation pressure or separation pressure head. For water, the limiting value of separation pressure head is
(*a*) 2.5 m (abs.) (*b*) 7.5 m (abs.)
(*c*) 10.3 m (abs.) (*d*) 5 m (abs.)

312. During suction stroke of a reciprocating pump, the separation may take place
(*a*) at the end of suction stroke
(*b*) in the middle of suction stroke
(*c*) in the beginning of suction stroke
(*d*) none of the above.

313. During delivery stroke of a reciprocating pump, the separation may take place
(*a*) at the end of delivery stroke
(*b*) in the middle of delivery stroke
(*c*) in the beginning of the delivery stroke
(*d*) none of the above.

314. Hydraulic accumulator is a device used for
(*a*) lifting heavy weights
(*b*) storing the energy of a fluid in the form of pressure energy
(*c*) increasing the pressure intensity of a fluid
(*d*) none of the above.

315. Hydraulic intensifier is a device used for
(*a*) storing energy of a fluid in the form of pressure energy
(*b*) increasing pressure intensity of a liquid
(*c*) transmitting power from one shaft to another
(*d*) none of the above.

316. Hydraulic ram is pump which works
(*a*) on the principle of water-hammer
(*b*) on the principle of centrifugal action
(*c*) on the principle of reciprocating action
(*d*) none of the above.

317. Hydraulic coupling is a device used for
(*a*) transmitting same torque to the driven shaft
(*b*) transmitting increased torque to the driven shaft
(*c*) transmitting decreased torque to the driven shaft
(*d*) none of the above.

318. Torque converter is a device used for
(*a*) transmitting same torque to the driven shaft
(*b*) transmitting increased torque to the driven shaft
(*c*) transmitting decreased torque to the driven shaft
(*d*) transmitting increased or decreased torque to the driven shaft.

319. Capacity of a hydraulic accumulator is given as equal to
(*a*) pressure of water supplied by pump × volume of accumulator
(*b*) pressure of water × area of accumulator
(*c*) pressure of water × stroke of the ram of accumulator
(*d*) none of the above.

320. Kaplan turbine is a propeller turbine in which the vanes fixed on the hub are
(*a*) non-adjustable (*b*) adjustable
(*c*) fixed (*d*) none of the above.

321. If the head on the turbine is more than 300 m, the type of turbine used should be
(*a*) Kaplan (*b*) Francis
(*c*) Pelton (*d*) Propeller.

322. If the specific speed of a turbine is more than 300, the type of turbine is
(*a*) Pelton
(*b*) Kaplan
(*c*) Francis
(*d*) Pelton with more jets.

323. Run-away speed of a Pelton wheel means
(*a*) Full load speed
(*b*) No load speed
(*c*) No load speed with no governor mechanism
(*d*) None of the above.

324. Spouting velocity means
(*a*) actual velocity of jet
(*b*) ideal velocity of jet
(*c*) half of ideal velocity of jet
(*d*) none of the above.

325. Surge tank in a pipe line is used to
(*a*) reduce the loss of head due to friction in pipe
(*b*) make the flow uniform in pipe
(*c*) relieve the pressure due to water hammer
(*d*) none of the above.

326. Hydraulic ram is a device used for
(*a*) storing energy of a water in the form of pressure energy
(*b*) increasing pressure intensity of water
(*c*) lifting small quantity of water to a greater height by means of large quantity of water falling through small height
(*d*) none of the above.

327. For low head and high discharge, the suitable turbine is
(*a*) Pelton (*b*) Francis
(*c*) Kaplan (*d*) None of the above.

328. For high head and low discharge, the suitable turbine is
(*a*) Pelton (*b*) Francis
(*c*) Kaplan (*d*) None of the above.

329. The flow of water, leaving the impeller, in a centrifugal pump casing is
(*a*) Forced vortex flow
(*b*) Free vortex flow
(*c*) Centrifugal flow
(*d*) None of the above.

330. Rotameter is used for measuring
(*a*) density of fluids
(*b*) velocity of fluids in pipes
(*c*) discharge of fluids
(*d*) viscosity of fluids.

331. A current meter is a device used for measuring
(*a*) velocity (*b*) viscosity
(*c*) current (*d*) pressure.

332. A hot wire anemometer is a device used for measuring
(*a*) viscosity (*b*) velocity of gases
(*c*) pressure of gases (*d*) none of the above.

ANSWERS

1. (*c*) **2.** (*b*) **3.** (*b*) **4.** (*b*) **5.** (*b*) **6.** (*c*) **7.** (*b*), (*d*) **8.** (*c*) **9.** (*b*) **10.** (*b*)
11. (*d*) **12.** (*b*) **13.** (*b*) **14.** (*c*) **15.** (*b*) **16.** (*d*) **17.** (*a*) **18.** (*c*) **19.** (*c*) **20.** (*b*)
21. (*d*) **22.** (*c*) **23.** (*d*) **24.** (*c*) **25.** (*b*) **26.** (*b*) **27.** (*d*) **28.** (*c*) **29.** (*c*) **30.** (*c*)
31. (*c*) **32.** (*a*) **33.** (*c*) **34.** (*d*) **35.** (*d*) **36.** (*b*),(*c*) **37.** (*c*) **38.** (*b*) **39.** (*c*) **40.** (*d*)
41. (*b*),(*c*) **42.** (*d*) **43.** (*c*) **44.** (*b*) **45.** (*c*) **46.** (*b*) **47.** (*c*) **48.** (*c*) **49.** (*b*) **50.** (*b*)
51. (*c*) **52.** (*d*) **53.** (*c*) **54.** (*a*) **55.** (*d*) **56.** (*d*) **57.** (*a*) **58.** (*c*) **59.** (*b*) **60.** (*c*)
61. (*d*) **62.** (*d*) **63.** (*d*) **64.** (*c*) **65.** (*d*) **66.** (*d*) **67.** (*c*) **68.** (*d*) **69.** (*b*) **70.** (*d*)
71. (*a*) **72.** (*b*) **73.** (*b*) **74.** (*c*) **75.** (*a*) **76.** (*c*) **77.** (*c*) **78.** (*c*) **79.** (*c*) **80.** (*c*)
81. (*d*) **82.** (*a*) **83.** (*b*) **84.** (*b*) **85.** (*b*) **86.** (*c*) **87.** (*b*) **88.** (*c*) **89.** (*a*) **90.** (*b*)
91. (*a*) **92.** (*b*) **93.** (*b*) **94.** (*a*) **95.** (*b*) **96.** (*b*) **97.** (*c*) **98.** (*a*) **99.** (*b*) **100.** (*c*)
101. (*c*) **102.** (*b*) **103.** (*c*) **104.** (*b*) **105.** (*a*) **106.** (*a*) **107.** (*d*) **108.** (*c*) **109.** (*b*) **110.** (*a*)
111. (*c*) **112.** (*d*) **113.** (*d*) **114.** (*b*) **115.** (*b*) **116.** (*c*) **117.** (*d*) **118.** (*c*) **119.** (*b*) **120.** (*c*)
121. (*b*) **122.** (*c*) **123.** (*a*) **124.** (*c*) **125.** (*c*) **126.** (*c*) **127.** (*c*) **128.** (*b*) **129.** (*d*) **130.** (*c*)
131. (*a*) **132.** (*b*) **133.** (*c*) **134.** (*d*) **135.** (*a*) **136.** (*b*) **137.** (*c*) **138.** (*d*) **139.** (*c*) **140.** (*b*)
141. (*d*) **142.** (*c*) **143.** (*b*) **144.** (*d*) **145.** (*b*) **146.** (*d*) **147.** (*d*) **148.** (*c*) **149.** (*c*) **150.** (*b*)
151. (*d*) **152.** (*c*) **153.** (*d*) **154.** (*a*) **155.** (*a*) **156.** (*a*) **157.** (*c*) **158.** (*c*) **159.** (*c*) **160.** (*c*)
161. (*c*) **162.** (*d*) **163.** (*a*) **164.** (*c*) **165.** (*a*) **166.** (*a*) **167.** (*b*) **168.** (*a*) **169.** (*b*) **170.** (*b*)
171. (*c*) **172.** (*b*) **173.** (*d*) **174.** (*c*) **175.** (*d*) **176.** (*b*) **177.** (*a*) **178.** (*c*) **179.** (*c*) **180.** (*b*)
181. (*b*) **182.** (*b*) **183.** (*c*) **184.** (*c*) **185.** (*b*) **186.** (*a*) **187.** (*b*) **188.** (*c*) **189.** (*b*) **190.** (*c*)
191. (*b*) **192.** (*b*) **193.** (*c*) **194.** (*b*) **195.** (*b*) **196.** (*a*) **197.** (*d*) **198.** (*a*) **199.** (*b*) **200.** (*b*)
201. (*a*) **202.** (*a*) **203.** (*b*) **204.** (*c*) **205.** (*d*) **206.** (*c*) **207.** (*a*) **208.** (*b*) **209.** (*b*) **210.** (*a*)
211. (*b*) **212.** (*c*) **213.** (*d*) **214.** (*a*) **215.** (*b*) **216.** (*c*) **217.** (*c*) **218.** (*c*) **219.** (*b*) **220.** (*b*)
221. (*a*) **222.** (*b*) **223.** (*b*) **224.** (*a*) **225.** (*b*) **226.** (*b*) **227.** (*c*) **228.** (*b*) **229.** (*c*) **230.** (*a*)
231. (*c*) **232.** (*b*) **233.** (*b*) **234.** (*a*) **235.** (*c*) **236.** (*a*) **237.** (*c*) **238.** (*a*) **239.** (*b*) **240.** (*c*)
241. (*c*) **242.** (*b*) **243.** (*c*) **244.** (*b*) **245.** (*a*) **246.** (*b*) **247.** (*a*) **248.** (*c*) **249.** (*b*) **250.** (*c*)
251. (*b*) **252.** (*a*) **253.** (*b*) **254.** (*b*) **255.** (*c*) **256.** (*c*) **257.** (*b*) **258.** (*d*) **259.** (*b*) **260.** (*c*)
261. (*b*) **262.** (*a*) **263.** (*b*) **264.** (*c*) **265.** (*d*) **266.** (*c*) **267.** (*d*) **268.** (*a*) **269.** (*a*) **270.** (*a*)
271. (*b*) **272.** (*a*) **273.** (*c*) **274.** (*b*), (*c*) **275.** (*b*) **276.** (*c*) **277.** (*c*) **278.** (*c*) **279.** (*c*)
280. (*c*) **281.** (*a*) **282.** (*b*) **283.** (*c*) **284.** (*a*) **285.** (*b*) **286.** (*b*) **287.** (*c*) **288.** (*a*) **289.** (*c*)
290. (*b*) **291.** (*b*) **292.** (*c*) **293.** (*b*) **294.** (*a*) **295.** (*b*) **296.** (*a*) **297.** (*a*) **298.** (*c*) **299.** (*a*)
300. (*b*) **301.** (*c*) **302.** (*c*) **303.** (*b*), (*d*) **304.** (*c*) **305.** (*a*) **306.** (*c*) **307.** (*b*) **308.** (*a*)
309. (*b*) **310.** (*a*) **311.** (*a*) **312.** (*c*) **313.** (*a*) **314.** (*b*) **315.** (*b*) **316.** (*a*) **317.** (*a*) **318.** (*d*)
319. (*a*) **320.** (*b*) **321.** (*c*) **322.** (*b*) **323.** (*c*) **324.** (*b*) **325.** (*c*) **326.** (*c*) **327.** (*c*) **328.** (*a*)
329. (*b*) **330.** (*c*) **331.** (*a*) **332.** (*b*).

APPENDIX

(A) Base and Derived Units in MKS and SI Units :

S. No.	*Physical Quantity*	*Unit Symbol in MKS Units*	*Unit Symbol in SI Units*	*Name*
1.	Mass	kg (*M*)	kg(*M*)	kilogram
2.	Length	m (*L*)	m (*L*)	metre
3.	Time	Sec (*t*)	s (*t*)	second
4.	Force	kgf	N	newton
5.	Work, energy	kgf-m	Nm = J	joule
6.	Power	kgf-m/sec	Nm/s =J/s = W	watt

(B) Conversion Factors from MKS to SI Units :

$$1 \text{ kgf} = 9.81 \text{ N}$$

$$1 \text{ kgf-m} = 9.81 \text{ N m} = 9.81 \text{ J}$$

$$1 \text{ kgf-m/sec} = 9.81 \text{ N m/s} = 9.81 \text{ J/s} = 9.81 \text{ W}$$

$$1 \text{ metric h.p.} = 75 \text{ kgf-m/sec} = 75 \times 9.81 \text{ N m/s}$$

$$= 735.75 \text{ N m/sec} \simeq 736 \text{ W} \qquad (\because \text{ N m/s = W})$$

(C) Conversion Factors from MKS to CGS Units :

$$1 \text{ kgf} = 1000 \text{ gmf} = 1000 \times \text{gm} \times \text{g}$$

$$= 1000 \times \text{gm} \times 981 \text{ cm/sec}^2$$

$$= 1000 \times 981 \times \text{gm-cm/sec}^2 = 1000 \times 981 \text{ dyne}$$

$$= 9.81 \times 10^5 \text{ dyne} \qquad (\because \text{ gm-cm/sec}^2 = \text{dyne})$$

$$1 \text{ kgf/m}^2 = 9.81 \times 10^5 \text{ dyne}/10^4 \text{ cm}^2 \qquad (\because \text{ m}^2 = 10^4 \text{ cm}^2)$$

$$= 9.81 \times 10 \text{ dyne/cm}^2 = 98.1 \text{ dyne/cm}^2$$

$$1 \text{ m}^3 = 1000 \text{ litre.}$$

(D) Conversion Factors for Poise and Stoke in MKS and SI Units :

$$\text{Poise} = \frac{1}{98.1} \text{ (MKS unit)} = \frac{1}{10} \text{ (SI unit)}$$

$$\text{Stoke} = 10^{-4} \text{ (MKS unit)} = 10^{-4} \text{ (SI unit)}$$

(E) Important Values in S.I. Units :

$$\text{Pascal (Pa)} = \text{N/m}^2 = \text{kg/ms}^2$$

$$\text{Newton (N)} = \text{kg m/s}^2$$

$$\text{Joule (J)} = \text{N m} = \text{kg m}^2/\text{s}^2.$$

$$\text{Atmospheric pressure} = 101.325 \text{ kN/m}^2$$

$$1 \text{ bar} = 10^5 \text{ Pa} = 10^5 \text{ N/m}^2$$

$$1 \text{ m bar or } 1 \text{ mb} = 10^{-3} \text{ bar} = 10^2 \text{ Pa} = 10^2 \text{ N/m}^2$$

Pascal and bar are used for pressure.

SUBJECT INDEX

R

S

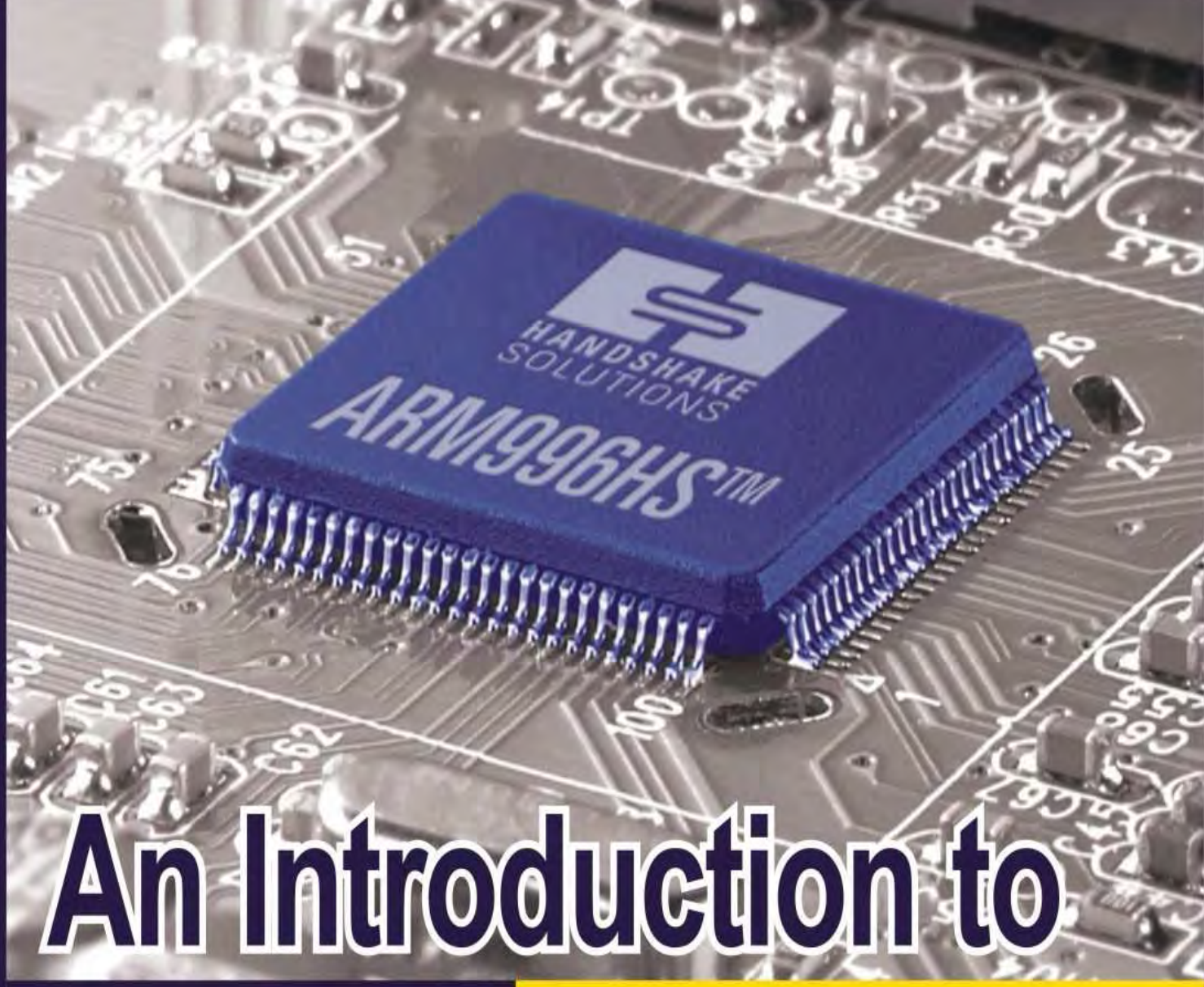
T

U

V

W

Z



An Introduction to Microprocessor 8085

Dr. D.K. Kaushik

DHANPAT RAI PUBLISHING COMPANY

An Introduction To Microprocessor 8085
Dr. D.K. Kaushik

Dr. D. K. Kaushik is presently working as the Principal, Manohar Memorial Post-Graduate College, Fatehabad (Haryana). Earlier, he was the Head, Department of Electronics, Dayanand Post-Graduate College, Hisar. He obtained his master's degree in Physics from Meerut University and Ph.D. in 1981 from Kurukshetra University, Kurukshetra. He has more than 28 years teaching experience in the subject of Electronics and published more than 24 research papers in journals of high repute. He has two patents for indigenous design of Electronic Thin Film Thickness Monitors, to his credit. He had been the Member board of studies in Electronics of Kurukshetra University, Kurukshetra many times. His areas of interest are Semiconductor Electronics, Microprocessor, Computer Architecture and Thin Film Technology.

SOME OTHER USEFUL BOOKS BY THE AUTHOR

Analog Electronics (Circuits and Design).....Dr.D.K.Kaushik

Digital Electronics .....Dr.D.K.Kaushik

Analog Electronic Circuits .....Dr.D.K.Kaushik

DHANPAT RAI PUBLISHING COMPANY (P) LTD.

4779/23, Ansari Road, Darya Ganj, New Delhi - 110002

Phones : 011-23257511, 23257526 Fax : 011-23257525 email : mail@dhanpatrai.in

Website : www.dhanpatrai.in

AN INTRODUCTION TO MICROPROCESSOR 8085

By

Dr. D. K. Kaushik
Principal, Manohar Memorial (P.G.) College,
Fatehabad (Haryana) India

Dhanpat Rai Publishing co., New Delhi

AN INTRODUCTION TO MICROPROCESSOR 8085

1

Evolution and History of Microprocessors

To understand the working and programming of microprocessors, it is very necessary to know the details of evolution and history of microprocessors which will be discussed in this chapter. In the succeeding chapters of this book, the detailed discussion on the basics of microprocessors will be made. For the beginners it is very difficult to understand the operation of a digital computer as it contains large details, so efforts have been made to envisage the operation of a digital computer in step by step manner.

1.1 INTRODUCTION

Microprocessor is a digital device on a chip which can fetch instructions from a memory, decode and execute them i.e. performs certain arithmetic and logical operations, accept data from input device, and send results to output devices. Therefore, a microprocessor interfaced with memory and Input/ Output devices forms a Microcomputer. Basically, there are five building blocks of a digital computer namely:

Input Unit Through this unit data and instructions are fed to the memory of the computer. The basic purpose of this unit is to read the data into the machine. The program from the memory is read into the machine along with the input data which are required to solve or compute the problem by the machine. The typical devices which are used for this purpose are keyboards, paper tape reader and toggle switches etc.

Memory Unit The memory unit of a digital computer consists of devices which are capable of storing information. The memory of a computer is used for storing two distinct type of information such as data to be processed by the computer and program through which the result of the desired problem is obtained. Computer program and data are stored in the Memory Unit. This usually consists of chips of both ROMs (Read Only Memories) and RAMs (Random Access Memories) either bipolar or MOS.

Arithmetic and Logical Unit (ALU)

This unit is used for performing arithmetic operations such as Addition, Subtraction, Multiplications, division and other logical operations on the data.
The control unit guides ALU which of the operations are to be performed. The sequence of the instructions is controlled by the control unit.

Control Unit

The control unit performs the most important function in a computer. It controls all other units and also controls the flow of data from one unit to another for performing computations. It also sequences the operations. It instructs all the units to perform the task in a particular sequence with the help of clock pulses.

Output Unit

After processing of the data in the Arithmetic and Logical Unit, the results are displayed to the output world through this unit. The CRTs (Cathode Ray Tubes), LEDs (Light Emitting Diodes) and Printer etc. form the output unit.

In a computer system ALU and Control Unit are combined in one unit called Central Processing Unit (CPU). The block diagram of a computer is shown in figure 1.1.

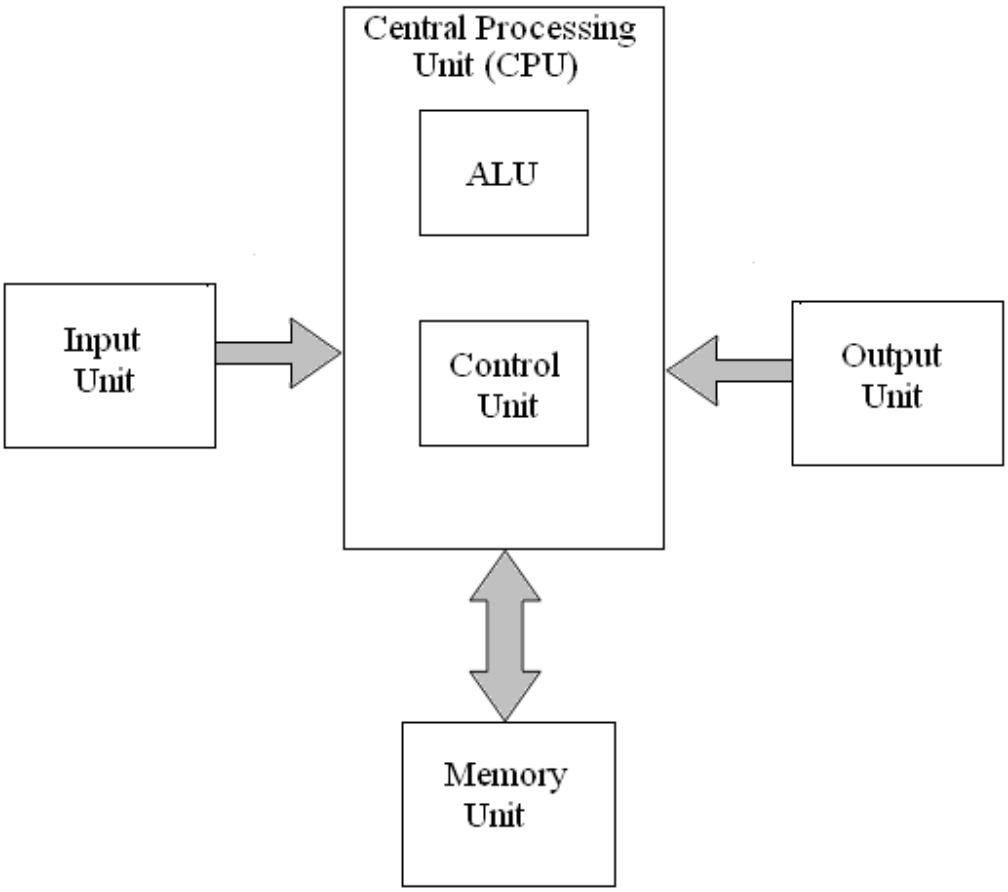


Fig. 1.1

The Central Processing Unit is analogous to the human brain as all the decisions as per the instructions are made by CPU. All other parts are also controlled by this unit. A microprocessor is an integrated circuit designed for use as Central Processing Unit of a computer. The CPU is the primary and central player in communicating with devices such as memory, input and output. However, the timing of communication process is controlled by the group of circuits called control unit.

The term 'Microprocessor' came into existence, in 1971, when the Intel Corporation of America, developed the first microprocessor (INTEL-4004) which is a 4-bit microprocessor (μ**p**). A microprocessor is a programmable digital electronic component that incorporates the functions of a Central Processing Unit (CPU) on single semi-conducting Integrated Circuits (ICs). As such, a system with microprocessor as an integral part is termed as a microprocessor based system. When a computer is microprocessor based, it is called a microcomputer (μ**c**).

A microprocessor is specified by its 'Word Size', e.g. 4-bit, 8-bit, 16-bit etc. By the term 'word size" means the number of bits of data that is processed by the microprocessor as a unit. For example, an 8-bit microprocessor performs various operations on 8-bit data. It also specifies the width of the data bus. As discussed above, a microcomputer consists of input/ output devices, and memory, in addition to microprocessor which acts its CPU. In fact CPU is commonly referred to microprocessor (μ**p**)**.** Microprocessors made possible the advent of the microcomputer in the mid-1970s. Before this period, electronic CPUs were typically made from bulky discrete switching devices. Later on small-scale integrated circuits were used to design the CPUs. By integrating the processor onto one or very few large-scale integrated circuit package (containing the equivalent of thousands or millions of discrete transistors), the cost of processor was greatly reduced.

The evolution of microprocessors has been known to follow Moore's law when it comes to steadily increasing performance over the years. This law suggests that the complexity of an integrated circuit, with respect to minimum component cost, doubles in every 18 months. This dictum has generally proven true since the early 1970's. This lead to the dominance of microprocessors over every other form of computer; every system from the largest mainframes to the smallest hand held computers now uses a microprocessor as its core.

The microprocessor based systems play significant role in the every functioning of industrialized societies. The microprocessor can be viewed as a programmable logic device that can be used to control processes or to turn on/off devices. So the microprocessor can be viewed as a data processing unit or a computing unit of a computer. The microprocessor is a programmable integrated device that has computing and decision making capability similar to that of the central processing unit (CPU) of a computer. Nowadays, the microprocessor is being used in a wide range of products called microprocessor based products or systems. The microprocessor communicates and operates in the binary numbers 0 and 1, called bits. Each microprocessor has a fixed set

of instructions in the form of binary pattern called a machine language. However, it is difficult for human beings to communicate in the language of 0s and 1s. Therefore, the binary instructions are given abbreviated names, called mnemonics, which form the assembly language for a given microprocessor.

1.2 EVOLUTION/HISTORY OF MICROPROCESSORS

An English Mathematician Charles Babbage was the first man to propose the basic principle of modern computers from 1792-1871. He gave the concept of a programmable machine having computer similar to modern digital computers. He is, therefore, known Father of Modern Computers. In 1930s successful general purpose mechanical computer was developed. Before this, mechanical calculators were built to perform simple mathematical operations such as addition, subtraction, multiplication and division. Improvement continued and in 1944 Prof. H. Aiken developed a first practical electro-mechanical digital computer in collaboration with IBM. This computer was known as HAWARD MARK-I. It was in large size (51' long and 8'high) and weighing about 2 tons. The punch cards ware used to input the data in the computer.

During the development of the HAWARD MARK-I Computer, Konard Joos of Germany was busy in developing another computer based on 0's and 1's rather than decimal numbers. So he developed a computer making use of relays (on-off for 1's and 0's) during 1936-44. Joos also developed a language for the computer. The giant machines during 1940-50 were thus designed using relays and vacuum tubes.

In 1945, John J. Mauchy and J. Presper Eckert of University of Pennsylvania developed first electronic computer ENIAC (Electronic Numerical Integrator and Calculator). It was too huge weighing 30 tons and occupied an area $30' X 50'$ and made use of 18000 vacuum tubes, more than 30000 resistors, 10000 capacitors and 6000 switches. It took 200 μS for addition, and 3mS for 10-digit multiplication. It had separate memory for program and data. It used 20 electronic accumulators for memory. Each accumulator stored signed 10-digit decimal number. A number of computers using vacuum tubes were developed during 1940-55. The main drawback with the ENIAC was the life of the vacuum tube components, which required the frequent maintenance.

The invention of semiconductor transistors in 1948 at Bell Laboratories leads further development of computers. The use of semiconductor transistors could not only reduced the size of computers but also increased its capability to a great extent. This leads reduction in cost. Further, the invention of Integrated circuits in 1958 by Jack Kilby of Texas Instrument made a revolution in electronic circuitry. The use of ICs made the size of computers very small and became more versatile in functions. Finally, the advent of IC technology leads to the development of first microprocessor (INTEL 4004) in 1971 at Intel Corporation by an engineer Marcian E. Hoff. It was a 4-bit microprocessor – a programmable controller on a chip. This was called the first generation microprocessor. It was fabricated using P-channel MOSFET technology and had an instruction set of 45 different instructions. It addressed 4096 four-bit wide memory locations. The P-channel MOSFET technology gave low cost but low speed not compatible with TTL (Transistor-

Transistor Logic) technology. It has to use at least 30 ICs to form a system. As INTEL 4004 had very small number of instructions, it could be used in limited applications such as early video games and small microprocessor-based controllers. Seeing microprocessor as a viable product, Intel Corporation released the 8008 microprocessor – an extended 8 bit version of the 4004 microprocessor in 1972. Soon a variety of microprocessors was released by different manufactures. A few first generation microprocessors are listed in table 1.1.

Table 1.1

4-bit Microprocessors	8-bit Microprocessors
INTEL 4004	INTEL 8008
INTEL 4040	NATIONAL IMP-8
FAIRCHILD PPS-25	ROCKWELL PPS-8
ROCKWELL PPS-4	AMI 7200
NATIONAL IMP-4	MOSTEK 5065

Second generation microprocessors appeared in 1973 and used NMOS-technology which offered faster speed, higher density and still better reliability. In the year 1974, an 8-bit microprocessor INTEL 8080 was developed using NMOS technology. It requires only two additional devices to design a functional CPU. It is much faster than 8008 and has more instructions than 8008 that facilitates the programming. The 8080 was compatible with TTL, whereas the 8008 was not directly compatible. The 8080 microprocessor could also address four times more memory (64K bytes) than the 8008 microprocessor (16K bytes). INTEL Corporation in 1977 developed another 8-bit microprocessor 8085 which was proved to be a better version than 8080. The execution time of two 8-bit numbers is 2.0 μS for 8085 whereas it is 1.3 μS for 8080. The main advantages of the 8085 were its internal clock generator, internal system controller, and higher clock frequency. Some of the important second generation microprocessors are given in table 1.2.

Table 1.2

8-bit Microprocessors	12-bit Microprocessors
INTEL 8080	INTERSIL 6100
INTEL 8085	TOSHIBA TLCS-12
FAIRCHILD F8	
MOTOROLA M6800	
ZILOG Z-80	
SIGNETICS 2650	

The advantages of second generation microprocessor are given below:

(i) Larger chip size (170 X 200 mils),
(ii) 40 pins,
(iii) More number of on-chip decoded timing signals,
(iv) Ability to address larger memory space,
(v) Ability to address more I/O Ports,
(vi) More powerful instruction set,
(vii) Faster operation,
(viii) Better Interrupt handling capabilities.

Third generation microprocessors were introduced in 1978. These were 16-bit microprocessors, designed using HMOS (High Density MOS) technology. These microprocessors offered better speed and higher packing density than NMOS. Some important third generation microprocessors are given in table 1.3.

Table 1.3

16-bit Microprocessors		
INTEL 8086 INTEL 8088 INTEL 80186 INTEL 80286	MOTOROLA-68000 MOTOROLA-68010 NATIONAL NS-16016 TEXAS INSTRUMENT-TMS-99000	INTERSIL 6100 TOSHIBA TLCS-12 ZILOG Z-8000

In 1978, 16-bit INTEL 8086 microprocessor of 64 pins was introduced and in 1979 other 16-bit microprocessor 8088 was developed. In addition to the other performances, these µPs contain multiply/divide/arithmetic hardware. The memory addressing capabilities has been increased to very large i.e., 1MB to 16MB through a variety of flexible and powerful addressing mode. The other characteristics of third generation are given below:

(i) These microprocessors were 40/48/64 pins,
(ii) High speed and very strong processing capability,
(iii) Easier to program,
(iv) Allow for dynamically re-locatable programs,
(v) Size of internal registers were 8/16/32 bits,
(vi) These µPs had the multiply/divide/arithmetic hardware,
(vii) Physical memory space was from 1 to 16 Mega-bytes (MB),
(viii) Flexible 10 port addresses,
(ix) More powerful interrupt and hardware capabilities,
(x) Segmented address and virtual memory features.

Fourth generation microprocessors of 32 bits were introduced in the form of 80386 in 1985 and 80486 in 1989. The instruction set of the 80386 microprocessor was upward compatible with the earlier 8086, 8088 and 80286 microprocessors. However, the 80486 is an improved version of the 80386 microprocessor. The 80386 executes many instructions in 2 clock cycles while the 80486 executes in one clock cycle. These microprocessors are of low power version of HMOS technology. Some important fourth generation microprocessors are given in table 1.4.

Table 1.4

32-bit Microprocessors	
INTEL 80386 INTEL 80486 NATIONAL NS16022 MOTOROLA MC 88100	MOTOROLA M-68020 MOTOROLA M-68030 BELLMAC-32

Fifth generation microprocessor was introduced by INTEL Corporation in 1993 in the form of PENTIUM with 64 data bus. The Pentium was similar to the 80386 and 80486 microprocessor. The two introductory versions of the Pentium operated with a clock frequency of 60 MHz and 66 MHz and a speed of 110 MIPS (Million Instructions Per Second). With better and more advanced technologies, the speed of µPs has increased tremendously. The old 8085 of 1977 executed 0.5 million instruction/sec. (0.5 MIPS), while the 80486 executes 54 million instruction per sec.

The Pentium Pro Processor is the Sixth generation microprocessor introduced in 1995 having better architecture but more in size. The Pentium Pro Microprocessor contains 21 million transistors, 3 integer units as well as a floating unit to increase the performance of most software. The basic clock frequency is 150 MHz and 166 MHz.

1.3 BASIC MICROPROCESSOR SYSTEM

The Microprocessor alone does not serve any useful purpose unless it is supported by memory and I/O ports. The combination of memory and I/O ports with microprocessor is known as microprocessor based system. As discussed above the microprocessor which is the central processing unit executes the program stored in the memory and transfer data to and from the outside world through I/O ports. The microprocessor is interconnected with memory and I/O ports by the data bus, the Address bus and the control bus.

A bus is basically a communication link between the processing unit and the peripheral devices as shown in figure 1.2.

1.3.1 Address Bus

The address bus is unidirectional and is to be used by the CPU to send out address of the memory location to be accessed. It is also used by the CPU to select a particular input or output port. It may consist of 8, 12, 16, 20 or even more number of parallel lines. Number of bits in the address bus determines the minimum number of bytes of data in the memory that can be accessed. A 16-bit address bus for instance can access 2^{16} bytes of data. It is labeled as A_0……A_{n-1}, where **n** is the width of bits of the address bus.

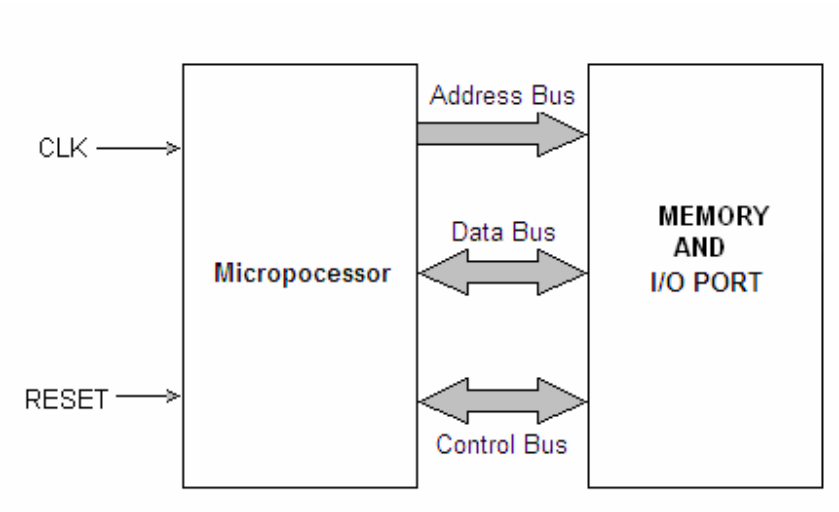


Fig. 1.2

1.3.2 Data Bus

Data bus is bidirectional, that is, data flow occurs both to and from CPU and peripherals. There is an internal data bus which may not be of the same width as the external data bus by that connects the I/O and memory. A microprocessor is characterized by the width of its data bus. All those microprocessors having internal and external data buses of different widths are characterized either by their internal or external data buses. The size of the internal data bus determines the largest number that can be processed by a microprocessor, for instance, having a 16-bit internal data bus is 65536 (64K). The bus is labeled as: D_0 ……$D_{n-1,}$ where n is the data bus width in bits

1.3.3 Control Bus

Control bus contains a number of individual lines carrying synchronizing signals. The control bus sends out control signal to memory, I/O ports and other peripheral devices to ensure proper operation. It carries control signals like MEMORY READ, MEMORY WRITE, READ INPUT PORT, WRITE OUTPUT PORT, HOLD, INTERRUPT etc. For instance, if it is desired to read the contents of a particular memory location, the CPU first sends out address of that very location on the address bus and a 'Memory Read' control signal on the control bus. The memory responds by outputting data stored in the addressed memory location on the data bus.

This book will confine the detailed study of 8085 microprocessor because it is most commonly used microprocessor. However, evolution of microprocessors has been discussed in this chapter, in order to have the knowledge microprocessors introduced so far. Before discussing the details of the 8085, brief discussion of 8-bit microprocessor 8080 is given here.

1.4 BRIEF DESCRIPTION OF 8-BIT MICROPROCESSOR 8080A

Intel 8080 microprocessor is a successor to the Intel 8008 CPU. The Intel 8080/8080A was not object-code compatible with the 8008, but it was source-code compatible with it. The 8080 CPU had the same interrupt processing logic as the 8008, which made porting of old applications easier. Maximum memory size on the Intel 8080 was increased from 16 KB to 64 KB. The number of I/O ports was increased to 256. In addition to all 8008 instructions and addressing modes the 8080 processor included many new instructions and direct addressing mode. The 8080 also included new Stack Pointer (SP) register. The SP was used to specify position of external stack in CPU memory, and the stack could grow as large as the size of memory. Thus, the CPU was no longer limited to 7-level internal stack, like the 8008 did.

The Intel 8080, an 8-bit microprocessor was very popular. Fig. 1.2 shows the shape of the microprocessor and Fig. 1.3 shows the pin out configuration of the 8080A microprocessor. Its salient features include:

a) A two-phase clock input Q1 and Q2
b) 16-bit address bus
c) 8-bit data bus
d) Power supply input of +5V, -5V and +12V required.
e) The 8080A places the status of the operation on the data bus during the earlier part of the cycle and places data on the bus during later part of cycle.

Fig. 1.2

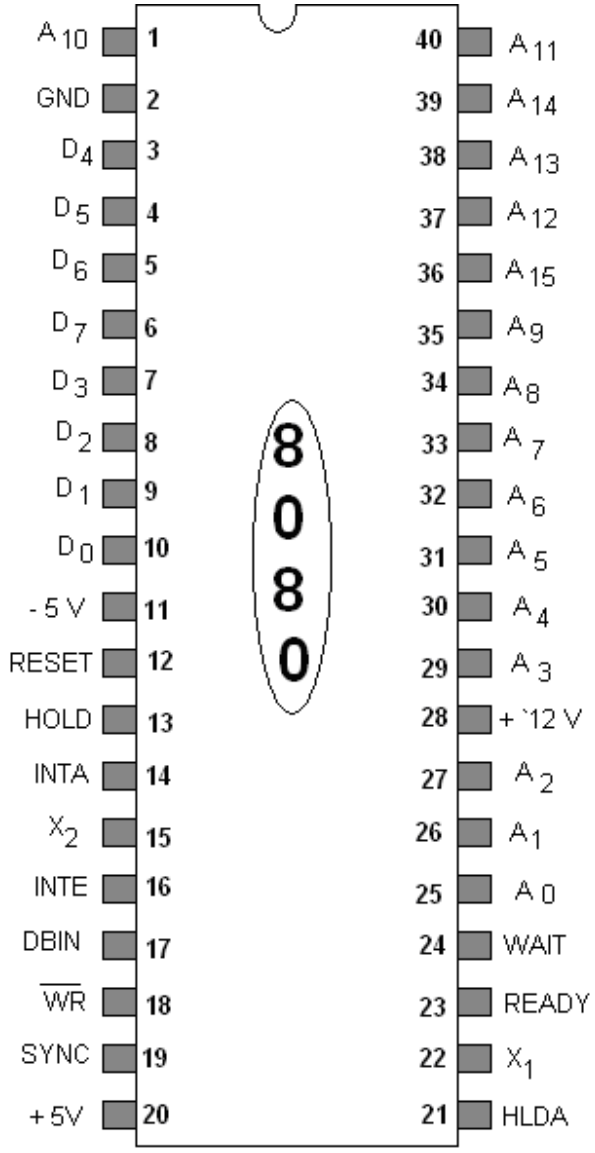


Fig. 1.3

Program memory Pprogram can be located anywhere in memory. Jump, branch and call instructions use 16-bit addresses, i.e. they can be used to jump/branch anywhere within 64 KB. All jump/branch instructions use absolute addressing.

Data memory The processor always uses 16-bit addresses so that data can be placed anywhere.

Stack memory It is limited only by the size of memory. Stack grows downward.

Interrupts The processor supports maskable interrupts. When an interrupt occurs the processor fetches from the bus one instruction, usually one of these instructions:

- One of the 8 RST instructions (RST0 - RST7). The processor saves current program counter into stack and branches to memory location N * 8 (where N is a 3-bit number from 0 to 7 supplied with the RST instruction).
- CALL instruction (3 byte instruction). The processor calls the subroutine, address of which is specified in the second and third bytes of the instruction.

The interrupt can be enabled or disabled using EI (Enable Interrupts) and DI (Disable Interrupts) instructions.

I/O ports

256 Input ports
256 Output ports

Registers

Accumulator or A register is an 8-bit register used for arithmetic, logic, I/O and load/store operations.

Flag is an 8-bit register contains the following five, 1-bit flags:

- Sign flag - set if the most significant bit of the result is set.
- Zero - set if the result is zero.
- Auxiliary carry flag - set if there was a carry out from bit 3 to bit 4 of the result.
- Parity flag - set if the parity (the number of set bits in the result) is even.
- Carry flag - set if there was a carry during addition, or borrow during subtraction/ comparison.

General registers:

- 8-bit B and 8-bit C registers can be used as one 16-bit BC register pair. When used as a pair the C register contains low-order byte. Some instructions may use BC register as a data pointer.
- 8-bit D and 8-bit E registers can be used as one 16-bit DE register pair. When used as a pair the E register contains low-order byte. Some instructions may use DE register as a data pointer.

- 8-bit H and 8-bit L registers can be used as one 16-bit HL register pair. When used as a pair the L register contains low-order byte. HL register usually contains a data pointer used to reference memory addresses.

Stack pointer It is a 16 bit register. This register is always incremented/ decremented by 2.

Program counter It is also a 16-bit register.

Instruction Set

8080 instruction set consists of the following instructions:

- Data moving instructions.
- Arithmetic - add, subtract, increment and decrement.
- Logic - AND, OR, XOR and rotate.
- Control transfer – conditional, unconditional, call subroutine, return from subroutine and restarts.
- Input/Output instructions.
- Other – setting/clearing flag bits, enabling/disabling interrupts, stack operations, etc.

1.5 COMPUTER PROGRAMMING LANGUAGES

In order for computers to accept commands from human and perform tasks vital to productivity, a means of communication must exist. Programming languages provide this necessary link between man and machine. Because they are quite simple compared to human language, rarely containing more than few hundred distinct words, programming languages must contain very specific instructions. There are more than 2,000 different programming languages in existence, although most programs are written in one of several popular languages, like BASIC, COBOL, C++, or Java. Programming languages have different strengths and weaknesses. Depending on the kind of program being written, the computer will run on the experience of the programmer, and the way in which the program will be used, the suitability of one programming language over another will vary. One can categorize computer language as low Level and high level programming languages which are being discussed in the subsequent subsections.

1.5.1 Low-Level Programming Language

A low-level programming language is a language that provides little or no abstraction from a computer's instruction set architecture. The word "low" refers to the small or nonexistent amount of abstraction between the language and machine language; because of this, low-level languages are sometimes described as being "close to the

hardware." A low-level language does not need a compiler or interpreter to run; the processor for which the language was written is able to run the code without using either of these.

By comparison, a high-level programming language isolates the execution semantics of computer architecture from the specification of the program, making the process of developing a program simpler and more understandable.

Low-level programming languages are sometimes divided into two categories: first generation, and second generation programming languages.

First Generation Programming Language

The First-Generation Programming Language (1GL) is machine code or machine language. Machine language is a language which is directly understood by a computer. It is also called binary language as it is based on 0s or 1s. Any instruction in machine language is represented in terms of 0's and 1's, even the memory addresses are given in binary mode. Programs in machine language are very difficult to read and understand as binary codes of each command can not easily be remembered. So it is very difficult and complicated to write the computer program in machine language. The experienced programmer can only work in machine language that too after having the good knowledge of machine hardware. Programs written in machine language cannot easily be understood by other programmers. Currently, programmers almost never write programs directly in machine code, because as discussed above it not only require attention to numerous details which a high-level language would handle automatically, but it also requires memorizing or looking up numerical codes for every instruction that is used.

Second Generation Programming Language

The Second-Generation Programming Language (2GL) is assembly language. Assembly language was first developed in the 1950s and it is different for different microprocessors. It was the first step to improve the computer programming. For writing the programs in assembly language it is necessary that the programmers should have the knowledge of machine hardware. The assembly language eliminated much of the error-prone and time-consuming first-generation programming needed with the earliest computers, freeing the programmer from tedious or boring jobs such as remembering numeric codes and calculating memory addresses. The assembly language was once widely used for all sorts of programming. However, by the 1980s (1990s on small computers), the use of assembly languages had largely been supplanted by high-level languages, in the search for improved programming productivity.

Assembly languages are basically a family of low-level languages for programming computers, microprocessors, microcontrollers etc. They implement a symbolic representation of the numeric machine codes and other constants needed to program a particular CPU architecture. This representation is usually defined by the hardware manufacturer, and is based on abbreviations (called mnemonics) that help the

programmer remember individual instructions, registers, etc. An assembly language is thus specific to certain physical or virtual computer architecture. Instructions (statements) in assembly language are generally very simple, unlike those in high-level languages. Generally, an opcode is a symbolic name for a single executable machine language instruction, and there is at least one opcode mnemonic defined for each machine language instruction. Each instruction typically consists of an *operation* or *opcode* plus zero or more *operands*. Most instructions refer to a single value, or a pair of values. Operands can either be immediate (typically one byte values, coded in the instruction itself) or the addresses of data located elsewhere in storage.

A typical assembly language statement of 8080A or 8085 microprocessor written by the programmer is given below, which is divided in to four fields namely, Label, Mnemonics or Operation code (Opcode), Operand and comments.

Label	**Mnemonics**	**Operand**	**Commnets**
START:	LXI H,	2500 H	; Initialize H-L register pair

A label for an instruction is optional, but it is very essential for specifying jump locations. Similarly, comments are also optional but it is required for good documentation. The four fields of assembly language statements shown above are separated by the following delimiters:

Delimiters	**Placement**
Colon (:)	A colon is placed after the Label. Label is optional.
Space ( )	Space is left between an opcode and operand.
Comma (,)	A comma is placed between two operands.
Semicolon (;)	Semicolon is placed between the comments.

The program written in assembly language is converted to machine language manually. For writing the program in machine language, the starting address, where the program is to be stored should be known. Now the op code of the instruction is to be written in first location (starting address) and in the consecutive memory locations data /address of the operand is written. While storing the address in the memory locations, lower byte of he address is stored first then the upper byte.

A utility program called an assembler is used to translate assembly language statements into the target computer's machine code. The assembler performs a more or less isomorphic translation (a one-to-one mapping) from mnemonic statements into machine instructions and data. The reverse process that is conversion of machine language to the assembly language is done by deassembler.

For software development for a microprocessor/ microcomputer (written in assembly language in large number of instructions), it is absolutely essential to use an assembler. In fact assembler translates mnemonics into binary code with speed and accuracy; thus eliminating human error in looking for the opcodes. Other advantages of using the assembler for the software development are as follows:

- It assigns appropriate values to the symbols used in a program. This facilitates specifying jump locations.
- The assembler checks syntax errors, such as wrong labels and expressions, and provides error messages. However it cannot check logic errors in a program.
- It is easy to insert or delete instructions in a program; the assembler can reassemble the entire program quickly with new memory locations and modified addresses fro jump locations. This avoids rewriting the program manually.

1.5.2 High-Level Programming Language

The machine language and assembly languages discussed above are the first and second generation programming languages which fall in the category of low level languages. These languages require deep knowledge of computer hardware. The high level computer languages developed around 1960s are machine independent i.e. computer hardware is not necessary to know for the programmers. In high level languages one has to know only the instructions in English word and logic of the problem irrespective of the types of computer being used. For the computer programming prepared in high level language, only the use of English alphabets and mathematical systems like +, -, /, * etc. are made. In fact high level language is more close to user and is easy to read and understand. The high level languages are called procedural language and are designed to solve general and specific problems.

The term "high-level language" does not imply that the language is superior to low-level programming languages - in fact high level refers to the higher level of abstraction from machine language. They have no opcodes that can directly compile the language into machine code, unlike low-level assembly language.

In high level languages the words in English are converted into binary language of different microprocessors with the help of a program called **Interpreter** or **Complier**. The compiler or interpreter accepts English like statements as the input called Source code. The source codes are translated into machine language compatible with the microprocessor being used in the machine. The translation in the machine language from the source code is called the object code. Figure 1.4 shows the block diagram for translation of high level language program into machine code. Each microprocessor needs its own compiler or an interpreter for each high level language.

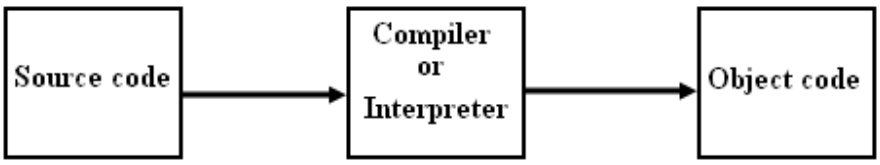


Fig. 1.4

The difference between a compiler and an interpreter is that the compiler reads the entire program first and then generates the object code; where as the interpreter reads one instruction at a time and produces its object code which is executed at the same time before reading the next instruction. The high level programming language developed so far may be categorized into third, fourth and fifth generation programming languages whose brief discussion is given below.

Third Generation Programming Language

A third-generation programming language (3GL) is a refinement of a second-generation programming language. Whereas a second generation language is more aimed to fix logical structure to the language, a third generation language aims to refine the usability of the language in such a way to make it more user friendly. A third generation language improves over a second generation language by having more refinement on the usability of the language itself from the perspective of the user.

Languages like ALGOL, COBOL, FORTRAN IV etc. are examples of this generation and were considered as high level languages. Most of these languages had compilers and the advantage of this was speed. Independence was another factor as these languages were machine independent and could run on different machines. FORTRAN (FORmula TRANslating) and COBOL (COmputer Business Oriented Language) were the first high-level programming languages to make an impact on the field of computer science. Along with assembly language, these two high-level languages have influenced the development of many modern programming languages, including Java, C++, and BASIC.

FORTRAN is well suited for math, science, and engineering programs because of its ability to perform numeric computations. The language was developed in New York by IBM's John Backus. FORTRAN was known as user’s friendly, because it was easy to learn in a short period of time and required no previous computer knowledge. It eliminated the need for engineers, scientists, and other users to rely on assembly programmers in order to communicate with computers. Although FORTRAN is often referred to as a language of the past, computer science students were still taught the language in the early 2000s for historical reasons, and because FORTRAN code still exists in some applications.

COBOL was another high level and third generation programming language well suited for creating business applications. COBOL's strength is in processing data, and in its simplicity.

Other languages like BASIC, C, C++, C#, Pascal, and Java are also third-generation languages.

Fourth-Generation Programming Language

A fourth-generation programming languages (4GL) (1970s-1990) are the programming language designed with a specific purpose in mind, such as the development of commercial business software. In the evolution of computing, the fourth generation language followed the third generation language in an upward trend toward higher abstraction and statement power. The fourth generation language was followed by efforts to define and use a fifth generation language (5GL). Basically the fourth generation languages are languages that consist of statements similar to statements in a human language. Fourth generation languages are commonly used in database programming and scripts. The commonly used fourth generation languages are FoxPro , SQL , MATLAB etc.

Fifth-Generation Programming Language

A fifth-generation programming language (5GL) is a programming language based around solving problems using constraints given to the program, rather than using an algorithm written by a programmer. Most constraint-based and logic programming languages and some declarative languages are fifth-generation languages.

While fourth-generation programming languages are designed to build specific programs, fifth-generation languages are designed to make the computer solve a given problem without the programmer. This way, the programmer only needs to worry about what problems need to be solved and what conditions need to be met, without worrying about how to implement a routine or algorithm to solve them. Fifth-generation languages are used mainly in artificial intelligence research. Prolog, OPS5, Visual Basic, and Mercury etc. are examples of fifth-generation languages.

Problems

1.1 Draw the block diagram of a general computer and discuss in detail the five blocks of a digital computer.
1.2 Discuss History and Evolution of Microprocessor.
1.3 What is microprocessor? What is the difference between a microprocessor and a microcomputer?
1.4 What is the difference between the 4-bit microprocessor and 8-bit microprocessor?
1.5 Name the main 8-bit microprocessors. Give the brief description of 8-bit microprocessor 8080A.

1.6 Discuss basic microprocessor system with the help of block diagram.
1.7 What are low level computer programming languages? Discuss them.
1.8 Discuss first generation computer programming languages.
1.9 Discuss second generation computer programming languages.
1.10 Write short note on high level computer programming languages.
1.11 What is the difference between assembly language and machine language?
1.12 Mention the brief description of Assembly Language. What are the advantages of assembler?

_____

2

SAP – I

For the beginners it is very difficult to understand the operation of a digital computer as it contains large details. To understand the step by step operation of a digital computer, the concept of Simple as Possible (SAP) computer has been introduced. Simple as possible computer, a conceptual computer will be discussed in three stages namely SAP – I, SAP – II and SAP – III computers. All the necessary details for the operation of digital computers will be discussed in three stages. After the study of these three stages of SAP computers, we will be in a position to understand clearly the fundamentals of microprocessor 8085 including the architecture, programming and interfacing devices. In this chapter the organization, programming and circuits of SAP – I computer will be discussed; the succeeding chapters will have the details of other stages of SAP computers.

2.1 ARCHITECTURE OF SAP - I

SAP – I, a conceptual computer is an 8-bit computer, as it can process the data of 8-bits. Further it is a simple computer and is considered for the basic understanding of the operation of digital computers, so it is assumed that it can store only 16 words (each word being 8 bit long). The length of the memory address register will of 4 bits since $2^4 = 16$ (16 is the total capacity of the memory unit). The address of 16 memory locations of memory address register (MAR) will be 0000 to 1111 i.e.

Memory locations (in Hexadecimal)	Address	Memory locations (in Hexadecimal)	Address
0 H	0000	8 H	1000
1 H	0001	9 H	1001
2 H	0010	A H	1010
3 H	0011	B H	1011
4 H	0100	C H	1100
5 H	0101	D H	1101
6 H	0110	E H	1110
7 H	0111	F H	1111

The basic architecture of this computer is shown in figure 2.1. It contains an 8-bit W-Bus (Wire Bus), which is used for data transfer to various 8-bit registers. A Bus is a group of conducting wires. In this figure, all register outputs connected to W-Bus are three-state which allows ordinary transfer of data. Remaining other register outputs continuously drive the boxes they are connected to.

A brief discussion of each block is given below:

Program Counter

First block of the SAP –I computers is the Program Counter (PC). It is basically the part of the control unit, its function is to send to the memory the address of the next instruction to be fetched and executed. Program counter is also called as the pointer as it is like someone pointing a finger at a list of instructions, saying do this first and do this next and so on.

In the beginning, program and data is stored in the memory through the input unit. A 4-bit binary address (0000 to 1111) is sufficient to address a word in the memory. The first instruction of the program is stored in the memory location 0000, second instruction at 0001 location, and the third instruction at 0010 location and so on. When the computer starts executing the program, the program counter is reset. The program counter is then incremented by one to get the next address of the memory location. After the first instruction is fetched and executed, the PC sends address 0001 to memory. Again the PC is incremented by one. After execution of second instruction, PC sends the next address to memory and so on.

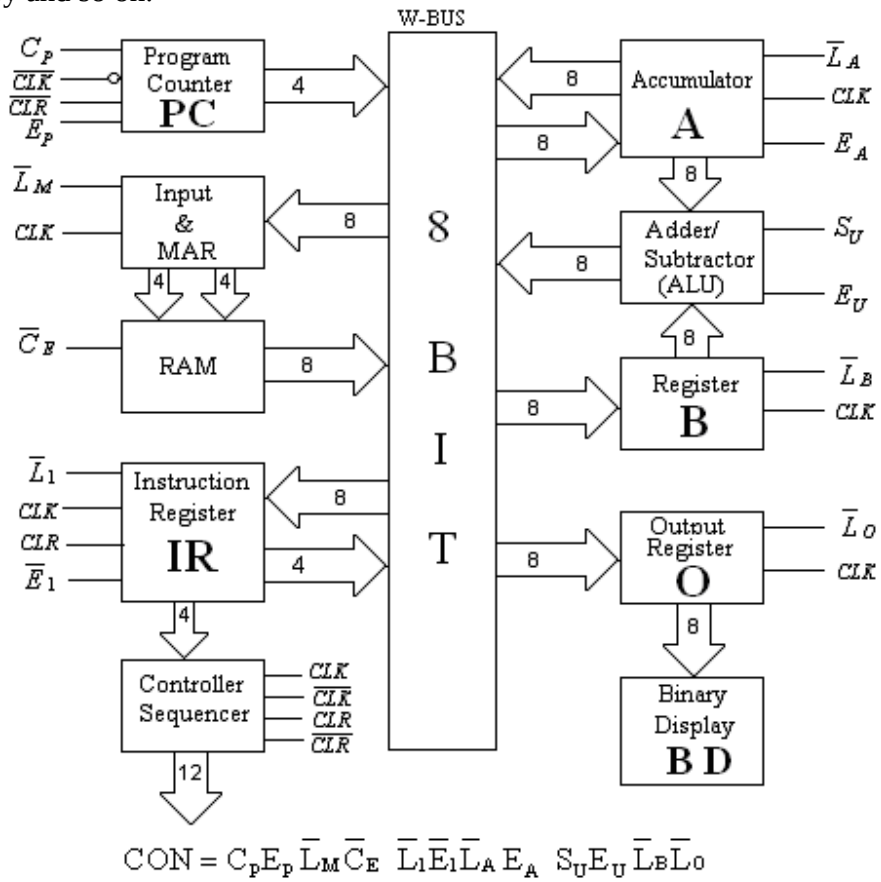


Fig. 2.1

Input and MAR

The second block of SAP –I computer is Input and Memory Address Register (MAR). It includes 4 bit address register and 8 bit data register. These registers are basically the parts of input unit. The input and MAR sends 4 bit address and 8 bit data to memory unit, this unit helps in storing the instructions and data to the memory, before the computer run starts. This unit includes a matrix of switches (micro-switches) for address

and data. In this SAP-I computer, the switch matrix allows to send 4 address bits and 8 data bits to memory. The relevant switch is opened or closed to program the memory. This can be done by providing switches on the front panel of the computer.

The memory address register is also the part of SAP-I memory. When SAP-I computer starts executing the program, the address in the PC is latched into the MAR. The MAR will then send this address to RAM for the read operation.

Memory Unit

It is 16x8 static TTL RAM (Random Access Memory) i.e. it is capable of storing 16 words (total capacity is only of 16 words) and each word is of 8 bit long. It stores the program and data before the execution of program; and during the execution it sends the stored instruction and data to the W-bus for further transfer to other registers as soon as address is received by it from MAR.

Instruction Register

The instruction register of SAP-I computer is the part of control unit. As already discussed during the execution of program, the content of addressed memory location is placed on the W-Bus. At the same time, during the next positive edge of the clock pulse this instruction (content of memory location) is loaded into the Instruction Register. The instruction register now splits the content of the instruction (8-bit) into two nibbles (one nibble is of 4-bit long). Instruction register sends the upper nibble directly to Controller/Sequencer, the lower nibble is, however, placed by the Instruction register to W-Bus for the read operation whenever needed.

Controller Sequencer

The block controller sequencer is basically control unit. This is a key unit for the automatic operation of this SAP-I computer. It generates a control word of 12 bits as given below:

$$CON = C_P E_P \overline{L}_M \overline{C}_E \ \overline{L}_1 \overline{E}_1 \overline{L}_A E_A \ S_U E_U \overline{L}_B \overline{L}_O$$

The symbolic notations of the control signal (CON) used in the computer are given in table 2.1.

Table 2.1

Notation	Interpretation
C_P	PC is incremented, when C_P is high.
E_P	Enable PC, when E_P is high.
$\overline{L}_M$	Loads MAR from W-Bus, when $\overline{L}_M$ is low.
$\overline{C}_E$	Loads RAM, when $\overline{C}_E$ is low.
$\overline{L}_1$	Loads instruction register, when it is low.
$\overline{E}_1$	Enable instruction register, when it is low.

$\overline{L}_A$	Loads the accumulator, when it is low.
E_A	Enable the accumulator, when it is high.
S_U	Enables subtraction, when if is high and enables addition when it is low.
E_U	Enable Adder/Subtrator to send the answer to W-Bus, when it is high.
$\overline{L}_B$	Loads the register B, when it is low.
$\overline{L}_O$	Loads the output register, when it is low.

Accumulator

The accumulator or register A is an 8-bit register and it is also called the buffer register. When $\overline{L}_A$ is low, the data from the W-Bus is loaded to the accumulator at the positive edge of the clock pulse. This data also directly goes to the Adder/Subtractor. The answer of Adder/Subtractor may also be loaded to the accumulator via W-Bus. The answer or the data stored in Accumulator will go to W-Bus when E_A is high.

Adder-Subtractor

The adder-Subtractor is the part of Arithmetic and Logical Unit (ALU) of the computer. This block can add the content of B-register to accumulator content when S_U is low. Similarly, this block can subtract the content of B-register from the accumulator content when S_U is high. The answer of addition or subtraction may be loaded to W-Bus when E_U is high. The subtraction in the block adder/subtrator is done using 2's complement method.

Register B

The register B is the part of Arithmetic and Logical Unit (ALU) of the computer. It is another buffer register of 8 bit long. The content to be added to the accumulator or subtracted from the accumulator, is loaded to B-register from W-Bus when $\overline{L}_B$ is low.

Output Register O

The output register **O** is the part of the output unit of the SAP-I computer. At the end of the execution of program the answer available at the accumulator may be transferred to output register **O**, at the positive edge of the clock pulse and when $\overline{L}_O$ is low, this register is also called the output port.

Binary Display (BD)

The last block of SAP-I computer is Binary Display (BD), which is the part of output unit. This display unit contains 8 LEDs (Light Emitting Diodes) connected to 8 bit output port through 8 flip-flops. When the data or answer is available at the output port, the same is transferred to LEDs indicating the answer is in binary form.

2.2 INSTRUCTION SET OF SAP-I COMPUTER

The instruction set is a set of instructions that can be executed by the computer. The computer programming is done using these instructions. Any problem to be solved on the computer is to be written in the form of a program. The SAP-I computer has got only five instructions as it is a very simple computer. These instructions are given in table 2.2. The instructions are written in abbreviated form or short form. These short forms of the instructions are known as Mnemonics (Memory aids). In fact the instructions in short form can easily be remembered by the programmers or users. Further, every instruction is stored in computer in coded form known as Op Code (Operation Code). The Op codes for the instructions of SAP-I computer given in table 2.2 are shown in binary form. From the above discussion it is clear that mnemonics is the short form of instruction for the user's remembrance and op code is the coded form of the instruction in machine language.

Table 2.2

Mnemonic	Op Code (Binary)	Operation
LDA	0000	It loads the accumulator the content from addressed location.
ADD	0001	It adds the content of the memory location to the accumulator content.
SUB	0010	It subtracts the content of the memory location from the accumulator content.
OUT	1110	It transfers the accumulator content to the output port.
HLT	1111	It stops the computer for further execution.

LDA Instruction

The LDA instruction loads the content from the specified memory location to the accumulator. There is always an operand with this instruction. The operand with the LDA instruction is the address of the memory location whose content is to be transferred to the accumulator.

For example

LDA 9 H

will load the accumulator, the content from the memory location 9 H. The alphabet H denotes hexadecimal number, as the op code for LDA is 0000 and binary equivalent of 9H is 1001. The machine language for this instruction is

0000 1001.

It must be remembered that LDA 9 H is known as the assembly language of the instruction and 0000 1001 (09H) is known as machine language of the instruction.

If the memory location 9H has the content 0001 0010 (12 H), then after execution of the instruction in SAP-I computer, it will load the content 0001 0010 to the accumulator, i.e. $A \leftarrow 00010010$ or $A \leftarrow 12H$

ADD Instruction

ADD instruction adds the content of memory location to the accumulator content. The address of the memory location is the operand of this instruction.

For example

ADD E H

adds the content of memory location E H (1110) to the accumulator content.

If before the execution of this instruction accumulator A is having data say 00010101, and 0000 0001 is already stored in memory location E H, then after execution of he instruction ADD E H the accumulator is loaded with the data 0001 0110,

as 0000 10101 + 0000 0001 = 0001 0110.

i.e. $A \leftarrow 00010110$ or $A \leftarrow 16H$

SUB Instruction

SUB instruction subtracts the content of the memory location from the accumulator content. The operand for this instruction is the address of the memory location.

For example

SUB C H

subtracts the content of memory location C H (1100) from the accumulator content.

If before the execution of this instruction accumulator A is having the content say 0000 0011 and the content in memory location C H is 0000 0010, then after the execution of this instruction, the accumulator A will have the answer as:

0000 0011 – 0000 0010 = 0000 0001

So $A \leftarrow 00000001$ or $A \leftarrow 01H$

OUT Instruction

The SAP-I instructions LDA, ADD and SUB discussed above are the memory reference instructions since the address of the memory location is the operand for these instructions. OUT instruction is not the memory reference instruction, as this instruction is itself complete and operand is not needed. The OUT instruction is used to transfer the accumulator content (answer after processing) to the output port.

HLT Instruction

HLT instruction is also not memory reference instruction, as it is complete itself and no operand is used with this instruction. HLT instruction stops further processing of the computer. This instruction must be used as the last instruction of every program otherwise meaningless answer will be obtained.

The complete assembly and machine code (Binary and Hexadecimal) of all the operations are shown in table 2.3.

Table 2.3

Assembly Code	Machine Code		Operations
	Binary	Hex	
LDA 5H	0000 0101	05 H	Loads the Acc., the content of memory location 5 H.
ADD CH	0001 1100	1C H	Adds the content of memory location CH to Acc. content.
SUB BH	0010 1011	2B H	Subtracts the content of memory location BH from the Acc. content.
OUT	1110 xxxx	Ex H	Sends Acc. content to the output port.
HLT	1111 xxxx	Fx H	Stops the processing.

2.3 PROGRAMMING OF SAP-I COMPUTER

Programming means set of instructions written by the user or programmer for performing a particular task. These instructions are then fed to the computer to get the

desired result. The program for the particular task is written by the programmer in assembly language (in mnemonic form) and then fed to the computer in the consecutive memory locations in the form of op codes (machine language) along with data. The program written in mnemonics form or assembly language is also called Source Program and the program written in machine language is known as Object Program. It will now be illustrated in more detail by taking an example.

Suppose we wish to get the addition of three numbers (decimal) 2, 3 and 5. Let these numbers are stored in three different memory locations 6, 7 and 8 respectively. Following steps will be taken for its execution:

Step-I	First number from the memory location 6 should be transferred to the accumulator (LDA 6 H).
Step-II	Second number from memory location 7 should be added to the accumulator and sum of these two numbers should remain in the accumulator (ADD 7 H).
Step-III	Third number from memory location 8 should now be added to the accumulator content and final sum should also remain in the accumulator (ADD 8 H).
Step-IV	Final answer (accumulator content) should be sent to the output port for the display in binary form (OUT).
Step-V	Stop the processing of the computer (HLT).

The program is now fed in the memory locations starting at 0 H and data in the memory locations starting at 6 H, as:

Memory location (in Hex)	**Mnemonic**
0 H	LDA 6 H
1 H	ADD 7 H
2 H	ADD 8 H
3 H	OUT
4 H	HLT
5 H	xx H
6 H	02 H
7 H	03 H
8 H	05 H

This program in machine language is given as:

Memory location	**Op code**
0000	0000 0110
0001	0001 0111
0010	0001 1000
0011	1110 xxxx
0100	1111 xxxx
0101	xxxx xxxx
0110	0000 0010

0111	0000 0011
1000	0000 0101

Note: x – sign represents any binary digit 0 or 1.

Example 2.1 *Write an assembly language program that performs the following operation in SAP-I computer.*

$$7 + 6 - 4 + 5 - 3$$

Use memory locations 7H to BH for the data. Further convert this program in machine language.

Solution. SAP-I assembly language program of the given problem is as given below:

Memory location (in Hex)	**Mnemonic**
0 H	LDA 7 H
1 H	ADD 8 H
2 H	SUB 9 H
3 H	ADD A H
4 H	SUB B H
5 H	OUT
6 H	HLT
7 H	07 H
8 H	06 H
9 H	04 H
A H	05 H
B H	03 H

This program in machine language is given by:

Memory location	**Op code**
0000	0000 0111
0001	0001 1000
0010	0010 1001
0011	0001 1010
0100	0010 1011
0101	1110 xxxx
0110	1111 xxxx
0111	0000 0111
1000	0000 0110
1001	0000 0100
1010	0000 0101
1011	0000 0011

Example 2.2 *Write an assembly language program to calculate the following expression on SAP-I computer:*

$$Y + 2Z - 3W$$

The data Y, Z and W (as 6, 7 and 2) are stored in memory locations 8H to AH.

Solution.

SAP-I assembly language program of the given problem is as given below:

Memory location (in Hex)	Mnemonic
0 H	LDA 8 H
1 H	ADD 9 H
2 H	ADD 9 H
3 H	SUB A H
4 H	SUB A H
5 H	SUB A H
6 H	OUT
7 H	HLT
8 H	06 H
9 H	05 H
A H	02 H

Example 2.3 *Write assembly program for SAP-I computer to solve the following problem:*

$$17 + 22 + 20 - 33.$$

The numbers are given in decimal. Also give the machine language program.

Solution.

First the decimal numbers are converted to Hexadecimal numbers. The Hexadecimal equivalents of these numbers are given as:

17 = 11 H
22 = 16 H
20 = 14 H
33 = 21 H

Let us assume that these numbers are stored in memory locations 7H to AH.

Memory location (in Hex)	Mnemonic
0 H	LDA 7 H
1 H	ADD 8 H
2 H	ADD 9 H
3 H	SUB A H
4 H	OUT
5 H	HLT
6 H	x x H
7 H	11 H
8 H	16 H
9 H	14 H
A H	21 H

The machine language program is given as follows:

Memory location	Op code
0000	0000 0111
0001	0001 1000
0010	0001 1001
0011	0010 1010
0100	1110 xxxx

0101	1111 xxxx
0110	xxxx xxxx
0111	0001 0001
1000	0001 0110
1001	0001 0100
1010	0010 0001

2.4 WORKING OF SAP-I COMPUTER

For the working of SAP-I computer, instructions are fetched from RAM and then executed one by one in a sequence till a Halt instruction is executed. The SAP-I computer executes program starting from 0 H memory location. For this controller / sequencer unit generates a CLK signal for all the register and CLR signal for the Program counter. As soon as the computer starts executing the program, program counter receives a CLR signal which resets the program counter. The program counter will send the address of 0 H memory location.

In SAP-I computer the loading of a register takes place only when setup and hold times of the clock pulse are satisfied. For this 50% duty cycle clock pulse is used and positive edge occurs half way through each state. Waiting half a cycle before loading the register satisfies setup time; waiting half a cycle after loading satisfies the hold time. Secondly, the reason for waiting half a cycle before loading a register is that when enable input of the sending register goes active, the content of the register suddenly dumped on to register. Stray capacitance and lead inductance may, however, prevent the voltage level immediately. In other words the transients on the W-Bus will be eliminated if the clock has the wait time to die out the transients and valid data is transferred. The control word generated by the controller sequencer unit does the automatic operation of the computer. The execution of a program is carried out by fetching and execution operations of the instructions.

The instruction cycle for the execution of an instruction consists of the following two cycles:

(i) Fetch cycle — fetches a word from memory
(ii) Execution cycle — executes the fetched instruction

We will now discuss these cycles in detail.

2.4.1 Fetch Cycle

The operation of fetch cycle is performed in three states namely:

(a) Address State
(b) Increment State
(c) Memory State

During the address state, the content of PC is transferred to MAR via W-Bus:

$$MAR \leftarrow PC$$

During the increment state, the PC gets incremented. The incremented content will be sent to MAR when next fetch cycle occurs.

$$PC \leftarrow PC+1$$

During the memory state, memory read operation is performed.

$$W-Bus \leftarrow M_{PC}$$

The control word generated by the controller sequencer for these three states is given below:

	C_P	E_P	$\overline{L}_M$	$\overline{C}_E$	$\overline{L}_1$	$\overline{E}_1$	$\overline{L}_A$	E_A	S_U	E_U	$\overline{L}_B$	$\overline{L}_O$	
T_1	0	1	0	1	1	1	1	0	0	0	1	1	**Address State** Only E_P and $\overline{L}_M$ are active. Figure 2.2 shows shaded boxes as active.
T_2	1	0	1	1	1	1	1	0	0	0	1	1	**Increment State** Only C_P is active. Figure 2.3 shows shaded boxes as active.
T_3	0	0	1	0	0	1	1	0	0	0	1	1	**Memory State** Only C_E and $\overline{L}_1$ are active. Figure 2.4 shows shaded boxes as active.

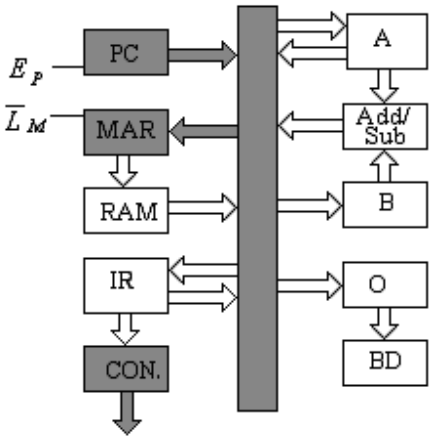


Fig. 2.2

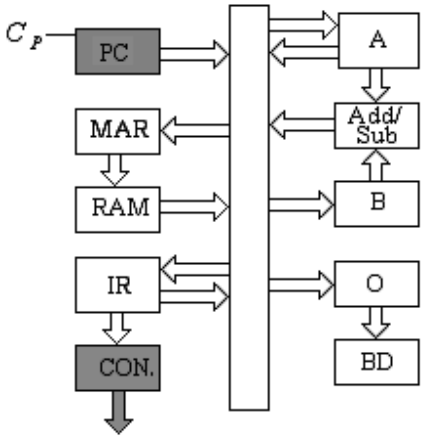


Fig. 2.3

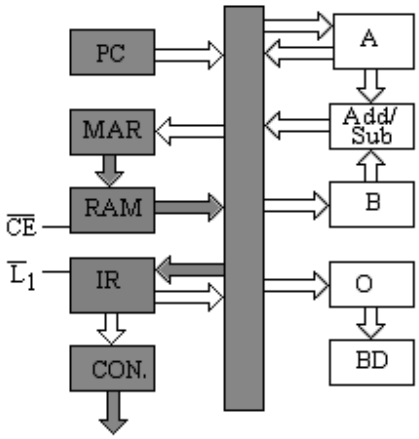


Fig. 2.4

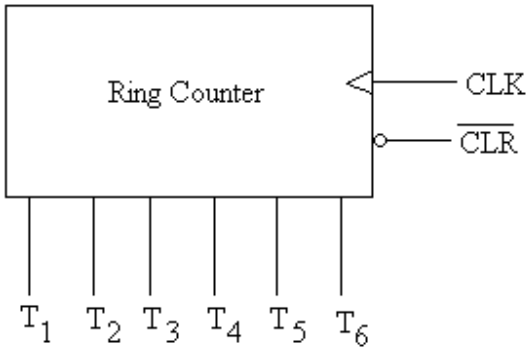


Fig. 2.5(a)

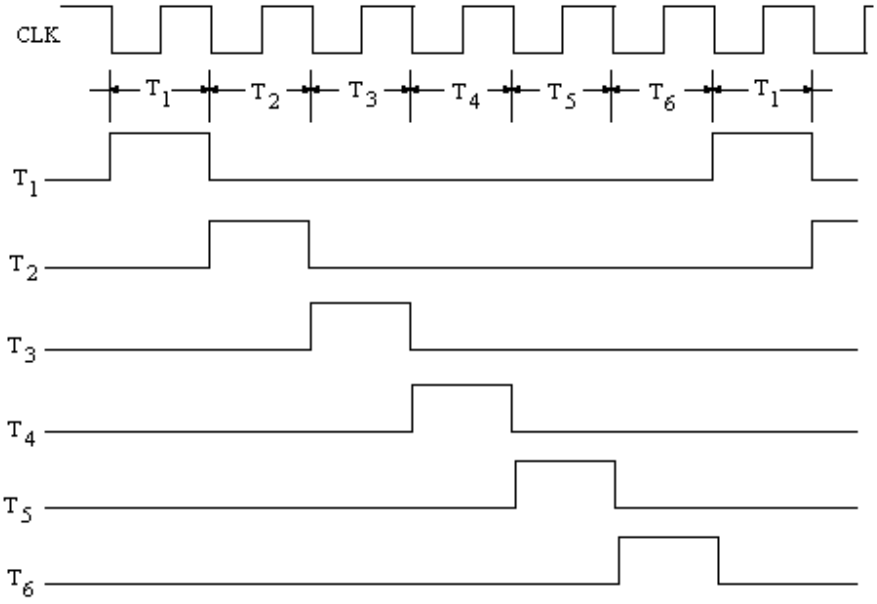


Fig. 2.5(b)

The Controller-sequencer keeps a track of three different states of Fetch cycle and generates control signals accordingly. A ring counter is used for this purpose.

Consider a 6-bit synchronous ring counter as shown in figure 2.5(a). The output T of the ring counter is given by:

$$T = T_6\ T_5\ T_4\ T_3\ T_2\ T_1$$

At the start of the computer run, 6-bit output of the ring counter at the successive clock pulse is shown in figure 2.5(b).

The details of the same are given below:

	T_6	T_5	T_4	T_3	T_2	T_1
1^{st} clock pulse =	0	0	0	0	0	1
2^{nd} clock pulse =	0	0	0	0	1	0
3^{rd} clock pulse =	0	0	0	1	0	0
4^{th} clock pulse =	0	0	1	0	0	0

5^{th} clock pulse = 0 1 0 0 0 0
6^{th} clock pulse = 1 0 0 0 0 0
7^{th} clock pulse = 0 0 0 0 0 1
and so on.

The different outputs are also called T-states or timing states. During first three clock pulses the operation of fetch cycle occurs which are explained below:

During T_1 state of the timing signal, Controller-sequencer generates a signal so that E_P is high and $\overline{L}_M$ is low which sends the content of PC (address of the instruction stored in RAM) to W-Bus and at the positive edge of the clock pulse (midway of the clock pulse) the content of the W-Bus are latched into MAR due to low $\overline{L}_M$ – **This state is called Address State of the fetch cycle.**

During T_2 state of the timing signal, controller sequencer generates a signal giving high to C_P. At the positive edge of the clock pulse (midway of the clock pulse), PC advances the count by 1 – **This state is called Increment State of the fetch cycle.**

During T_3 stat, control signal generated by Controller-sequencer makes $\overline{C}_E$ and $\overline{L}_I$ low. The addressed RAM word is transferred to W-Bus and at positive edge of CLK (midway of clock) the data is transferred to W-Bus and then loaded to Instruction register – **This state is called Memory State of the fetch cycle.**

The next three states of clock pulse are called the execution cycle of SAP-I instructions.

2.4.2 Execution Cycle

After the instruction is fetched and transferred to the Instruction Register (IR) during the fetch cycle, the IR splits the instruction word (8-bit) into two nibbles. The upper nibble goes directly to the controller-sequencer, where it is decoded and accordingly the control word is generated to act as per the direction of the instruction. The decoded output will be different for different instructions (LDA, ADD, SUB, OUT, and HLT). So during the execution cycle (T_4 to T_6 of the clock pulse), the operation of control signal will be different for different instructions. Now we will discuss these operations for each cycle.

Execution Cycle of LDA Instruction:

For the discussion of the operation LDA instruction, let us assume the instruction is LDA 6 H = 0 0 0 0 0 1 1 0

In this instruction 0 0 0 0 is the op code of LDA and 6 H (0 1 1 0) is the address of the location where the data is stored.

During T_3 state of CLK, 0000 0110 is loaded to Instruction Register (IR). The upper nibble 0000 directly goes to controller-sequencer, where it is decoded. During T_4, T_5 and T_6 states for the execution of LDA instruction, the controller-sequencer will generate the following control words in sequence:

	C_P	E_P	$\overline{L}_M$	$\overline{C}_E$	$\overline{L}_I$	$\overline{E}_I$	$\overline{L}_A$	E_A	S_U	E_U	$\overline{L}_B$	$\overline{L}_O$	
T_4	0	0	0	1	1	0	1	0	0	0	1	1	$MAR \leftarrow IR_{LNIBBLE}$
T_5	0	0	1	0	1	1	0	0	0	0	1	1	$A \leftarrow M(MAR)$
T_6	0	0	1	1	1	1	1	0	0	0	1	1	NO OPERATION

During T_4 state of CLK, lower nibble already available at the W-Bus is loaded into MAR since $\overline{E}_I$ and $\overline{L}_M$ are active.

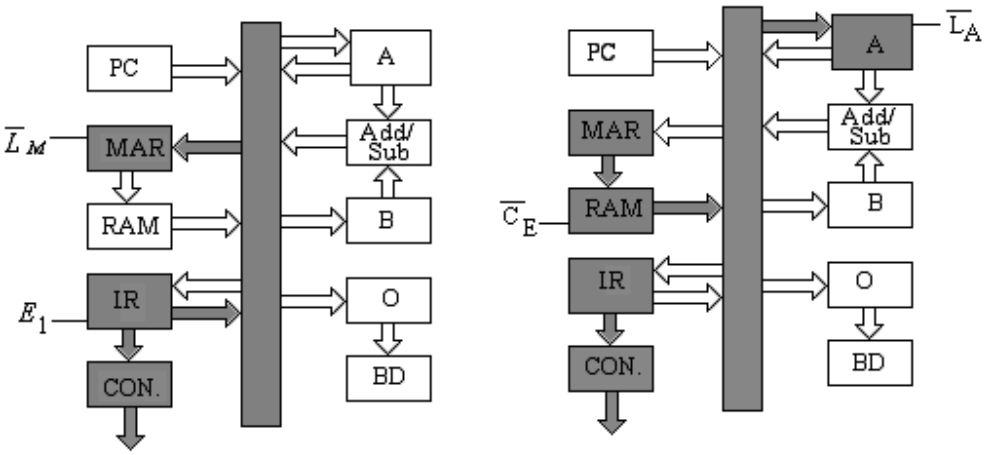
During T_5 state of CLK, $\overline{C}_E$ and $\overline{L}_A$ are active which loads the addressed content from RAM to Accumulator. This was the function of LDA instruction.

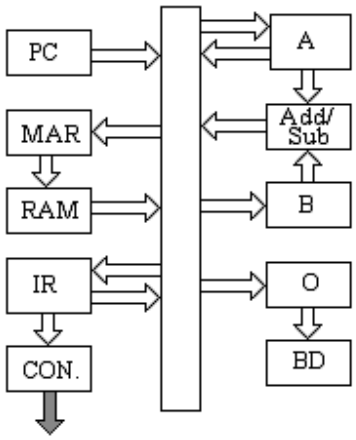
During T_6 state of CLK, no operation is performed as nothing was left for this instruction to do.

T_4 to T_6 states of LDA instruction are shown in figure 2.6 (a to c) with active parts as shaded boxes and figure 2.7 shows timing diagram for Fetch and execution cycle of LDA instruction.

The control word generated for each T-state is called as Microinstruction. One of the instructions in instruction set is known as Macroinstruction. The microinstruction for T_4, T_5 and T_6 states of LDA instruction is given below:

States	Microinstruction	(12 Bit Control Word)
T_4	1A3 H	0001 1010 0011
T_5	2C3 H	0010 1100 0011
T_6	3E3 H	0011 1110 0011





(c)

Fig. 2.6

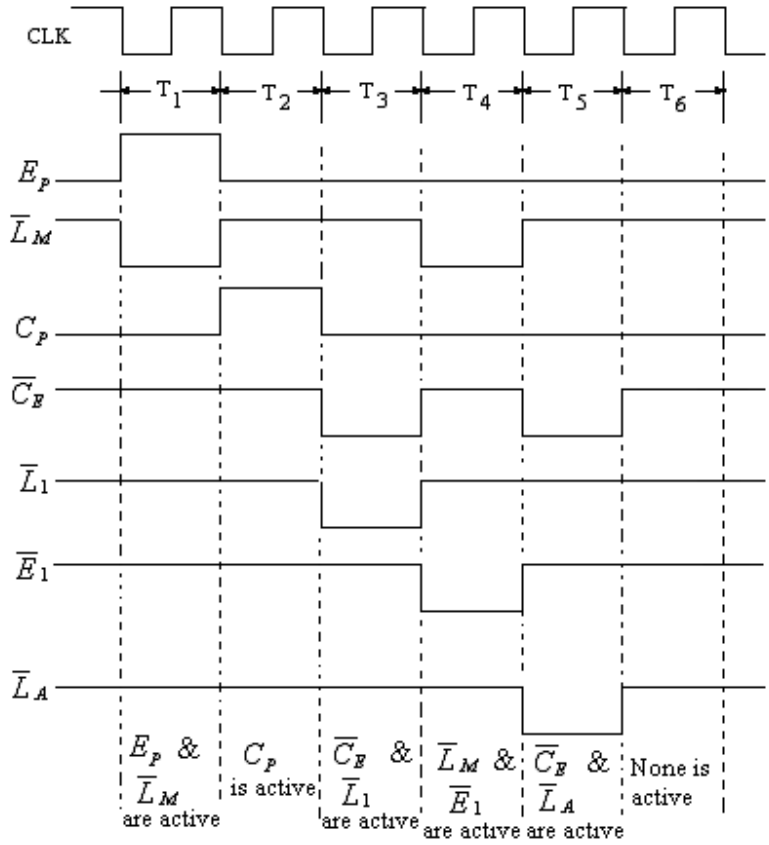


Fig. 2.7

Execution Cycle of ADD Instruction:

For the execution of ADD instruction following control signals in sequence during T_4, T_5 and T_6 states will be observed:

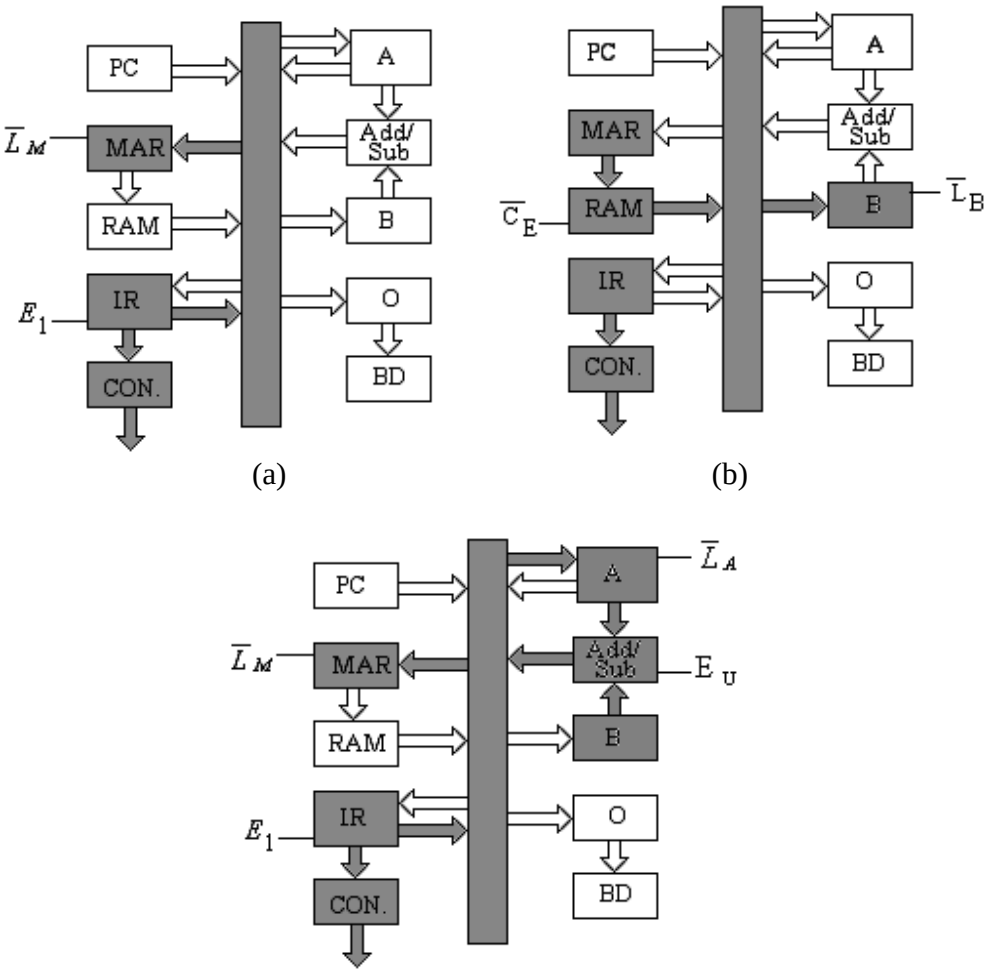
	C_P	E_P	$\overline{L}_M$	$\overline{C}_E$	$\overline{L}_I$	$\overline{E}_I$	$\overline{L}_A$	E_A	S_U	E_U	$\overline{L}_B$	$\overline{L}_O$	
T_4	0	0	0	1	1	0	1	0	0	0	1	1	$MAR \leftarrow IR_{LNIBBLE}$
T_5	0	0	1	0	1	1	1	0	0	0	0	1	$B \leftarrow M(MAR)$
T_6	0	0	1	1	1	1	0	0	0	1	1	1	$A \leftarrow SUM$

During T_4 state, the address part of ADD instruction moves to MAR as $\overline{L}_M$ and $\overline{E}_I$ are active. The data from MAR is fetched and loaded into register B during T_5 state as $\overline{C}_E$ and $\overline{L}_B$ are active. The adder-subtractor adds the content of Accumulator and register B as S_U is inactive. During T_6 state the answer (Sum) is loaded to accumulator, as E_U and $\overline{L}_A$ are active.

T states of ADD instruction are shown in figures 2.8 (a to c) with active parts as shaded boxes. The timing diagram of Fetch and ADD instruction is shown in figure 2.9.

The microinstruction for T_4, T_5 and T_6 states of ADD instruction is given below:

States	Microinstruction	(12 Bit Control Word)
T_4	1A3 H	0001 1010 0011
T_5	2E1 H	0010 1110 0001
T_6	3C7 H	0011 1100 0111



(a) (b)

(c)

Fig. 2.8

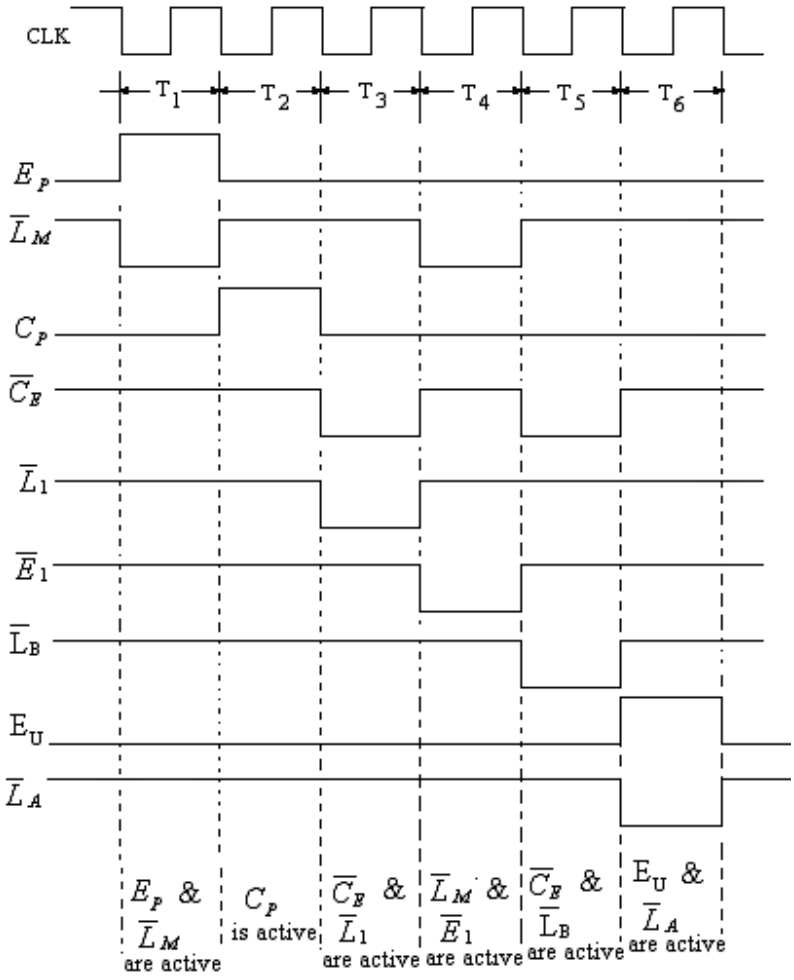


Fig. 2.9

Execution Cycle of SUB Instruction:

The third instruction of SAP-I instruction set is SUB-instruction. For the execution of SUB instruction, following control signals in sequence will be observed during T_4, T_5 and T_6 states.

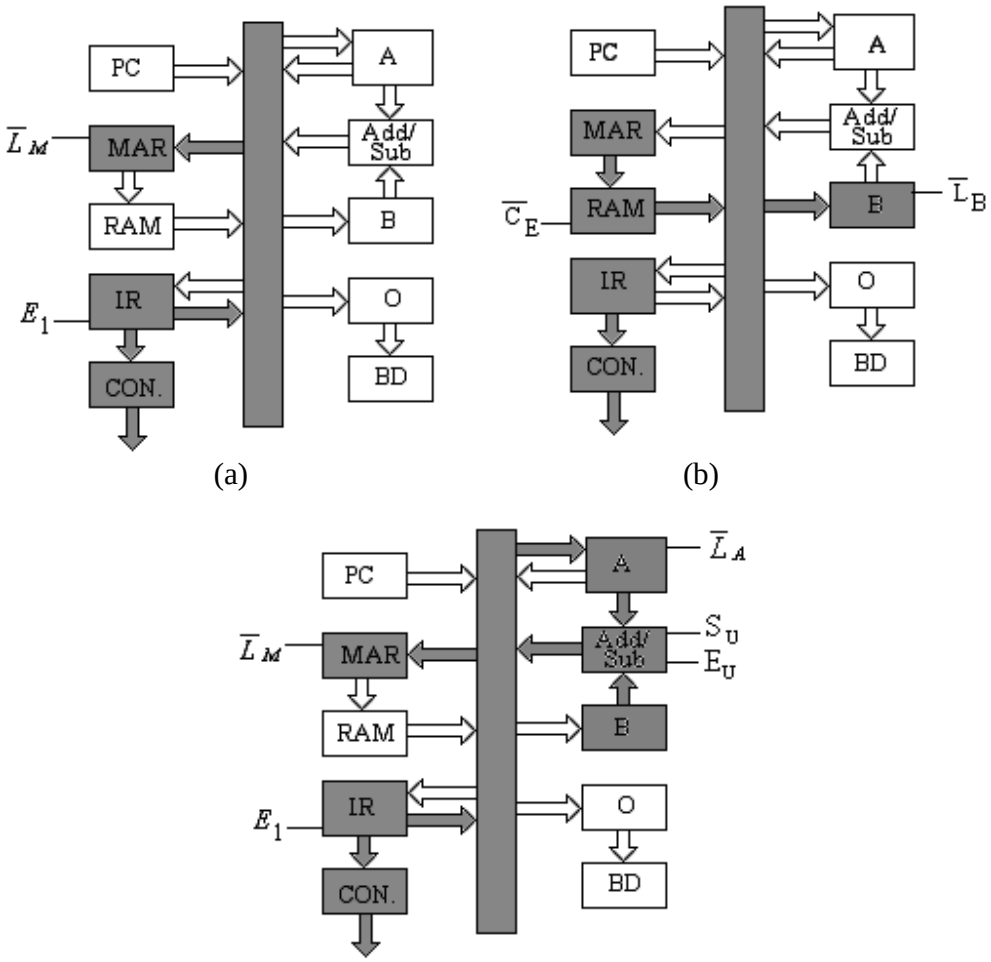
	C_P	E_P	$\overline{L}_M$	$\overline{C}_E$	$\overline{L}_1$	$\overline{E}_1$	$\overline{L}_A$	E_A	S_U	E_U	$\overline{L}_B$	$\overline{L}_O$	
T_4	0	0	0	1	1	0	1	0	0	0	1	1	$MAR \leftarrow IR_{LNIBBLE}$
T_5	0	0	1	0	1	1	1	0	0	0	0	1	$B \leftarrow M(MAR)$
T_6	0	0	1	1	1	1	0	0	1	1	1	1	$A \leftarrow DIFF.$

From the above sequence it is clear that SUB instruction follow the same sequence as ADD instruction except that during T_5 state S_U is high.

T states (T_4 to T_6) of SUB instruction are shown in figures 2.10 (a to c) with active parts as shaded boxes. The timing diagram of Fetch and SUB instruction is shown in figure 2.11.

The microinstruction for T_4, T_5 and T_6 states of SUB instruction is given below:

States	Microinstruction	(12 Bit Control Word)
T_4	1A3 H	0001 1010 0011
T_5	2E1 H	0010 1110 0001
T_6	3CF H	0011 1100 1111



(a) (b)

(c)

Fig. 2.10

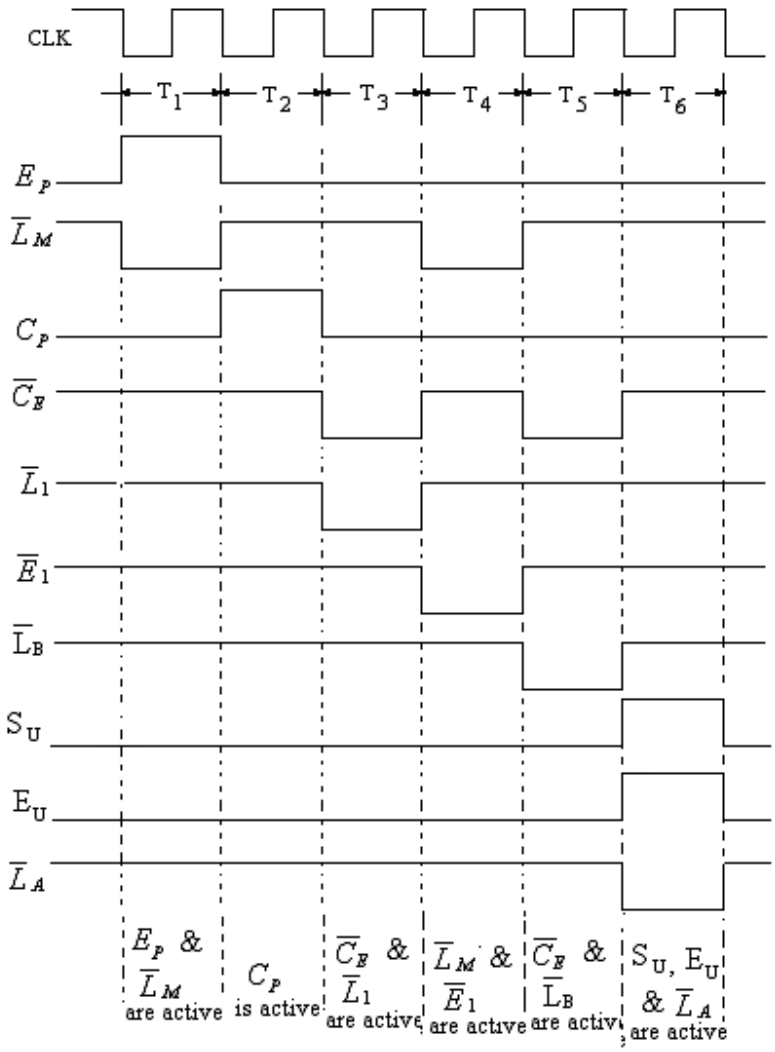


Fig. 2.11

Execution Cycle of OUT Instruction:

For the execution of this instruction, following control signals in sequence will be observed during T_4, T_5 and T_6 states of execution cycle.

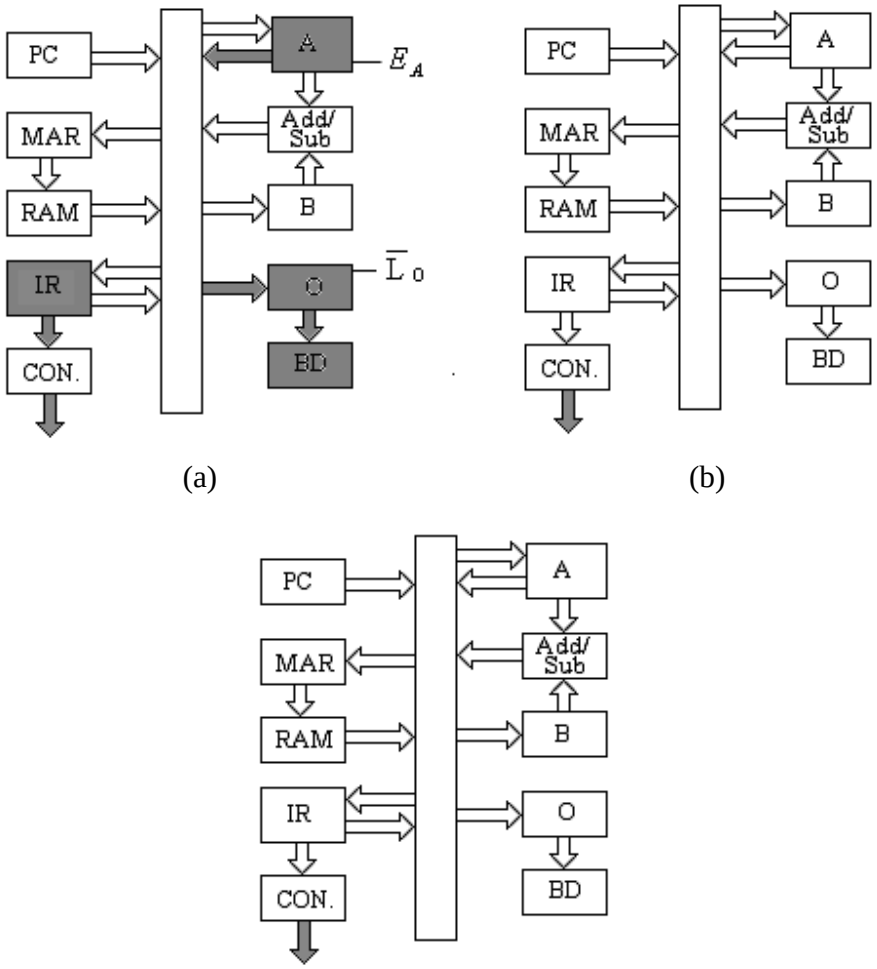
	C_P	E_P	$\overline{L}_M$	$\overline{C}_E$	$\overline{L}_1$	$\overline{E}_1$	$\overline{L}_A$	E_A	S_U	E_U	$\overline{L}_B$	$\overline{L}_O$
T_4	0	0	1	1	1	1	1	1	0	0	1	0
T_5	0	0	1	1	1	1	1	0	0	0	1	1
T_6	0	0	1	1	1	1	1	0	0	0	1	1

During T_4 state, E_A and $\overline{L}_O$ are active due to which at the positive edge of clock pulse accumulator content is loaded to the output register. High E_A send the accumulator data to W-Bus and low $\overline{L}_O$ loads the data from W-Bus to output register. The work of out instruction is complete. So no-operation is performed during T_5 and T_6 state i.e. all the signals of the control word are inactive.

T states of OUT instruction are shown in figures 2.12 (a to c) with active parts as shaded boxes. The timing diagram of Fetch and OUT instruction is shown in figure 2.13.

The microinstruction for T_4, T_5 and T_6 states of OUT instruction is given below:

States	Microinstruction	(12 Bit Control)
T_4	3F2 H	0011 1111 0010
T_5	3E3 H	0011 1110 0011
T_6	3E3 H	0011 1110 0011



(a) (b)

(c)

Fig. 2.12

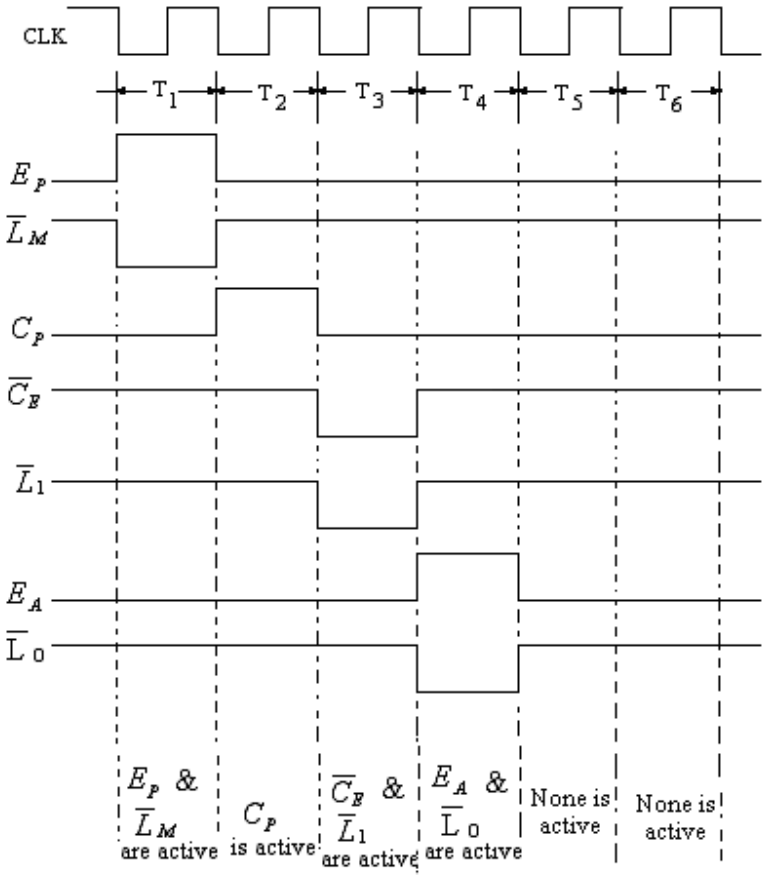


Fig. 2.13

Execution Cycle of HLT Instruction:

HLT instruction does not require any T-cycle for its execution. Thus during T_4, T_5 and T_6 state all the signals of the control word are inactive i.e. no-operation is performed during these states.

	C_P	E_P	$\overline{L}_M$	$\overline{C}_E$	$\overline{L}_I$	$\overline{E}_I$	$\overline{L}_A$	E_A	S_U	E_U	$\overline{L}_B$	$\overline{L}_O$
T_4	0	0	1	1	1	1	1	0	0	0	1	1
T_5	0	0	1	1	1	1	1	0	0	0	1	1
T_6	0	0	1	1	1	1	1	0	0	0	1	1

The microinstruction for T_4, T_5 and T_6 states of HLT instruction is given below:

States	**Microinstruction**	**(12 Bit Control Word)**
T_4	3E3 H	0011 1110 0011
T_5	3E3 H	0011 1110 0011
T_6	3E3 H	0011 1110 0011

Example 2.4 *What are Microinstructions for:*
(i) LDA instruction
(ii) ADD instruction

Solution. (i) During T_4 state of LDA instruction $\overline{L}_M$ and $\overline{E}_I$ are active.
During T_5 state of LDA instruction $\overline{C}_E$ and $\overline{L}_A$ are active
and During T_6 state of LDA instruction None is active.
So Microinstructions for LDA are

For T_4 1A3 H
T_5 2C3 H
T_6 3E3 H

(ii) Similarly, Microinstruction for ADD instruction are

For T_4 1A3 H
T_5 2C3 H
T_6 3E7 H

2.5 HARDWARE DESIGN OF SAP-I COMPUTER

In this section we shall discuss the hardware details of the following blocks of SAP-I computer:

Program Counter
Memory Unit
Instruction Register
Controller-Sequencer
Accumulator
Adder-Sutractor
B-Register
Output Register and Binary Display

2.5.1 Design of Program Counter

The program counter (PC) is designed using four J-K flip-flops. The circuit diagram of program counter is shown in figure 2.14. From this diagram it can well be understood that during the start of computer run a low $\overline{CLR}$ signal resets the program counter to 0 0 0 0. During T_1 state of fetch cycle, a high E_P sends the address to the W-Bus. During T_2 state, high C_P increments the Program counter, at the midway through T_2. the program counter is, however, inactive during T_3 through T_6 states.

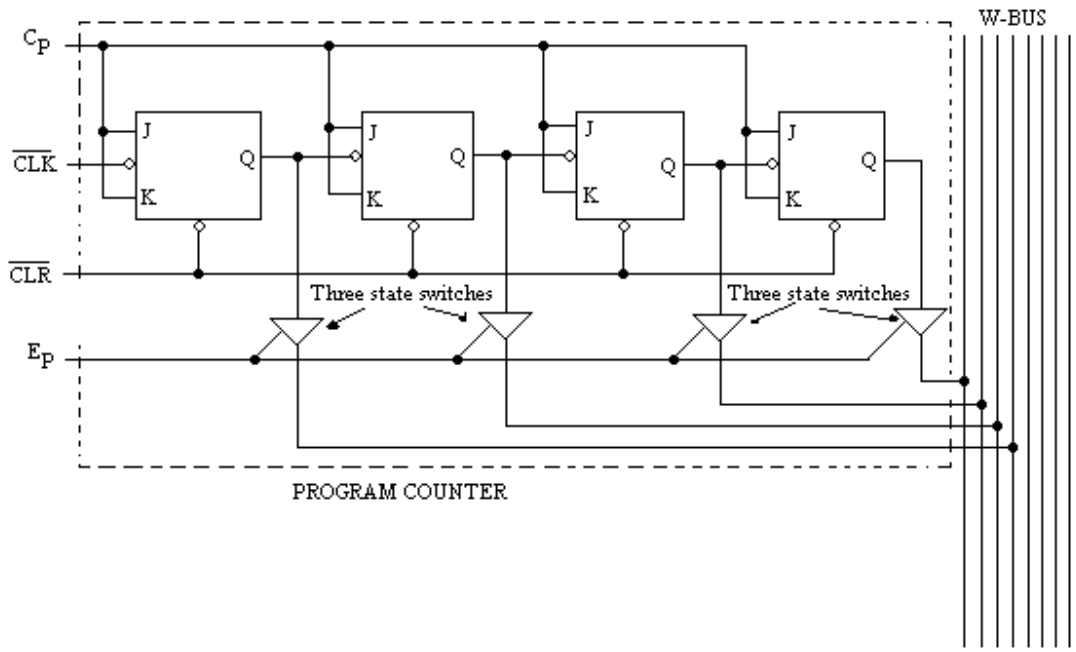


Fig. 2.14

2.5.2 Memory Unit

Figure 2.15 shows the circuit diagram of memory unit, which consists of Input & MAR and RAM. In this figure IC1 and IC2 form Input & MAR; IC3 and IC4 form read/write memory. IC1 is a 4-bit buffer register. It is a three state output register which is configured here in a two state output, as it is not to be connected to W-Bus. IC2 is basically 2 to 1 nibble multiplexer.

When the switch S_1 is connected to Run position, the address bits are directly available at the output of IC2. If on the contrary S_1 is at the Run position, the address bit from the W-Bus will be available at the output of this IC2.

IC3 and IC4 are two 16X4 static RAM, which form the RAM of SAP-I. These two IC's are configured to form, 16X8 read/write memory. When switch S_1 is at PROG position, S_2 is at write position, the data from D_0 to D_7 will be written into static RAM. If S_1 is at Run position, the data already stored in the memory will be available at the W-Bus.

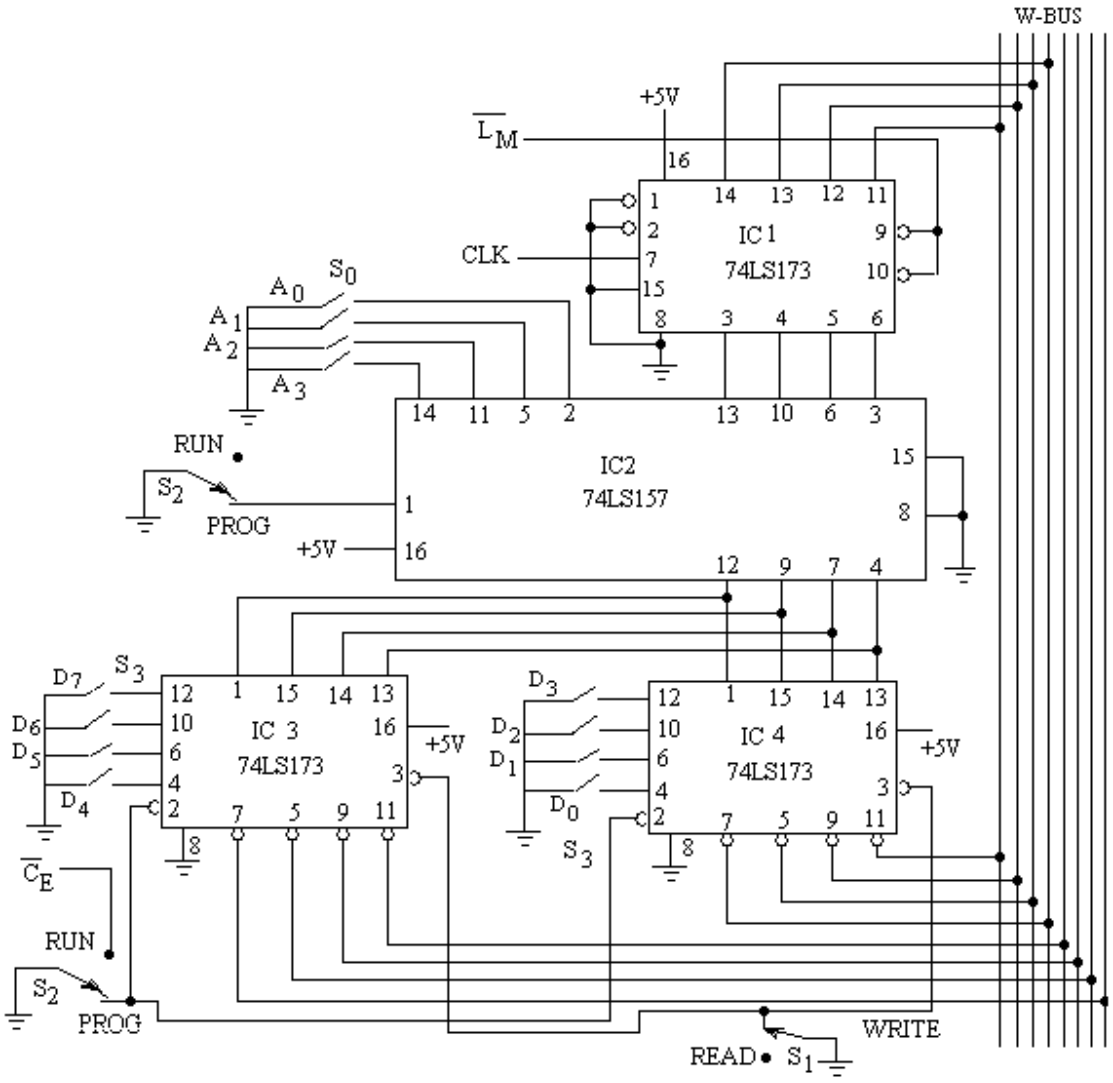


Fig. 2.15

2.5.3 Instruction Register

IC1 and IC2 forms the Instruction Register. These are two four-bit buffer registers. These buffer registers are three state registers. IC1 is configured as two state output. The outputs of this IC are upper nibble I_7 I_6 I_5 I_4 which directly goes to controller-sequencer where these are decoded. The outputs of IC2 are the lower nibble (operand of the fetched instruction), which are directly connected to W-Bus. The logic circuit diagram of the Instruction register is shown in figure 2.16.

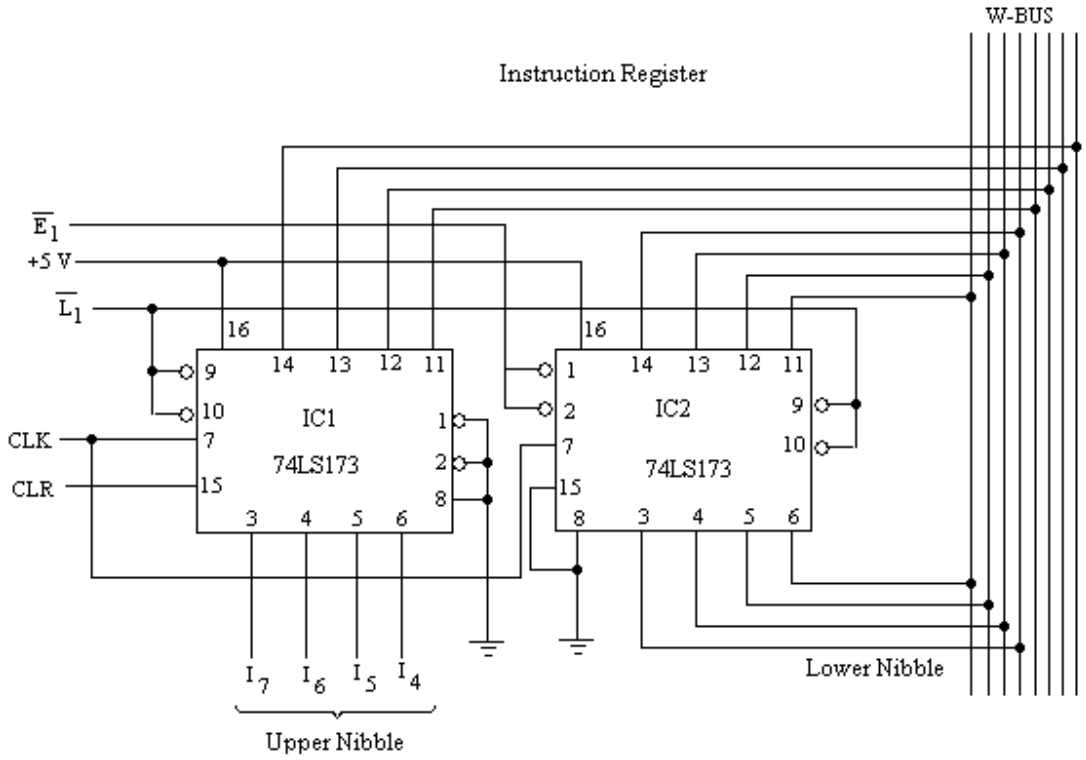


Fig. 2.16

2.5.4 Controller-Sequencer

The schematic block diagram of controller-sequencer is shown in figure 2.17. It consists of a ring counter, instruction decoder and control matrix for the generation of 12-bit control word. The instruction decoder is a simple decoder, which provides the outputs LDA, ADD, Sub, OUT and HLT. In fact it is a 4-to-5 decoder.

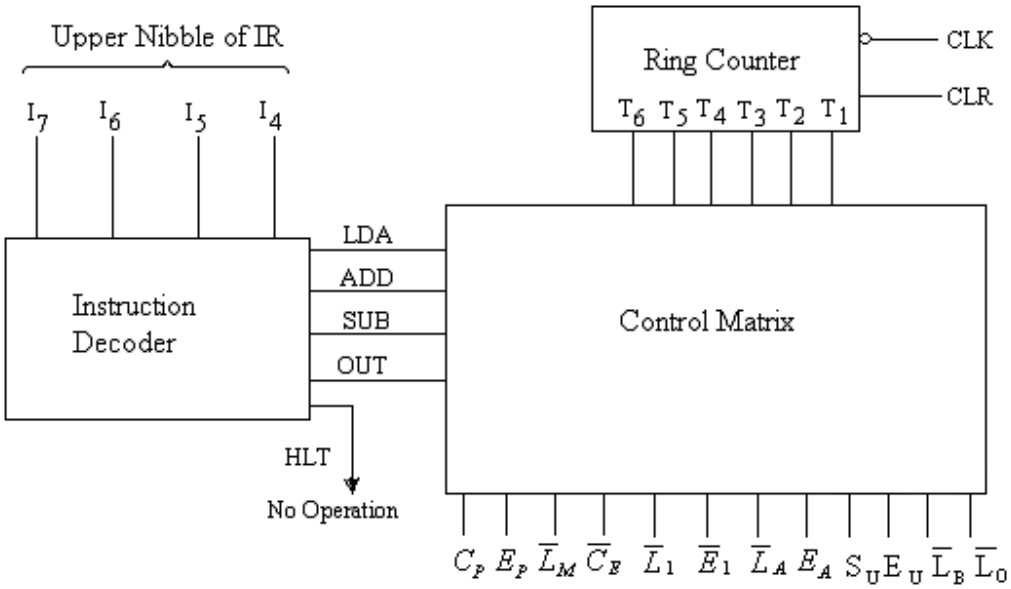


Fig. 2.17

The required logic is shown in table 2.4.

Table 2.4

I_7	I_6	I_5	I_4	OUTPUT
0	0	0	0	LDA
0	0	0	1	ADD
0	0	1	0	SUB
1	1	1	0	OUT
1	1	1	1	HLT

The logic diagram for the Instruction decoder of SAP-I computer is shown in figure 2.18, which is as per the table 2.4.

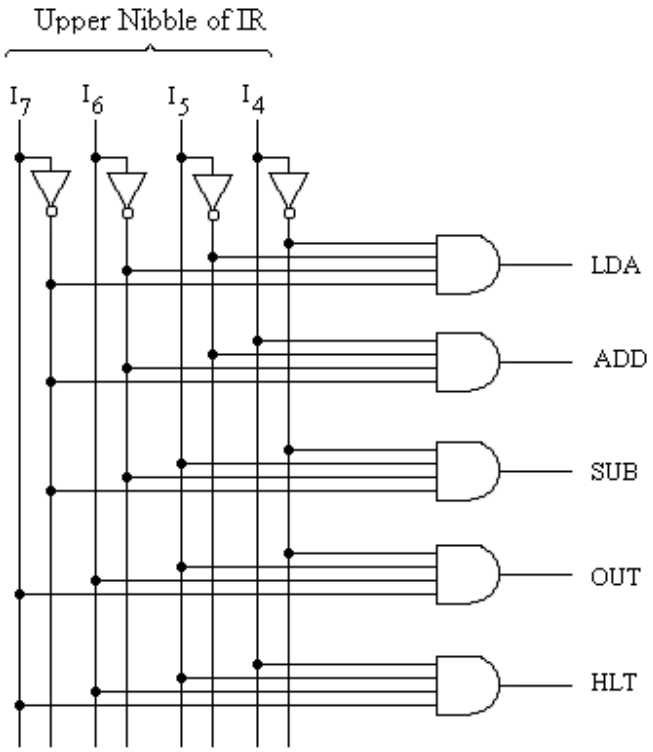


Fig. 2.18

The synchronous counter has already been discussed. This ring counter gives 6-bit output (T_1 to T_6) at the successive clock pulse.

For the generation of 12-bit control signal, the twelve outputs may be obtained from the logic expressions given below:

$C_P = T_2$

$E_P = T_1$

$\overline{L}_M = T_1 + T_4.LDA + T_4.ADD + T_4.SUB$

$\overline{C}_E = T_3 + T_5.LDA + T_5.ADD + T_5.SUB$

$\overline{L}_I = T_3$

$\overline{E}_I = T_4.LDA + T_4.ADD + T_4.SUB$

$\overline{L}_A = T_5.LDA + T_6.ADD + T_6.SUB$

$E_A = T_4.OUT$

$S_U = T_6.SUB$

$E_U = T_6.ADD + T_6.SUB$

$\overline{L}_B = T_5.ADD + T_5.SUB$

$\overline{L}_O = T_4.OUT$

The logic circuit diagram of the above expressions is shown in figure 2.19.

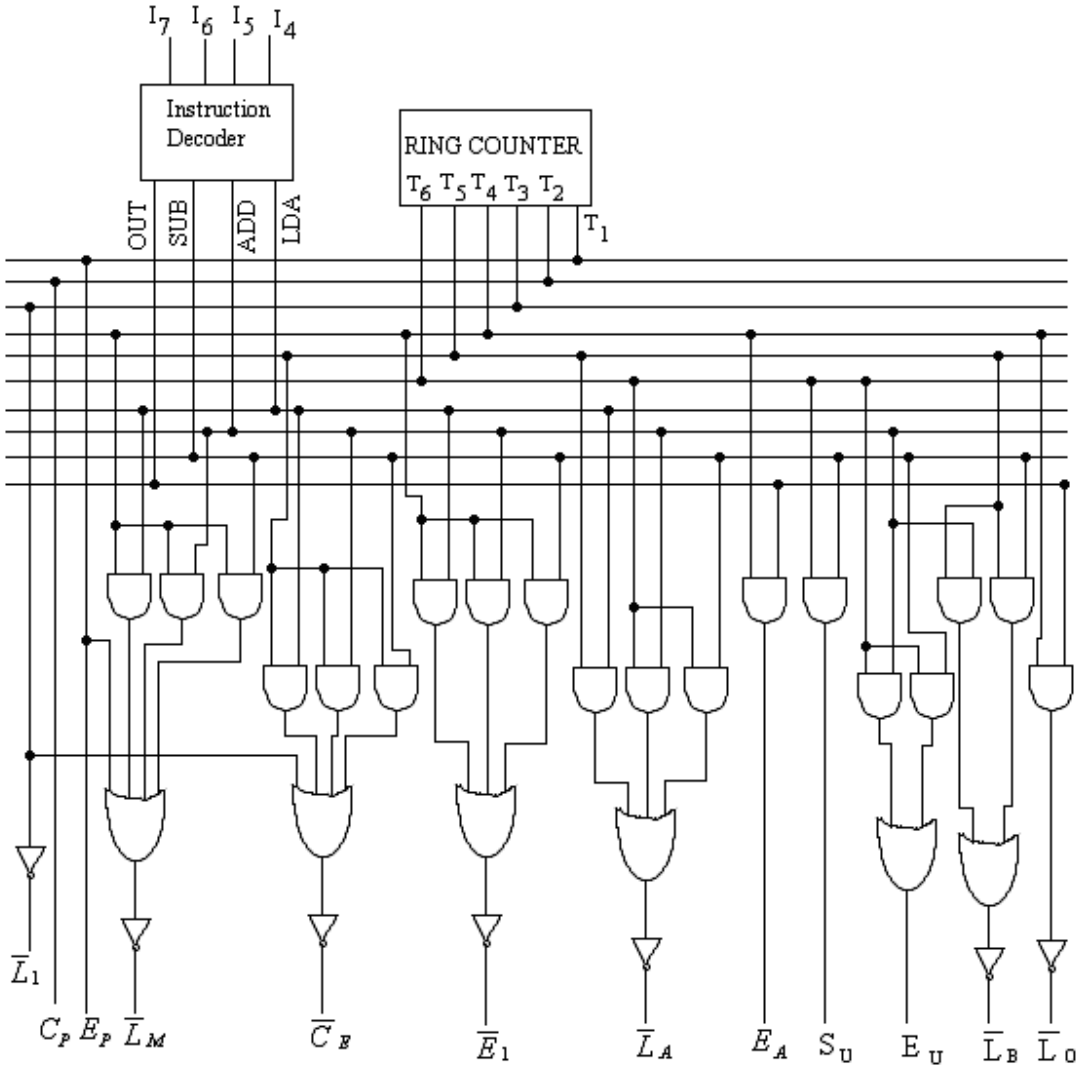


Fig. 2.19

2.5.5 Accumulator

Figure 2.20 shows the logic diagram of 8-bit buffer register known as accumulator which is designed using two ICs' IC1 and IC2 (both 74LS173). These ICs are configured as two-state output. The outputs of both the two ICs directly go to Adder-Subtractor. Three state switches send the content of accumulator to W-Bus when the signal E_A is high.

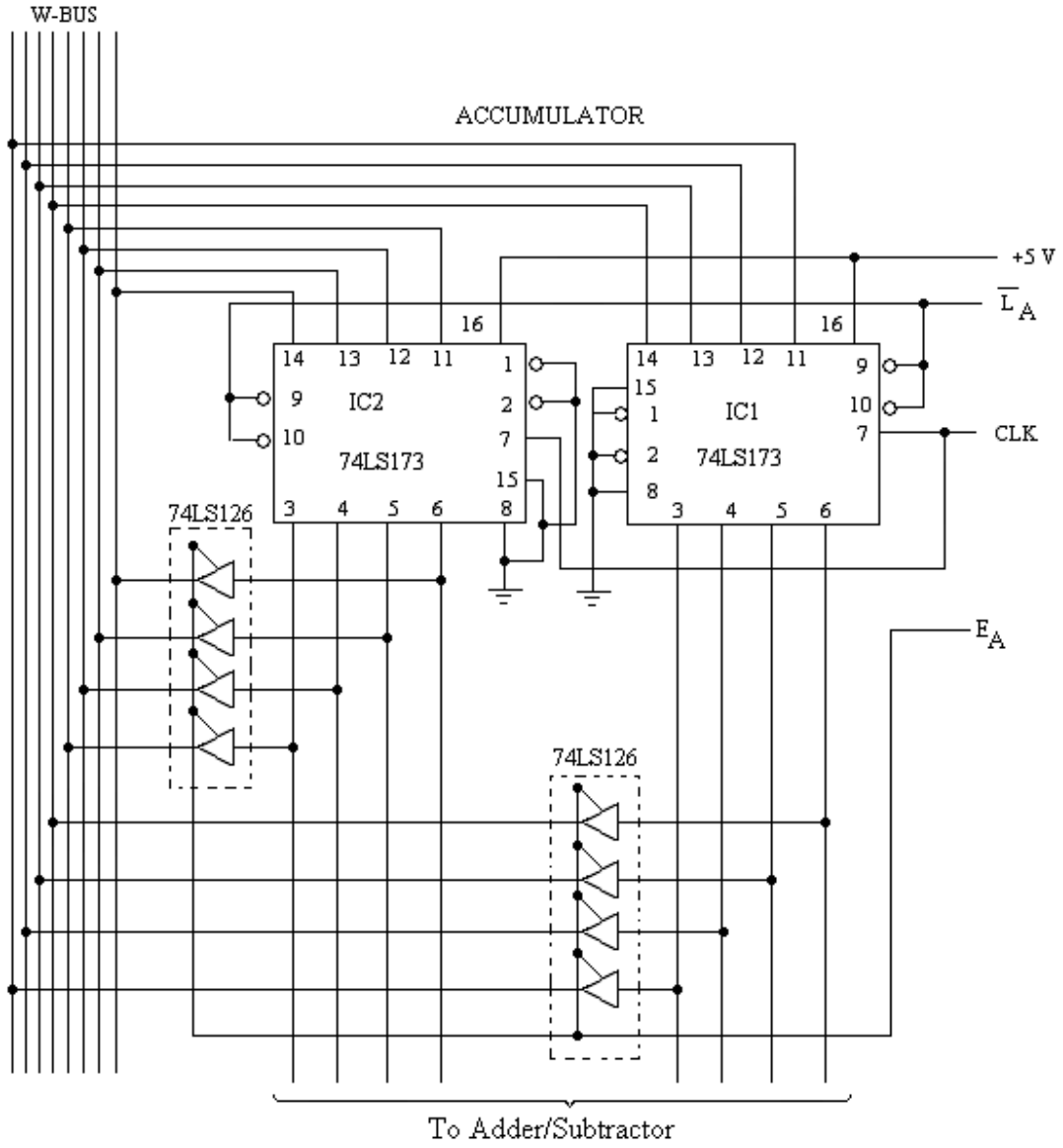


Fig. 2.20

2.5.6 Adder-Subtractor

Figure 2.21 shows the logic circuit used for Adder-subtractor of SAP-I computer. It basically consists of two 2's complement adder/subtractor ICs (7483). The data from the accumulator directly goes to the Adder-subtractor as shown in figure 2. . The outputs of B-register are also connected to these two ICs through eight exclusive-OR gates. The one terminal each of the eight exclusive-OR gates is being used as control Input (S_U). When S_U is low the content of B-register is directly loaded to Adder/Subtractor i.e. it will act as simple adder. If on the other hand S_U is high, 1's complement of B-register is being added to adder/subtractor ICs. Logic 1 is also added to LSB to form its as 2's complement (it therefore works as 2's complement subtractor). These two

adder/subtractor ICs give the 8-bit addition or subtraction. The 8-bit three state switches load the answer (sum or difference) to W-bus.

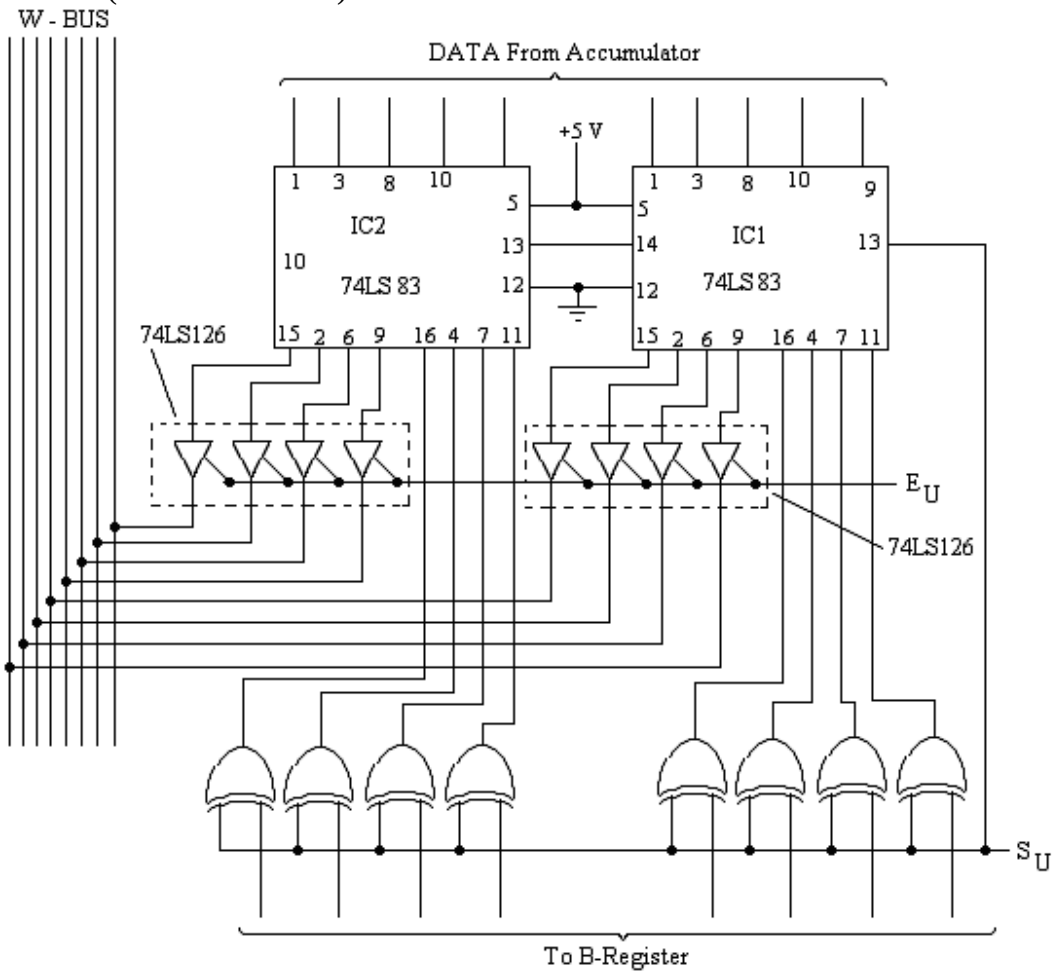


Fig. 2.21

2.5.7 B-Register

Similar to Accumulator, B-register is also designed using the Buffer register ICs 74LS173. The logic circuit diagram of this register is shown in figure 2.22. These two ICs are configured as two state register. When $\overline{L}_B$ is low, the data from the W-Bus is loaded to B-register and then transferred to Adder/Subtractor.

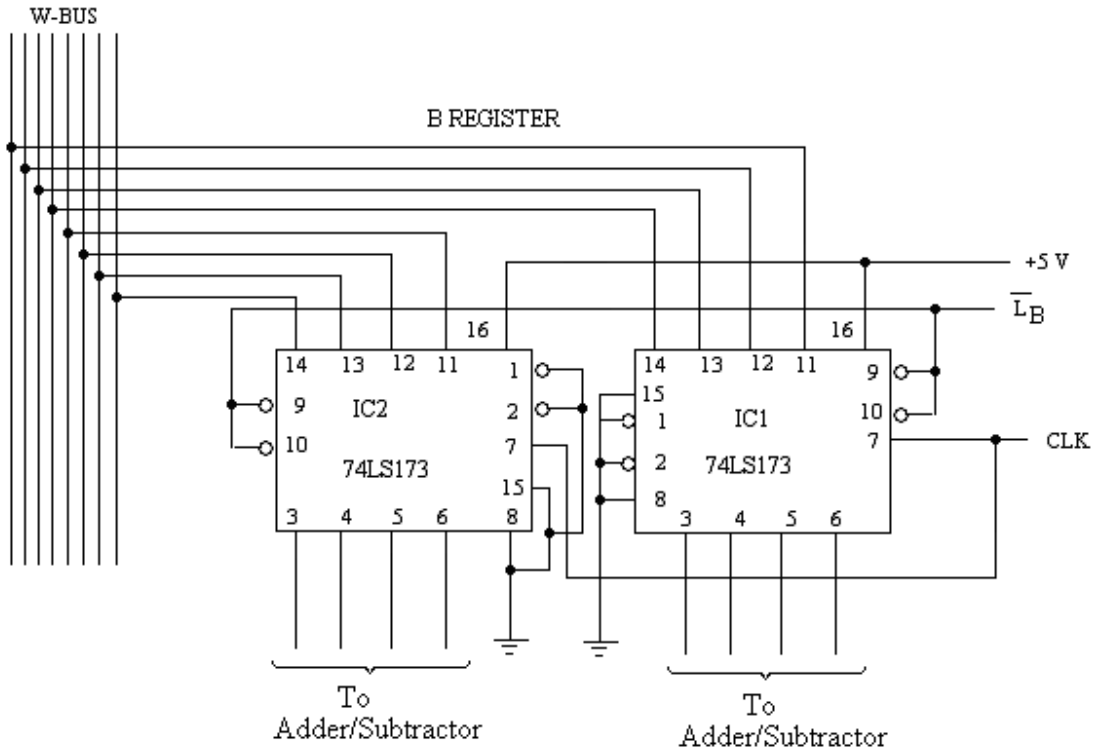


Fig. 2.22

2.5.8 Output Register

Output register is also formed by two state Buffer register using 74LS173 ICs. The output register with Binary display unit of SAP-I computer is shown in figure 2.23. When $\overline{L}_O$ is low, the data from the W-Bus will be loaded to output register. The output terminals of output register are connected to 8 LEDs, which will glow as per the data available at the output register.

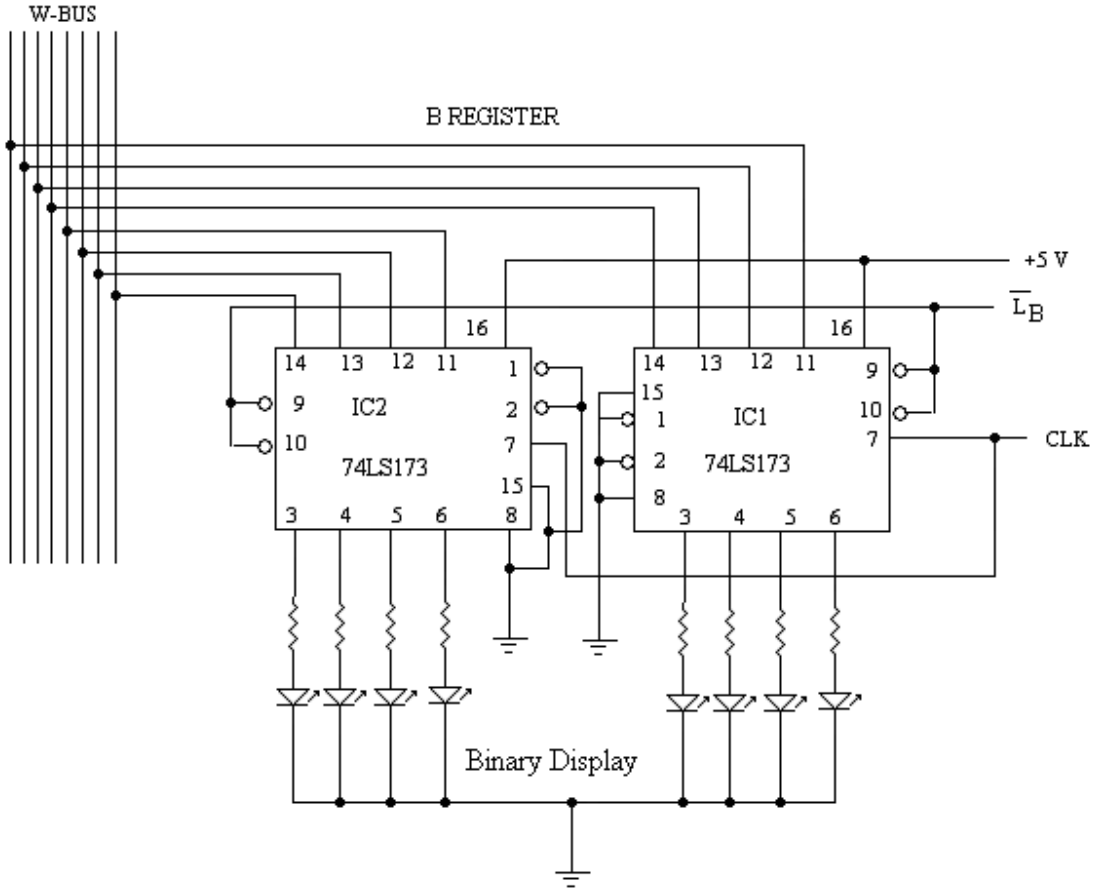


Fig.2.23

Problems

2.1 Draw the block diagram of SAP-I computer. Discuss the function of each block.

2.2 What is the function of Controller/Sequencer in SAP-I computer? How a control word is generate in SAP-I computer?

2.3 Draw the logic circuit diagram of Program counter and discuss its function.

2.4 Explain with the help of timing diagram the fetch and execution cycle of LDA-instruction.

2.5 Explain with the help of timing diagram the fetch and execution cycle of ADD-instruction.

2.6 Explain with the help of timing diagram the fetch and execution cycle of SUB-instruction.

2.7 Explain with the help of timing diagram the fetch and execution cycle of OUT-instruction.

2.8 Explain with the help of timing diagram the fetch and execution cycle of HLT-instruction. Further explain what will happen if HLT instruction is not given at the end of the program?

2.9 Differentiate between:

(i) Micro-instruction and Macro-instruction

(ii) Assembly language and machine language

(iii) Source program and object program

2.10 Explain the instruction set of SAP-I computer. What is the size of MAR of the SAP-computer?

2.11 Why the positive clock edge occurs half way through each T-state in SAP-I computer?

2.12 What is the role of instruction register in SAP-I computer? Draw the logic diagram for Instruction register of SAP-I computer?

2.13 Write an assembly language program that perform the following operation in SAP-I computer.

$$8-3+5+2-4$$

Use memory locations 7H to BH for the data. Write also its program in machine language.

2.14 Write an assembly language program for SAP-I computer that will display the result of $9+3-2$.

2.15 Write an assembly language program for SAP-I computer that will display the result of $7+3-2+4-1+6$.

Use AH to FH memory locations for starting the data.

2.16 Write an assembly language program for calculating the following expression $A+3B-2C$ on SAP-I computer. The data A, B and C are stored in memory locations 9 H to B H. Write also its machine language.

2.17 What are microinstructions for SUB and OUT instructions of SAP-I computer. Express the answer in binary also.

2.18 The timing diagram for Fetch and Execution cycle of ADD instruction is shown in figure 2.24. What are the microinstructions for Fetch and Execution cycle of ADD instruction?

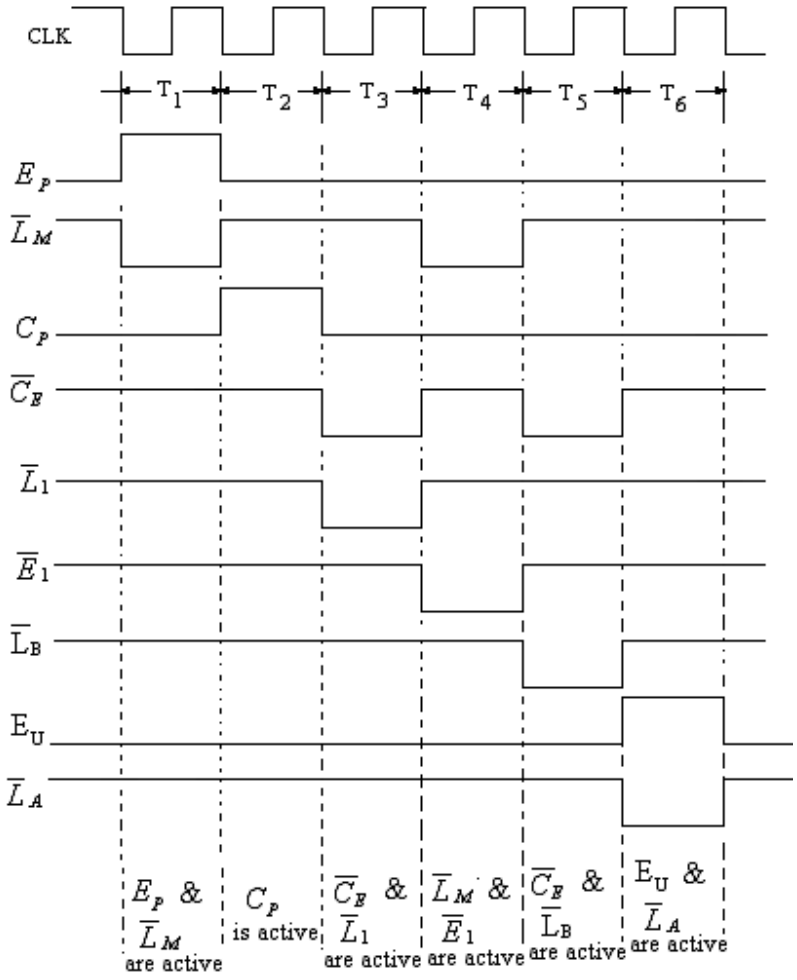

CLK
T1
T2
T3
T4
T5
T6
E_P
$\overline{L}_M$
C_P
$\overline{C}_E$
$\overline{L}_1$
$\overline{E}_1$
$\overline{L}_B$
E_U
$\overline{L}_A$
E_P & $\overline{L}_M$ are active
C_P is active
$\overline{C}_E$ & $\overline{L}_1$ are active
$\overline{L}_M$ & $\overline{E}_1$ are active
$\overline{C}_E$ & $\overline{L}_B$ are active
E_U & $\overline{L}_A$ are active

Fig. 2.24

3

SAP – II

In the preceding chapter of this book, the architecture, Instruction set and Programming of a conceptual computer named SAP-I computer was discussed. In order to understand the basic knowledge of a computer, it was the first step in the evolution towards the modern computers. The further step in this direction is SAP-II computer. The details of SAP-II computers will now be discussed.

3.1 ARCHITECTURE OF SAP-II COMPUTER

The architecture of another conceptual computer named as SAP-II computer is shown in figure 3.1. The SAP-II computer has more registers and other blocks, for giving more flexibility in programming. The architectural details of the Simple as Possible computer SAP-II are basically same as that of SAP-I, but SAP-II has more additional features which are given below:

(1) It is an eight bit computer, as 8-bit data can be processed in this computer. The SAP-I computer was also 8-bit computer.

(2) It has bidirectional BUS shown by double headed arrows, which indicate that the data can move either way after having the proper control signal. In SAP-I computer separate BUS is used for each way.

(3) The memory size of SAP-II computers is 64K (65536) words and each word is of 8-bit long. The data may be loaded or retrieved from the memory, through a buffer register known as Memory Data Register (MDR) or Memory Buffer Register (MBR).

The address for the location is 16 bits as $64K = 65536 = 2^{16}$.

The address will start from:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

to

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

The hexadecimal equivalent of the address is

0 0 0 0 H to F F F F H.

However, memory size of SAP-I computers was only 16. As $2^4 = 16$, so 4 address lines were used.

(4) The width of the W-BUS is 16, since it has to carry a 16 bit address. The width of W-bus for SAP-I computer was only of 8-bits.

(5) The program counter of SAP-II computer is 16-bit wide as it has to send the address of 16 bits to W-BUS.

(6) Memory Address Register (MAR) of SAP-II computer is of 16 bits as it receives the address of 16-bits from the W-BUS.

(7) The controller sequencer of SAP-II computer generates a control word of bigger size than that of SAP-I computer; since it has to control more registers and blocks of the computer. The control word of SAP-I computers was of 12 bits.

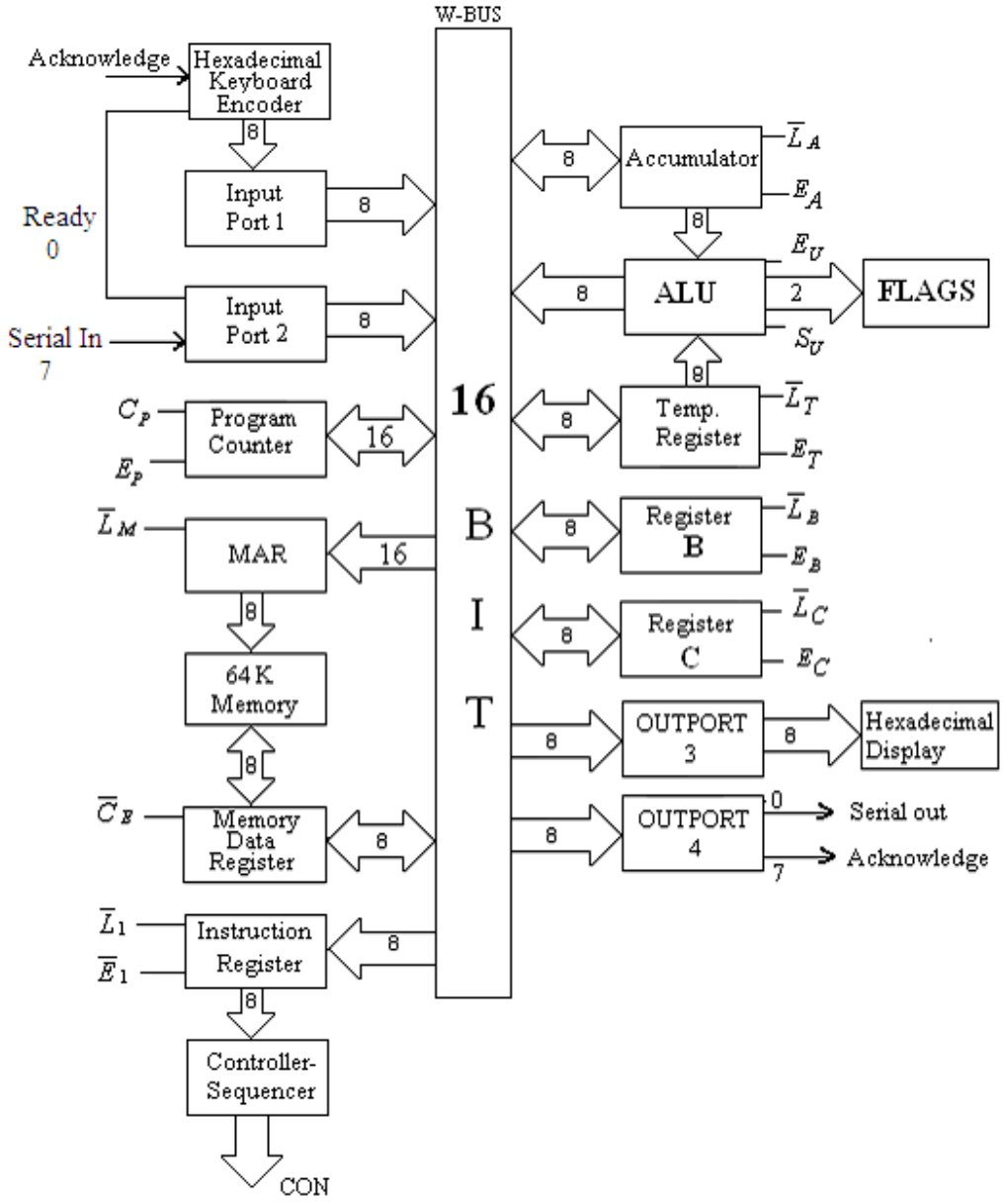


Fig. 3.1

(8) There are three registers in SAP-II computers namely Accumulator (Register A), Register B and Register C. All these registers are 8-bit registers. The registers B and C are general purpose registers. These registers give more flexibility in moving the data from one register to other register during the computer run. In SAP-I computer B-register was available for holding the data being added to or subtracted from the

accumulator. In SAP-II computers, there is a temporary register (T-register) for this purpose.

(9) SAP-II computer has Arithmetic and Logical unit (ALU) which can perform the arithmetic and logical operations on 8 bit data. Two flags (Sign flag **S** and Zero flag **Z**) are available with ALU. The accumulator content goes negative or zero during execution of some instructions. This affects the sign and zero flags. Figure 3.2 shows the circuit used for setting of flags of SAP-II computers.

It is clear from this figure that when accumulator content is zero all the bits (A_0 to A_7) are zero and the output of NOR gate is one. This NOR gate derives the AND gate (numbered 2), if the gating signal L_F is high and the flag will be updated. Similarly, if the accumulator content is negative, most significant bit (A_7) of the accumulator will be one which drives the lower AND gate (numbered 3) and sign flag will be updated when L_F is high. The sign flag will be set if accumulator content is negative and reset if positive. The zero flag will be set if accumulator content is zero and reset if it is non zero.

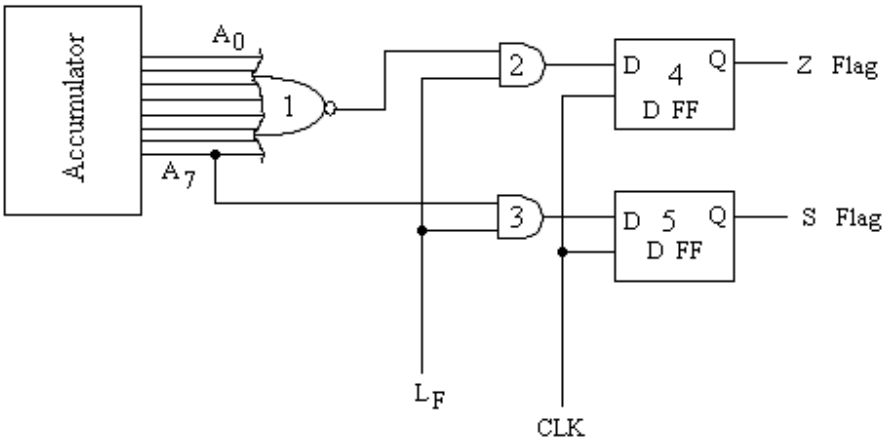


Fig. 3.2

(10) There are two input ports (Input Port-1 and Input Port-2) in SAP-II computers as shown in figure 3.1. A hexadecimal keyboard encoder is connected to Port 1. The instruction and data fed to the computer through this input port 1. The encoder sends a READY signal to bit 0 of Input Port-2. The data may also be accepted in serial form through the Input Port-2.

(11) SAP-II computer has two Output Ports namely output port-1 and output port-2. Through the output port-1, the processed data (answer from accumulator) can be visualized on the hexadecimal display. The accumulator content can also be sent to output port serially.

(12) Because of the availability of more registers and large size of memory, SAP-II computer has an instruction set with 42 instructions. SAP-I computer has only 5 instruction in its instruction set.

3.2 INSTRUCTION SET OF SAP-II COMPUTERS

As already discussed SAP-II computer has 42 instructions in its instruction set which are listed in table 3.1. These instructions are divided into 5 different categories as:

Memory Reference Instructions
Register Instructions
Jump and Call instructions
Logical Instructions
Miscellaneous Instructions

Table 3.1

Category	**Instruction**	**Op. Code**	**Operation**
Memory Reference instructions	LDA address	3A	$[\mathrm{A}] \leftarrow [\mathrm{M}_{\mathrm{address}}]$
	STA address	32	$[\mathrm{M}_{\mathrm{address}}] \leftarrow [\mathrm{A}]$
	MVI A, data	3E	$[\mathrm{A}] \leftarrow$ Data
	MVI B, data	06	$[\mathrm{B}] \leftarrow$ Data
	MVI C, data	0E	$[C] \leftarrow Data$
Register Instructions	MOV A, B	78	$[A] \leftarrow [B]$
	MOV A, C	79	$[A] \leftarrow [C]$
	MOV B, A	47	$[B] \leftarrow [A]$
	MOV B, C	41	$[B] \leftarrow [C]$
	MOV C, A	4F	$[C] \leftarrow [A]$
	MOV C, B	48	$[C] \leftarrow [B]$
	ADD B	80	$[A] \leftarrow [A] + [B]$
	ADD C	81	$[A] \leftarrow [A] + [C]$
	SUB B	90	$[A] \leftarrow [A] - [B]$
	SUB C	91	$[A] \leftarrow [A] - [C]$
	INR A	3C	$[A] \leftarrow [A] + 1$
	INR B	04	$[B] \leftarrow [B] + 1$
	INR C	0C	$[C] \leftarrow [C] + 1$
	DCR A	3D	$[A] \leftarrow [A] - 1$
	DCR B	05	$[B] \leftarrow [B] - 1$
	DCR C	0D	$[C] \leftarrow [C] - 1$
Jump and Call instructions	JMP address	C3	$[PC] \leftarrow Address$
	JM address	FA	$[PC] \leftarrow Address$ **;** if acc. content is negative.
	JZ address	CA	$[PC] \leftarrow Address$ **;** if acc. content is zero.
	JNZ address	C2	$[PC] \leftarrow Address$ **;** if acc. content is not zero.
	CALL address	CD	$Stack \leftarrow [PC]$ and

			$[PC] \leftarrow Address$
	RET	C9	$[PC] \leftarrow Stack$
Logical instructions	CMA	2F	$A \leftarrow \overline{A}$
	ANA B	A0	$A \leftarrow A.AND.B$
	ANA C	A1	$A \leftarrow A.AND.C$
	ORA B	B0	$A \leftarrow A.OR.B$
	ORA C	B1	$A \leftarrow A.OR.C$
	XRA B	A8	$A \leftarrow A \oplus B$
	XRA C	A9	$A \leftarrow A \oplus C$
	ANI, data	E6	$A \leftarrow A.AND.data$
	ORI, data	F6	$A \leftarrow A.OR.data$
	XRI, data	FE	$A \leftarrow A \oplus data$
Miscellaneous instruction	NOP	00	No operation
	IN Port	DB	$[A] \leftarrow [Portadress]$
	OUT Port	D3	$[Outport] \leftarrow [A]$
	RAL	17	Rotate acc. left
	RAR	1F	Rotate acc. right
	HLT	76	Stops Processing

The operation to be performed by each instruction has been explained in table 3.1. After going through these instructions, it will be possible to write complicated programs in solving the different problems.

The SAP-II computer has memory reference instruction like 'STA address" which allows to store the content of accumulator to the memory location whose address is given with the instruction. For example if accumulator has the content $A = 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0$ before execution of an instruction. Let the instruction is

STA 2050 H

After execution of this instruction the memory location 2050 H will have the data

1 0 1 0 1 0 1 0.

i.e. $M_{2050H} = 10101010$

In register instructions, **MOV C, B** copies the content of B-register into C-register $(C \leftarrow B)$.

In Call instructions, the instruction 'CALL address' copies the present address of program counter to Stack and the given address with the instruction is copied into program counter. So whenever the next instruction is fetched, the program counter will send this new address. This instruction helps in jumping from the normal routine of the program to subroutine program. The instruction RET stands for return. This statement is used at the end of subroutine program and its sends back to the original program as the content of the stack is copied back to the program counter. In SAP-II computer there is, however, no register for Stack. The last two memory locations FFFE H and FFFF H are used for this purpose as shown in figure 3.3 i.e. these two locations are exclusively used for saving the return address of the subroutine program.

The SAP-II computer has the logical instructions such as 'ANA reg', 'ORA reg', 'XRA reg', 'ANI data', 'ORI data', 'XRI data' and 'CMA'. The 'ANA B' instruction

AND the contents of A register with the contents of B register bit by bit and the answer is loaded into accumulator.

In miscellaneous instructions, 'IN port' loads the data word from an input port to the accumulator.

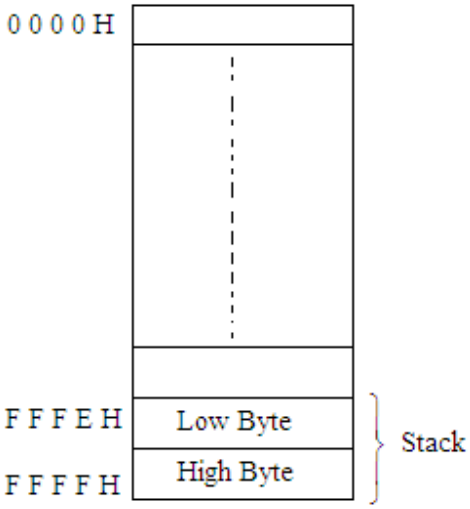


Fig. 3.3

The RAL instruction rotates the accumulator left. This instruction will shift all bits of accumulator content to the left and move the MSB into LSB position as illustrated in figure 3.4.

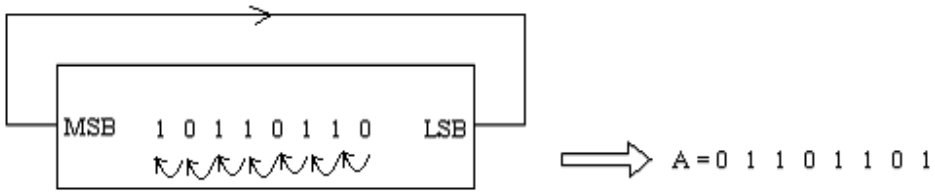


Fig.3.4

Similarly, the RAR instruction rotates the accumulator right. This instruction will shift all bits of accumulator content to the right and move the LSB into MSB position as illustrated in figure 3.5.

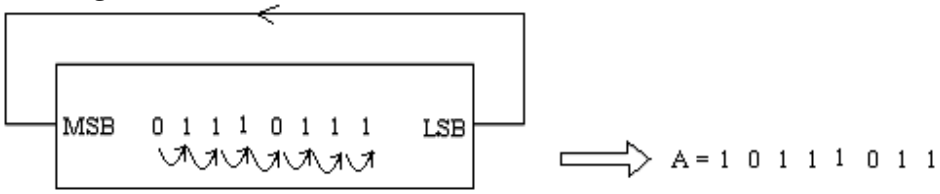


Fig. 3.5

3.3 MACHINE CYCLE AND INSTRUCTION CYCLE

It has been discussed in the last chapter that the SAP-I computer takes 6-T states to fetch and execute an instruction. These six states form a machine cycle. The number of T-states needed to fetch and execute an instruction is called Instruction cycle. So in SAP-I computer instruction cycle is machine cycle as shown in figure 3.6.

The SAP-II computer has, however, more than one machine cycle to fetch and execute an instruction. As shown in figure 3.7, first three T-states are used to fetch an

instruction and other 9 T-states are used for the execution of an instruction. So in SAP-II or other computers the instruction cycle may have two or more machine cycles. Table 3.2 shows the number of T-states used for the instruction cycle of each instruction.

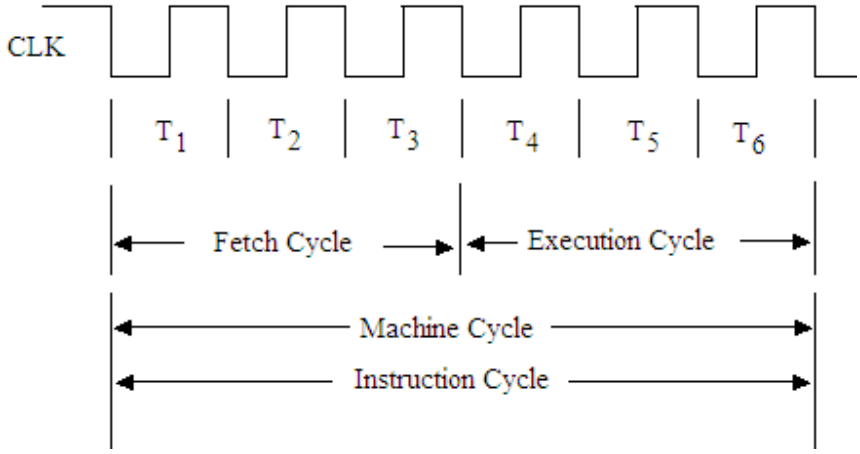


Fig. 3.6

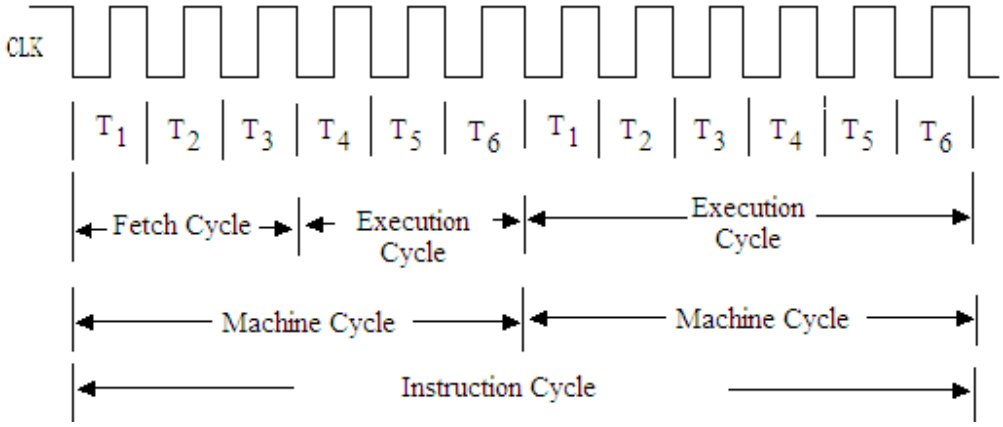


Fig. 3.7

The instruction 'STA address' takes 13 T-states to fetch and execute this instruction. If the system clock frequency is 3 MHz, then 'STA address' will take

$$13x\frac{1}{3}\mu\text{sec} = 4.33\mu\text{sec}$$

time to fetch and execute this instruction i.e. it is the time of instruction cycle of the instruction 'STA address'.

Table 3.2

Instruction	Addressing	No of T-states	Flags affected	No of bytes of instr.
LDA address	Direct	13	None	3
STA address	Direct	13	None	3
MVI A, data	Immediate	7	None	2
MVI B, data	Immediate	7	None	2
MVI C, data	Immediate	7	None	2

MOV A, B	Register	4	None	1
MOV A, C	Register	4	None	1
MOV B, A	Register	4	None	1
MOV B, C	Register	4	None	1
MOV C, A	Register	4	None	1
MOV C, B	Register	4	None	1
ADD B	Register	4	S, Z	1
ADD C	Register	4	S, Z	1
SUB B	Register	4	S, Z	1
SUB C	Register	4	S, Z	1
INR A	Register	4	S, Z	1
INR B	Register	4	S, Z	1
INR C	Register	4	S, Z	1
DCR A	Register	4	S, Z	1
DCR B	Register	4	S, Z	1
DCR C	Register	4	S, Z	1
JMP address	Immediate	10	None	3
JM address	Immediate	10/7	None	3
JZ address	Immediate	10/7	None	3
JNZ address	Immediate	10/7	None	3
CALL address	Immediate	18	None	3
RET	Implied	10	None	1
CMA	Implied	4	None	1
ANA B	Register	4	S, Z	1
ANA C	Register	4	S, Z	1
ORA B	Register	4	S, Z	1
ORA C	Register	4	S, Z	1
XRA B	Register	4	S, Z	1
XRA C	Register	4	S, Z	1
ANI, data	Immediate	7	S, Z	2
ORI, data	Immediate	7	S, Z	2
XRI, data	Immediate	7	S, Z	2
NOP	-	4	--	1
IN Port	Direct	10	None	2
OUT Port	Direct	10	None	2
RAL	Implied	4	None	1
RAR	Implied	4	None	1
HLT	-	5	--	1

3.4 ADDRESSING MODES

There are various techniques to specify the data for instruction. These techniques are called addressing modes. SAP-II computer has the following addressing modes:

1. Direct Addressing
2. Register Addressing
3. Immediate Addressing
4. Implied Addressing

Direct Addressing

In this mode of addressing, the address of the operand is given in the instruction itself.

For example LDA 2100 H
OUT 03 H etc.

Register Addressing

In this mode of addressing, the operands are in registers.

For example MOV A , B
ADD C etc.

Immediate Addressing

In immediate addressing mode, the operand in specified in the instruction itself.

For example MVI B, 08 H
ANI 07 H etc.

Implied Addressing

There are certain instructions which operate on the contents of the accumulator. Such instructions do not require the address of the operand, since the operand is implied in the instruction itself. So this type of addressing mode is called as Implied Addressing.

For example CMA
RAL
RAR etc.

The addressing modes of all the instructions of SAP-II computers are given in table 3.2.

3.5 INSTRUCTION TYPES

The instruction set of SAP-II computer may be divided into three types. As already seen, there are different ways for specifying the data for instructions, so the machine codes of all instructions are not of same length. Following are the types of instructions:

(i) One Byte Instruction
(ii) Two Byte Instruction
(iii) Three Byte Instruction

One Byte Instruction

This type of instruction has only op code part of one byte and no operand is given. The instruction length is only of one byte. It can be stored only in one memory location.

For example MOV A, C
ADD C
CMA
RAL
RAR etc.

If 'MOV A, C' instruction is to be stored in some location say 2000 H, then its op code of one byte is to be fed in this memory location.

i.e. 2000 H 79 H

where 79 H is the op code of the instruction 'MOV A, C'.

Two Byte Instruction

In a two byte instruction, first byte of the instruction is its op code and second byte is the given data.

Such instruction is stored in two consecutive memory locations.

For example MVI A, 06 H
OUT 03 H
ANI 76 H etc.

In order to store the instruction say 'MVI A, 06 H' in the memory locations of the computer, we have to use two consecutive memory locations. In one memory location the op code of MVI A is to be stored and in the second location the data 06H is to be stored. This type of instruction to be stored in two locations say in 2101 H and 2102 H is given below:

2101H	3E H	(op code of MVI A)
2102H	06 H	(given data)

Three Byte Instruction

In a three byte instruction, first byte is used for its op code and second and third bytes are used for 16 bit address. Such an instruction is stored in three consecutive memory locations.

For example LDA 2100 H
STA 3000 H
JMP 2500 H etc.

In order to store the instruction say 'LDA 2100 H' three consecutive memory locations are to be used. In the first memory location op code of the instruction is stored, in second location lower byte of the address is to be stored and in the third byte upper byte of the address is to be stored. This instruction loaded in three consecutive memory location 2000H, 2001H and 2002H is given below:

2000H	3A H	(op code of LDA)
2001H	00 H	(Lower byte of address 2100 H)
2002H	21 H	(Upper byte of address 2100 H)

The type of instructions of all the 42 instructions of SAP-II computer is also given in table 3.2.

3.6 FLAGS

As already discussed, there are two flags associated with the accumulator of SAP-II computer, sign flag (S) and Zero flag (Z). These flags may be set or reset during certain instructions. If the accumulator content is negative after execution of certain instruction the sign flag is set otherwise it is reset. Similarly, if the accumulator content is zero after the operation of certain instruction the zero flag is set otherwise reset. These instructions which affect the flags are listed in table 3.3.

Table 3.3

Instruction	Flags affected
ADD	S, Z
SUB	S, Z
INR	S, Z
DCR	S, Z
ANA	S, Z
ORA	S, Z
ANI	S, Z
ORI	S, Z
XRI	S, Z

For example in ADD instruction say ADD B, the contents of B register gets added with the accumulator content and after addition the result is stored in accumulator. If the result is zero then zero flag is set otherwise reset. Similarly, if the result is negative the sign flag is set otherwise reset.

In INR or DCR instructions both sign and zero flags are affected. For example if there is an instruction INR B, the content of B register is incremented by sending the contents to accumulator and adds 1 to it. The result is then sent back to B register. If the accumulator goes negative while INR instruction is executed, the sign flag is set; and if the accumulator content is zero then the zero flag is set.

3.7 ASSEMBLY LANGUAGE PROGRAMMING

The programming of the problem is generally written in assembly language. The assembly language is written in mnemonics. The mnemonics are the initials or short form of the English word of the operation to be performed by the instruction. Assembly language statements are written in standard format as given below:

Label Mnemonic Operand Comment

Label A label is a symbol or group of symbols used to represent an address of the location which is not specifically known at the time of program is written. The label can be one to six characters, the first character of which must be a letter. Following are the acceptable labels.
NEXT, BACK, DELAY, A2 etc.

Mnemonic Short form of the operation to be performed.

Operand Operand is the data on which the operation is performed. It can be a data, memory address, register or port address.

Comment The comment statement is started with the semicolon. It gives the idea of the program to the user. The comments are not the part of the machine language program.

The program written in assembly language can be converted to machine language by hand. For writing the program in machine language, the starting address, where the program is to be stored should be known. Now the op code of the instruction is to be written in first location (starting address) and in the consecutive memory locations data /address of the operand is written. While storing the address in the memory locations, lower byte of he address is stored first then the upper byte as discussed above.

Example 3.1 *Write assembly language program using the instructions of SAP-II computer of the following statement. Also write the program in machine language.*

Load the contents of memory locations 2100 H and 2101 H in B-register and C-register respectively. The content of memory locations 2100 H and 2101H are 16 H and 19 H respectively.

Solution.

Label Mnemonic Operand Comment

	LDA	2100H	; Loads the content of 2100H into accumulator.
	MOV B,	A	; moves the content of accumulator to B-register ($B \leftarrow A$).
	LDA	2101H	; Loads the content of 2100H into accumulator.
	MOV C,	A	; moves the content of accumulator to C-register ($C \leftarrow A$).
	HLT		; Stop processing.

The assembly language program may be converted to machine language by writing the op codes of the instructions as given below. Let the starting address of the program is 2000H.

Memory Address	**Content**	
2000 H	3A H	LDA 2100 H
2001 H	00 H	
2002 H	21 H	
2003 H	47 H	MOV B, A
2004 H	3A H	LDA 2101 H
2005 H	01 H	
2006 H	21 H	
2007 H	4F H	MOV C, A
2008 H	76 H	HLT
2101 H	16 H	Data
2102 H	19 H	Data

Example 3.2 *Write an assembly language program to find the 2's complement of a hexadecimal number. The hexadecimal number 6A H is stored in memory location 2100H and the answer is to be stored in 2101 H. Use SAP-II instructions to program it.*

Solution.

Label	**Mnemonic**	**Operand**	**Comment**
	LDA	2100 H	; Loads the content of 2100 H into accumulator.
	CMA		; Complements the accumulator content (1's complement).
	INR A		; 1 is added to the accumulator content to get the 2's complement.
	STA	2101 H	; Loads the accumulator contents into memory location 2101 H.
	HLT		; Stop processing.

Example 3.3 *Write an assembly language program using SAP-II instructions to add two numbers (decimal) 38 and 64, then subtract decimal number 3 from the sum. The final answer is to be stored in memory location 2100 H.*

Solution. First of all decimal numbers 38 and 64 should be converted to hexadecimal numbers, as it works in hexadecimal.

Decimal number 38 = 26 H

Decimal number 64 = 40 H

Label	Mnemonic	Operand	Comment
	MVI A,	26 H	; Loads the first number to accumulator.
	MVI B,	40 H	; Loads the second number to B-register.
	MVI C,	03 H	; Loads the third number to C-register.
	ADD B		; Adds the contents of B-register with the contents of accumulator and the answer is stored in A.
	SUB C		; Content of C gets subtracted from accumulator and difference is stored in A.
	STA	2100 H	; Answer is stored in 2100 H location.
	HLT		; Stop processing.

Example 3.4 *Write a program in assembly language for SAP-II computer to mask off the least significant 4 bits of a given hexadecimal number. The answer should be stored in memory location 2200 H. Let the given number is B3 H.*

Solution. The binary equivalent of B3 H is 1011 0011. The masking of the 4 least significant bits 0011 means to make 0011 to 0000. However, the four most significant bits should not be changed.

This can be done if the given number is ANDed with F0 H (1111 0000). In doing so when 4 most significant bits are ANDed with 1111 no change will be there, but 4 least significant bits will be 0000 as required. The assembly language program for this will be as follows:

Label	Mnemonic	Operand	Comment
	MVI A,	B3 H	; Loads the number B3H to accumulator.
	ANI	F0 H	; ANDs the accumulator with F0H and answer is loaded to accumulator.
	STA	2200 H	; Answer is stored in 2200 H location.
	HLT		; Stop processing.

Example 3.5 *Write a program in assembly language for SAP-II computer to load a number 79 H in B-register and mask off all bits except* A_2 *bit. The result is to be transferred to C-register.*

Solution. To mask off the third LSB (A_2 bit) the ANDing of the given content will be with FBH (11111011). The program will be as given below:

Label	Mnemonic	Operand	Comment
	MVI B,	79 H	; Loads the number 79 H to accumulator.
	MOV A,	B	; Moves the content of B-register to accumulator.

	ANI	FB H	; ANDs the accumulator with FB H and answer is loaded to accumulator.
	MOV C,	A	; Answer is loaded to C-register.
.	HLT		; Stop processing.

Example 3.6 *Write a program in assembly language for SAP-II computer to interchange (swap) the contents of two memory locations 2100 H and 2101 H.*

Solution. The program is given below which is self explanatory.

Label	Mnemonic	Operand	Comment
	LDA	2100H	; Loads the content of 2100H into accumulator.
	MOV B,	A	; Moves the content of Acc to B-register.
	LDA	2101 H	; Loads the content of 2101H into accumulator.
	STA	2100 H	; Loads the acc. content to 2100 H location.
	MOV A,	B	; Moves the content of B-register to Acc.
	STA	2101 H	; Loads the acc. content to 2101 H location.
	HLT		; Stop processing.

Example 3.7 *Write a program in assembly language for SAP-II computer to multiply two decimal numbers 23 and 9 and store the answer in some memory location. Also write this program in machine language.*

Solution. Hexadecimal equivalent of decimal number 23 is 17 H.

The multiplication of these two numbers may be obtained by adding 23 (17 H) by 9 times in the accumulator which should be 00 H at the beginning. So the program of this problem may be as given below:

Label	Mnemonic	Operand	Comment
	MVI A,	00 H	; Loads the acc. 00 H. $[A] \leftarrow 00H$.
	MVI B,	17 H	; Loads B-register with 17H $[B] \leftarrow 17H$.
	MVI C,	09 H	; Loads C-register with 09H $[C] \leftarrow 09H$.
AGAIN	ADD B		; Adds the content of B-register to acc.
	DCR C		; Decrements C-register.
	JZ	END	; Checks for zero; if zero jump to END.
	JMP	AGAIN	; Repeats the addition.
END	STA	2100	; Stores the answer to memory location 2100 H.
	HLT		; Stop processing.

The program can be written in machine language stating at address 2000 H as given below:

Memory Address	Content	
2000 H	3E H	MVI A, 00 H
2001 H	00 H	
2002 H	06 H	MVI B, 17 H
2003 H	17 H	
2004 H	0E H	MVI C, 09 H
2005 H	09 H	
2006 H	80 H	ADD B
2007 H	0D H	DCR C
2008 H	CA H	JZ 200E H
2009 H	0E H	
200A H	20 H	
200B H	C3 H	JMP 2006 H
200C H	06 H	
200D H	20 H	
200E H	32 H	STA 2100 H
200F H	00 H	
2010 H	21 H	
2011 H	76 H	

Alternative method of writing this program using JNZ is as given below:

Label	Mnemonic	Operand	Comment
	MVI A,	00 H	; Loads the acc. 00 H. $[A] \leftarrow 00H$.
	MVI B,	17 H	; Loads B-register with 17 H $[B] \leftarrow 17H$.
	MVI C,	09 H	; Loads C-register with 09 H $[C] \leftarrow 09H$.
AGAIN	ADD B		; Adds the content of B-register to acc.
	DCR C		; Decrements C-register.
	JNZ	AGAIN	; Repeats the addition if the content of C-register is not zero.
	STA	2100 H	; Stores the answer to memory location 2100 H.
	HLT		; Stop processing.

This program can also be written by using subroutine program as follows:

Main Program

Label	Mnemonic	Operand	Comment
	MVI A,	00 H	; Loads the acc. 00 H. $[A] \leftarrow 00H$.
	MVI B,	17 H	; Loads B-register with 17 H $[B] \leftarrow 17H$.
	MVI C,	09 H	; Loads C-register with 09H $[C] \leftarrow 09H$.
	CALL	MUL	; Calls subroutine program for multiplication.

	STA	2100 H	; Stores the answer to memory location 2100 H.
	HLT		; Stop processing.

Subroutine Program

Label	Mnemonic	Operand	Comment
MUL	ADD B		; Adds the content of B-register to acc.
	DCR C		;Decrements C-register.
	JNZ	MUL	; Repeats the addition if the content of C-register is not zero.
	RET		; Returns to main program.

3.8 DELAY CALCULATIONS

Through programming time delay can be introduced, which is very useful in various applications such as digital clocks, traffic controls, digital process control and other data transfer controls. Time delay can be introduced by loading the registers with some desired number and then decremented through the loops to zero value. The delay introduced in the system will depend on the clock period of the system and the number of times the instructions are executed inside the loop.

To generate very small delay only one register can be used. Consider a subroutine program given below in which C-register is loaded with 10 H (decimal number 16). It is decremented in a loop to make it zero. Figure 3.8 shows the flow chart for the same.

Label	Mnemonic	Operand	No. of T-states
	MVI C,	10 H	7
LOOP	DCR C		4
	JNZ	LOOP	10/7
	RET		10

In this program the instruction MVI C, 10 H is executed only once and it takes 7 T-states to execute.

The instruction DCR C is executed 16 times and thus takes $16 \times 4 = 64$ T-states, since DCR instruction takes 4 T-states for its execution.

For the execution of JNZ instruction, it will go to loop 15 times, as the content of C-register will not be zero. So $15 \times 10 = 150$ T-states will be used in its calculation (till the content of C-register is not zero). When the contents of C-register becomes zero, it will jump to loop and it will take 7 T-states. For the execution of RET statement it will take 10 T-states.

Thus total number of T-states taken for the execution of this program will be given by:

Mnemonic	T-states
MVI C, 10 H	$7 \times 1 = 7$
DCR C	$4 \times 16 = 64$
JNZ	$10 \times 15 + 1 \times 7 = 157$
RET	$1 \times 10 = 10$
Total	**238 T-states**

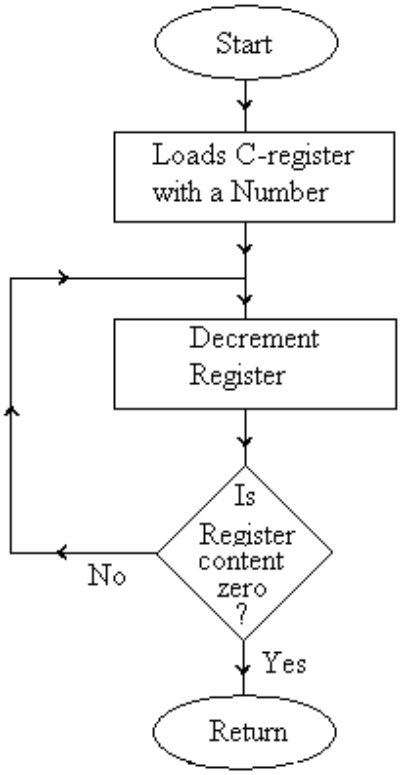


Fig. 3.8

Total 238 T-states are used for the execution of this program. If the system frequency is 2 MHz, the time taken by one T-state is $\frac{1}{2x10^6}\text{sec} = 0.5\mu\text{sec}$.

The total time delay introduced in the execution of this program is:

$$\text{Time delay} = 238x0.5\mu\text{sec}$$
$$= 119\mu\text{sec}$$
$$= 0.119\,\text{msec}$$

Maximum delay with this register C may be obtained by loading the number FF H (decimal number 255) in C-register. The maximum delay thus is:

MVI C, FF H	$7\,\text{x}\,1 = 7$
DCR C	$4\,\text{x}\,255 = 1020$
JNZ	$10\,\text{x}\,254 + 1\,\text{x}\,7 = 2547$
RET	$1\,\text{x}\,10 = 10$
Total	**3584 T-states**

$$\text{Time delay} = 3584x0.5\mu\text{sec}$$
$$= 1.792\,\text{msec}$$

Total delay introduced by the computer using only one register may be given in the following general form as:

$$T_{Delay} = [1x7 + 4N + (N-1)x10 + 1x7 + 10]x \text{ time of one T-state.}$$

where N is the decimal number given with the register.

$$= [14N + 14]x \text{ time of one T-state.}$$
$$= 14[N + 1]x \text{ time of one T-state.}$$

As discussed earlier the total time delay introduced by a register is 1.79 millisec if system clock frequency is 2 MHz. To introduce more time delay two registers may be used as shown in the flow chart shown in figure 3. 9, in which register B is used for the outer loop and register C is used for the inner loop.

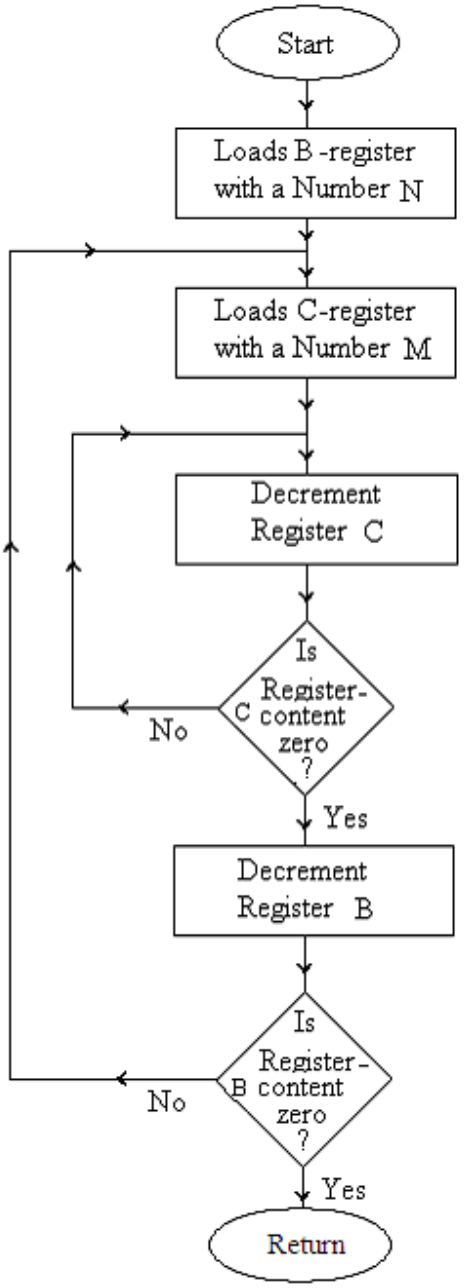


Fig. 3.9

Label	**Mnemonic**	**Operand**	**No. of T-states**
	MVI B,	M H	7
LOOP1	MVI C,	N H	7
LOOP	DCR C		4

JNZ	LOOP	10/7
DCR B		4
JNZ	LOOP1	10/7
RET		10

Total T-states used in this program will be given by:

$$= 1x7 + M\underbrace{[1x7}_{\text{MVI}} + \underbrace{Nx4}_{\text{DCR}} + \underbrace{(N-1)X10}_{\text{JNZ}} + \underbrace{1x7]}_{\text{JNZ}} + \underbrace{(Mx4)}_{\text{DCR}} + \underbrace{(M-1)x10}_{\text{JNZ}} + \underbrace{1x7}_{\text{JNZ}} + \underbrace{1x10}_{\text{RET}}$$

$$= 1x7 + M[7 + 4N + 10N - 10 + 7] + 4M + 10M - 10 + 7 + 10$$

$$= 14 + M[14N + 4] + 14M$$

$$= 14MN + 18M + 14$$

This is the general form for calculating the number of T-states of the program given above, in which M is the counts in register-B (in decimal number) in outer loop and N is the counts in register-C (in decimal number) in inner loop.

Total time delay is therefore given by:

$$T_{Delay} = (14MN + 18M + 14)\text{x Time of one T-state}$$

Example 3.8 *Write a delay subroutine program for SAP-II computer to introduce a time delay of 1 millisec using only one register. Let the system frequency is 2 MHz.*

Solution. System frequency = 2 MHz

Time delay of T-state = 0.5 μsec.

We shall now find the value of N to be stored to the register.

$$T_{Delay} = 14[N + 1]x\,0.5\ \mu\text{sec}$$

$$1000\ \mu\text{sec} = 14[N + 1]x0.5\,\mu\text{sec}$$

$$N = \frac{1000}{7} - 1$$

$$= \frac{993}{7} \cong 142 \text{ decimal number}$$

$$142 = 8\text{E H}$$

So the program for the delay of 1 millisec is as given below:

Label	**Mnemonic**	**Operand**
	MVI C,	8E H
LOOP	DCR C	
	JNZ	LOOP
	RET	

Example 3.9 *Write a delay subroutine program for SAP-II computer to introduce a time delay of 10 millisec using two registers. Decimal number 10 may be stored in register B (for outer loop). Let the system frequency is 2 MHz.*

Solution. System frequency = 2 MHz

In this case M = 10_{10} = 0A H

N is to be calculated as:

$$T_{Delay} = (14MN + 18M + 14)\text{x Time of one T-state}$$

Time of T-state = 0.5 μsec

$$10ms = (14x10xN + 18x10 + 14)x0.5\mu S$$

$$10000\mu s = (140N + 194)x0.5\mu S$$
$$20000 = (140N + 194)$$
$$140N = (20000 - 194)$$
$$N = \frac{19806}{140} = 142 = 8E \text{ H}$$

The Subroutine program for the required time delay is therefore given as:

Label	**Mnemonic**	**Operand**
	MVI B,	0A H
LOOP1	MVI C,	8E H
LOOP	DCR C	
	JNZ	LOOP
	DCR B	
	JNZ	LOOP1
	RET	

Example 3.10 *(a) What will be the time delay introduced in the computer, if the following subroutine program is executed. Assume the system frequency is 2 MHz.*

	MVI B,	*64 H*
LOOP1	*MVI C,*	*8E H*
LOOP	*DCR C*	
	JNZ	*LOOP*
	DCR B	
	JNZ	*LOOP1*
	RET	

(b) How much maximum delay can be introduced by this subroutine program?
(c) Modify this program to introduce a time delay of 1 sec.
(d) Modify this program to introduce a time delay of 10 sec.

Solution. In this subroutine program M = 64 H = 100_{10}
N = 8E H = 142_{10}
Time of one T-state = 0.5 μsec

$$T_{Delay} = (14MN + 18M + 14) \text{ x Time of one T-state}$$
$$= (14MN + 18M + 14) \text{ x } 0.5\mu\text{sec}$$
$$= (14x100x142 + 18x100 + 14) \text{ x } 0.5\mu\text{sec}$$
$$= (198800 + 1800 + 14) \text{ x } 0.5\mu\text{sec}$$
$$= 200614 \text{ x } 0.5\mu\text{sec}$$
$$= 100307 \ \mu\text{sec}$$
$$= 100.3 \text{ msec}$$
$$\cong 0.1 \text{ sec}$$

Approx. 0.1 sec delay is introduced by this subroutine program.

(b) The maximum time delay that can be introduced by this subroutine program is calculated if we consider the maximum value of M and N as FF H.

i.e. M = FF H = 255_{10}
N = FF H = 255_{10}

$$T_{Delay} = (14MN + 18M + 14) \text{ x Time of one T-state}$$
$$= (14MN + 18M + 14) \text{ x } 0.5\mu\text{sec}$$

$$= (14x255x255 + 18x255 + 14) \text{ x } 0.5\mu\text{sec}$$
$$= 914954 \text{ x } 0.5\mu\text{sec}$$
$$= 457477 \ \mu\text{sec}$$
$$\cong 0.46 \text{ sec}$$

(c) To introduce a time delay of 1 sec, the given program can be run 10 times using one more register say A-register. The modified program is given below:

Label	**Mnemonic**	**Operand**
	MVI A,	0A H
LOPP3	MVI B,	64 H
LOOP2	MVI C,	8E H
LOOP1	DCR C	
	JNZ	LOOP1
	DCR B	
	JNZ	LOOP2
	DCR A	
	JNZ	LOOP3
	RET	

(d) To introduce a time delay of 10 sec, in the above program 64 H (100_{10}) can be taken in place of 0A H (10_{10}). The subroutine program will be as given below:

Label	**Mnemonic**	**Operand**
	MVI A,	64 H
LOPP3	MVI B,	64 H
LOOP2	MVI C,	8E H
LOOP1	DCR C	
	JNZ	LOOP1
	DCR B	
	JNZ	LOOP2
	DCR A	
	JNZ	LOOP3
	RET	

Example 3.11 *Write a program in assembly language of SAP-II computer to count continuously in hexadecimal from FF H to 00 H. Use subroutine program also to set up a one millisecond delay between each count and display the number at one of the output port. Assume the system frequency is 2 MHz.*

Solution. For its programming load a count 00 H in one register say B-register. Decrement the content of B-register, so that in B-register the counts are FF H. Display this count on one of the output port say 04 H. Then introduce a delay of 1 millisec in a subroutine program. After the delay, decrement the counts in B-register and proceed continuously as discussed above.

The program is given below:

Main Program

Label	**Mnemonic**	**Operand**	**Comment**
	MVI B,	00 H	; Loads the count 00 H in B-register. $[B] \leftarrow 00H$.

```
START   DCR B           ;  Decrements the counts of B-register.
        CALL   DELAY    ;  Calls the delay subroutine program.
        MOV A,  B       ;  Moves the content of B to A.
        OUT    PORT 04 H;  Displays the data at the output port.
        JMP    START
```

Subroutine Program (Delay program of 1msec)

```
Label   Mnemonic   Operand
DELAY   MVI C      8E H
LOOP    DCR C
        JNZ        LOOP
        RET
```

Subroutine program is the same as discussed in example 3.7.

Example 3.12 *Suppose SAP-II computer can input a data (one byte) from port 2. Write an assembly language program to find if the bit 1 (A_1) is zero or one. If the bit 1 is one, it should load the accumulator with ASCII Y otherwise ASCII N. The accumulator data should be available at output port 4. The hexadecimal number for ASCII Y is 59 H and for ASCII N is 4E H.*

Solution.

```
Label   Mnemonic   Operand   Comment
        IN         02 H      ;  Input a byte (data) from input port 02H.
        ANI        02 H      ;  Isolate bit A1.
        JZ         NO        ;  Jump if A1 is zero.
        MVI A,     59 H      ;  Loads Y to Acc.
        JMP        END       ;  Jump to END.
NO      MVI A,     4E H      ;  Loads N to Acc.
END     OUT        04 H      ;  Output is available at port 04 H.
        HLT                  ;  Stop processing.
```

Example 3.13 *Write a program in assembly language of SAP-II computer to subtract a number stored in memory location 2100 H from the number in memory location 2101 H using addition method. The result should be stored in memory location 2102 H. If the result is negative, memory location should be loaded with 00 H.*

Solution. In this problem subtraction is to be carried out using addition method. So we have to add 2's complement of the number in memory location 2101 H.

```
Label   Mnemonic   Operand   Comment
        LDA        2100 H    ;  Loads the Acc. the content stored in 2100 H.
        CMA                  ;  Takes 1's complement of the number in Acc.
        INR A                :  Acc. content is added with to get 2's complement in Acc.
```

```
        MOV B,    A          ; Moves the Acc. content to B-register.
        LDA       2101 H     ; Loads the Acc. the content stored in 2101 H (second number).
        ADD B                ; Adds 2's complement of the number with Acc.
        JM        END        ; Jump if the sum is negative.
        STA       2102 H     ; Stores the answer if positive.
        HLT                  ; Stop processing.
END     MVI A,    00 H       ; Moves 00 H to Acc.
        STA       2102 H     ; If answer is negative store 00 H to 2102 H.
        HLT                  ; Stop processing.
```

Example 3.14 *Write a program in assembly language using SAP-II instructions that inputs a byte from port 2 and determine if decimal number is even or odd. If the input byte is even load FF H otherwise 00 H to memory location 2500 H.*

Solution.

Label	Mnemonic	Operand	Comment
	IN	02 H	; Input a byte (data) from input port 02H.
	ANI	01 H	; Isolate bit A_0.
	JNZ	ODD	; Jump if Odd.
	MVI A,	FF H	; Loads FF H to Acc.
	JMP	END	; jump to END.
ODD	MVI A,	00 H	; Loads 00 H to Acc.
END	STA	2500 H	; Stores the answer in 2500 H.
	HLT		; Stop processing.

PROBLEMS

1. Draw the block diagram of the architecture of SAP-II computers. Discuss the working of each block.
2. What is the difference between the architectures of SAP-I and SAP-II computers?
3. How the sign and zero flags work in arithmetic and logic unit of SAP-II computers?
4. Name and discuss the five different categories in which the instruction set of SAP-II computers are divided.
5. Name and discuss the different addressing modes to specify the instruction of SAP-II computers.
6. Discuss Implied addressing to specify the data of instructions of SAP-II computers.
7. Describe with examples one byte, two byte and three byte instructions of SAP-II computers.
8. What is the difference between assembly language program and machine language program? What do you understand by mnemonics?

9. How delay is introduced through software in SAP-II computers using only one register? How much maximum delay can be introduced with one register if the frequency of the clock is (i) 1 MHz (ii) 3 MHz?
10. How much delay can be introduced using two registers in SAP-II computers, if the system frequency is 2 MHz?
11. Write the subroutine program to introduce a delay of 1 sec using all the three registers of SAP-II computers. Further assume that the system clock frequency is 2 MHz.
12. Write a program in assembly language using SAP-II instructions to multiply the two decimal numbers 13 and 10. The answer should be stored in memory location 2100 H.
13. Write a program in assembly language using SAP-II instructions to complement a number lying at 2100 H memory location. Store the complement at 2101 H.
14 Write a program in assembly language using SAP-II instructions to perform the following arithmetic operation:

$$X + Y - Z - W$$

where X, Y, Z and W are hexadecimal numbers stored in memory locations 2101 H to 2104 H. The final answer should be stored in 2100 H. Assume that there is no carry or borrow.

15. Write a program in assembly language using SAP-II instructions to get 2's complement of a number stored in memory location 2501 H. Store the answer at 2502 H .
16. Write a program in assembly language using SAP-II instructions to add decimal numbers 70 and 36. Answer is to be stored in memory location 2100 H.
17. Write a program in assembly language using SAP-II instructions to multiply two decimal numbers 33 and 6 and store the answer in memory location 2100 H. Use subroutine for multiplication.
18. Write a subroutine program (in assembly language of SAP-II) to introduce a time delay of 20 msec. Let the system frequency is 2 MHz.
19. Write a subroutine program (in assembly language of SAP-II) to introduce a time delay of 1 minute. Let the system frequency is 1 MHz.
20. Write an assembly language program using SAP-II instructions to mask off A_1 and A_2 bits of a given number. Let the given number is 6E H.

4

SAP – III

In this chapter, programming model and instruction set of SAP-III computer will be discussed. After the study of this computer we will be in a position to understand the details of the popular 8-bit microprocessor 8085 with ease.

4.1 PROGRAMMING MODEL OF SAP-III COMPUTER

One step head of the evolution of modern computers is the 8-bit microcomputer named as SAP-III computer. The CPU of this computer is upward compatible with 8085 microprocessor. The programming model or software model of SAP-III computer is given in figure 4.1. It consists of some more registers than SAP-II computers. In addition to Accumulator (A), B and C registers it contains four more 8-bit registers named as D, E, H and L registers. Because of these registers the flexibility in programming is much more. With special instructions the registers B, C, D, E, H and L may be used as extended register pairs B-C register pair, D-E register pair and H-L register pair, so that 16-bit data may be operated with these registers. It has 16-bit Program counter and 16-bit Stack pointer. The stack pointer is used for stack purposes which will be discussed later in this chapter.

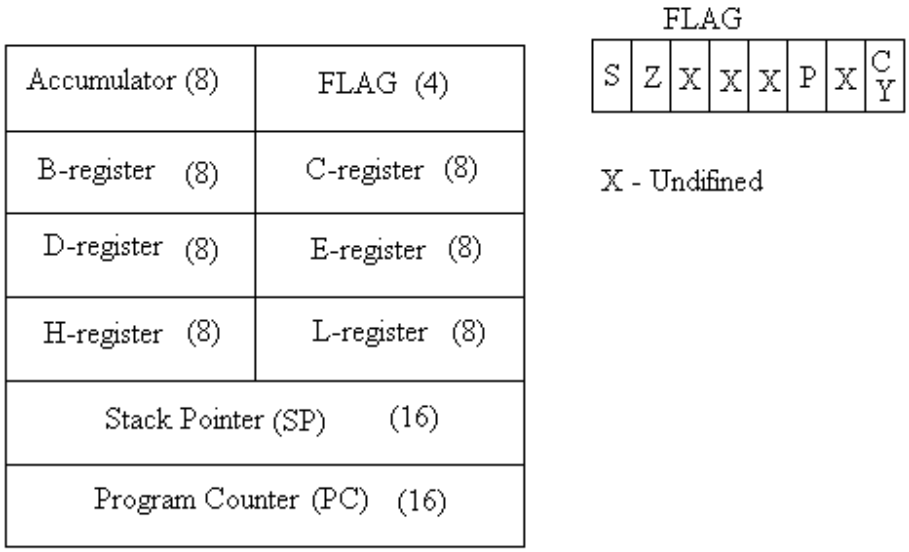


Fig. 4.1

SAP-III computer has a four-bit flag register associated with Accumulator. The four flags are Sign flag (S), Zero flag (Z), Carry flag (CY) and Parity flag (P).

Sign Flag (S)

The sign flag is set (S = 1), if accumulator content is negative and it is reset (S = 0) if accumulator content is positive. The logic circuit is the same as discussed for SAP-II computers.

Zero Flag (Z)

The zero flag is set (Z = 1), if accumulator content is zero and it is reset (Z = 0) if accumulator content is not zero. The logic circuit is the same as discussed for SAP-II computers.

Carry Flag (CY)

The carry flag (CY) is used to detect overflow in some arithmetic and logic operations. In SAP-III computer, the accumulator of CPU is only 8-bit wide. The unsigned binary numbers from 0 to 255 or signed numbers (2's complement) from – 128 to +127 can be the content of the accumulator. The arithmetic operations addition and subtraction are performed using 2's complement adder / subtractor circuit. The logic circuit diagram of such a 2's complement adder / subtractor is shown in figure 4.2.

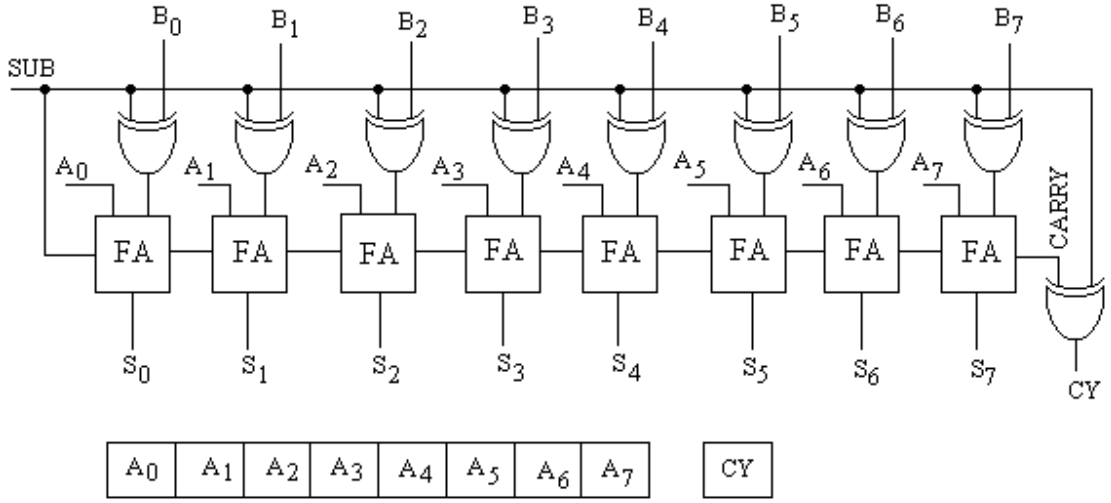


Fig. 4.2

In this circuit one input of each of the exclusive – OR gates are connected to common terminal named as SUB; this common terminal is also connected to carry bit of first full adder. During the addition, SUB terminal is kept low and for subtraction it is kept high. The working of this circuit may be described as given below.

The accumulator content directly goes to full adders. The content, which is to be added to or subtracted from the accumulator content is applied to full adders through the exclusive OR gates. SUB terminal being low during the addition, so the exclusive-OR gates send directly the addend to the full adders. The full adders add the two contents. If there is any overflow, CARRY becomes high which finally sets CY flag. If on the other hand there is no overflow, CARRY is low and CY flag is reset.

So CY = 1, if there is carry and CY = 0, if there is no carry.

Similarly, during subtraction SUB = 1 and the exclusive OR gates converts the subtrahend to its equivalent 1's complement. Since SUB terminal (SUB = 1) is connected to first full adder so 1's complement of the subtrahend gets converted to its equivalent 2's complement. For subtraction, 2's complement of the subtrahend is added with the

accumulator content. If the CARRY signal is high (CARRY = 1), there is an end around carry (EAC), the final exclusive OR gate gives CY = 0 indicating there is no borrow. On the other hand, if there is no CARRY (CARRY = 0 or no EAC), carry flag CY will be set (CY = 1) indicating that there is borrow.

So CY = 0 (Carry flag is restet) there is no carry or borrow.

CY = 1 (Carry flag is set) there is carry or borrow.

The carry flag acts as carry bit for addition and it works as borrow bit for subtraction.

Parity Flag (P)

In addition to sign, carry and zero flags there is also a parity flag. The parity flag is set if there are even number of 1's in the accumulator and it is reset if there are odd number of 1's.

So P = 1 for even parity

and P = 0 for odd parity.

The logic circuit diagram of the parity bit generator for showing the correct parity is shown in figure 4.3.

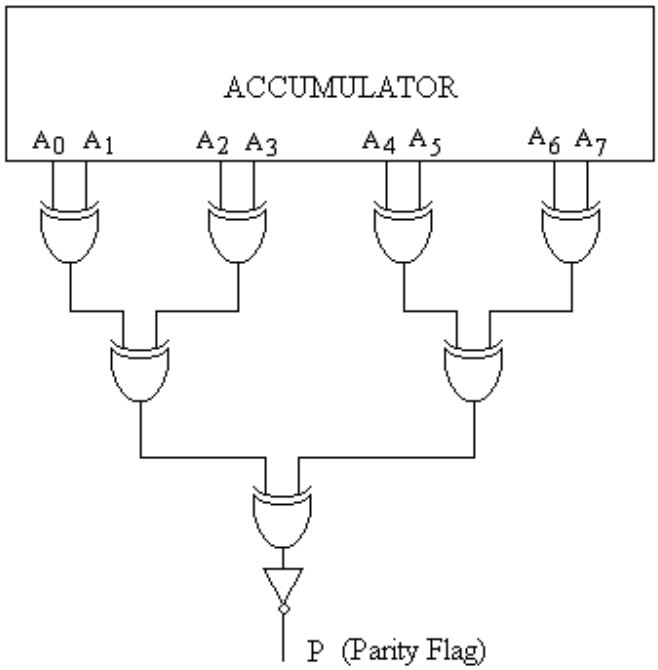


Fig. 4.3

4.2 INSTRUCTION SET OF SAP-III COMPUTER

All the instructions which have been discussed in SAP-II computers are also used in SAP-III computers. In addition to these instructions there are many more instructions which will be discussed here.

The instruction set of SAP-III computers has been classified into following groups.

1. Data Transfer Group
2. Arithmetic Group
3. Branch Group
4. Stack, Input/Output and Machine Control Group

4.2.1 Data Transfer Group

The function of data transfer group of instructions is to transfer the data from register to register, register to memory and also immediate transfer of data (given) to memory location. This group of instruction is also there in SAP-II computers but in SAP-III computers more instruction are there due to more registers. Data transfer group of instructions are given in table 4.1.

These data transfer instructions can further be subdivided on the basis of modes of addressing i.e. direct, immediate and register addressing.

Table 4.1

Data Transfer Group

Instruction	Op. Code	Addressing modes	No of T-states	No of bytes of instr.	Flags affected	Operation
LDA address	3A	Direct	13	3	None	$[A] \leftarrow [M_{address}]$
STA address	32	Direct	13	3	None	$[M_{address}] \leftarrow [A]$
MOV A, A	7F	Register	4	1	None	$[A] \leftarrow [A]$
MOV A, B	78	Register	4	1	None	$[A] \leftarrow [B]$
MOV A, C	79	Register	4	1	None	$[A] \leftarrow [C]$
MOV A, D	7A	Register	4	1	None	$[A] \leftarrow [D]$
MOV A, E	7B	Register	4	1	None	$[A] \leftarrow [E]$
MOV A, H	7C	Register	4	1	None	$[A] \leftarrow [H]$
MOV A, L	7D	Register	4	1	None	$[A] \leftarrow [L]$
MOV B, A	47	Register	4	1	None	$[B] \leftarrow [A]$
MOV B, B	40	Register	4	1	None	$[B] \leftarrow [B]$
MOV B, C	41	Register	4	1	None	$[B] \leftarrow [C]$
MOV B, D	42	Register	4	1	None	$[B] \leftarrow [D]$
MOV B, E	43	Register	4	1	None	$[B] \leftarrow [E]$
MOV B, H	44	Register	4	1	None	$[B] \leftarrow [H]$
MOV B, L	45	Register	4	1	None	$[B] \leftarrow [L]$
MOV C, A	4F	Register	4	1	None	$[C] \leftarrow [A]$
MOV C, B	48	Register	4	1	None	$[C] \leftarrow [B]$
MOV C, C	49	Register	4	1	None	$[C] \leftarrow [C]$
MOV C, D	4A	Register	4	1	None	$[C] \leftarrow [D]$
MOV C, E	4B	Register	4	1	None	$[C] \leftarrow [E]$
MOV C, H	4C	Register	4	1	None	$[C] \leftarrow [H]$
MOV C, L	4D	Register	4	1	None	$[C] \leftarrow [L]$
MOV D, A	57	Register	4	1	None	$[D] \leftarrow [A]$
MOV D, B	50	Register	4	1	None	$[D] \leftarrow [B]$

MOV D, C	51	Register	4	1	None	$[D] \leftarrow [C]$
MOV D, D	52	Register	4	1	None	$[D] \leftarrow [D]$
MOV D, E	53	Register	4	1	None	$[D] \leftarrow [E]$
MOV D, H	54	Register	4	1	None	$[D] \leftarrow [H]$
MOV D, L	55	Register	4	1	None	$[D] \leftarrow [L]$
MOV E, A	5F	Register	4	1	None	$[E] \leftarrow [A]$
MOV E, B	58	Register	4	1	None	$[E] \leftarrow [B]$
MOV E, C	59	Register	4	1	None	$[E] \leftarrow [C]$
MOV E, D	5A	Register	4	1	None	$[E] \leftarrow [D]$
MOV E, E	5B	Register	4	1	None	$[E] \leftarrow [E]$
MOV E, H	5C	Register	4	1	None	$[E] \leftarrow [H]$
MOV E, L	5D	Register	4	1	None	$[E] \leftarrow [L]$
MOV H, A	67	Register	4	1	None	$[H] \leftarrow [A]$
MOV H, B	60	Register	4	1	None	$[H] \leftarrow [B]$
MOV H, C	61	Register	4	1	None	$[H] \leftarrow [C]$
MOV H, D	62	Register	4	1	None	$[H] \leftarrow [D]$
MOV H, E	63	Register	4	1	None	$[H] \leftarrow [E]$
MOV H, H	64	Register	4	1	None	$[H] \leftarrow [H]$
MOV H, L	65	Register	4	1	None	$[H] \leftarrow [L]$
MOV L, A	6F	Register	4	1	None	$[L] \leftarrow [A]$
MOV L, B	68	Register	4	1	None	$[L] \leftarrow [B]$
MOV L, C	69	Register	4	1	None	$[L] \leftarrow [C]$
MOV L, D	6A	Register	4	1	None	$[L] \leftarrow [D]$
MOV L, E	6B	Register	4	1	None	$[L] \leftarrow [E]$
MOV L, H	6C	Register	4	1	None	$[L] \leftarrow [H]$
MOV L, L	6D	Register	4	1	None	$[L] \leftarrow [L]$
MOV A, M	7E	Register	7	1	None	$[A] \leftarrow [M_{H-L}]$
MOV B, M	46	Register	7	1	None	$[A] \leftarrow [M_{H-L}]$
MOV C, M	4E	Register	7	1	None	$[A] \leftarrow [M_{H-L}]$
MOV D, M	56	Register	7	1	None	$[A] \leftarrow [M_{H-L}]$
MOV E, M	5E	Register indirect	7	1	None	$[A] \leftarrow [M_{H-L}]$
MOV H, M	66	Register indirect	7	1	None	$[A] \leftarrow [M_{H-L}]$
MOV L, M	6E	Register	7	1	None	$[A] \leftarrow [M_{H-L}]$
MOV M, A	77	Register indirect	7	1	None	$[M_{H-L}] \leftarrow [A]$
MOV M, B	70	Register	7	1	None	$[M_{H-L}] \leftarrow [B]$

		Indirect				
MOV M, C	71	Register indirect	7	1	None	$[M_{H-L}] \leftarrow [C]$
MOV M, D	72	Register indirect	7	1	None	$[M_{H-L}] \leftarrow [D]$
MOV M, E	73	Register indirect	7	1	None	$[M_{H-L}] \leftarrow [E]$
MOV M, H	74	Register indirect	7	1	None	$[M_{H-L}] \leftarrow [H]$
MOV M, L	75	Register indirect	7	1	None	$[M_{H-L}] \leftarrow [L]$
MVI A, data	3E	Immediate	7	2	None	$[A] \leftarrow data$
MVI B, data	06	Immediate	7	2	None	$[B] \leftarrow data$
MVI C, data	0E	Immediate	7	2	None	$[C] \leftarrow data$
MVI D, data	16	Immediate	7	2	None	$[D] \leftarrow data$
MVI E, data	1E	Immediate	7	2	None	$[E] \leftarrow data$
MVI H, data	26	Immediate	7	2	None	$[H] \leftarrow data$
MVI L, data	2E	Immediate	7	2	None	$[L] \leftarrow data$
MVI M, data	36	Immediate	10	2	None	$[M_{H-L}] \leftarrow data$

(a) Direct Data Transfer Instructions

There is basically two direct data transfer instructions for SAP-III computers which are the same used in SAP-II computers. These are:

(i) LDA address

This is mnemonic for **Loads the accumulator direct**. It transfers the content stored in the addressed memory location (given by address) to accumulator.

$$[A] \leftarrow [M_{address}]$$

No flag is affected in this instruction. It is three byte instruction.

For example if 2AH data is stored in memory location 2500H before the execution of LDA 2500H instruction, then after the execution of this instruction, the data 2AH will be transferred to accumulator.

i.e. $[A] \leftarrow 2A$

(ii) STA address

This is mnemonic for **Stores the accumulator direct**. It transfers the content stored in the accumulator to addressed memory location (given by address).

$$[M_{address}] \leftarrow [A]$$

No flag is affected in this instruction. It is also three byte instruction.

For example if 16H data is stored in the accumulator before the execution of STA 2100H instruction, then after the execution of this instruction, the data 16H will be transferred to the addressed memory location.

i.e. $[M_{2100}] \leftarrow 16$

(b) Register Data Transfer Instructions

These instructions transfer 8-bit data stored in one register to other register. The registers in SAP-III computer are A, B, C, D, E, H and L so the data stored in one of these registers may be transferred to other register. The format for this data transfer instruction is given as:

MOV **reg1, reg2** (**Move Register)**

where *reg1 = A, B, C, D, E, H or L*
reg2 = A, B, C, D, E, H or L

This instruction copies (transfers) the content stored in reg2 to reg1.

$$[reg1] \leftarrow [reg2]$$

Following are the possible combinations of MOV instruction.

MOV A, A	*MOV B, A*
MOV A, B	*MOV B, B*
MOV A, C	*MOV B, C*
MOV A, D	*MOV B, D*
MOV A, E	*MOV B, E*
MOV A, H	*MOV B, H*
MOV A, L	*MOV B, L*
MOV C, A	*MOV D, A*
MOV C, B	*MOV D, B*
MOV C, C	*MOV D, C*
MOV C, D	*MOV D, D*
MOV C, E	*MOV D, E*
MOV C, H	*MOV D, H*
MOV C, L	*MOV D, L*
MOV E, A	*MOV H, A*
MOV E, B	*MOV H, B*
MOV E, C	*MOV H, C*
MOV E, D	*MOV H, D*
MOV E, E	*MOV H, E*
MOV E, H	*MOV H, H*
MOV E, L	*MOV H, L*
MOV L, A	
MOV L, B	
MOV L, C	
MOV L, D	
MOV L, E	
MOV L, H	
MOV L, L	

These instructions are of one byte instruction and no flag is affected in these instructions.

For example if L = 23 H before the execution of instruction *MOV* C, L then after the execution of this instruction 23 H data in register L will be transferred to register C.

i.e. $[L] \leftarrow 23H$

(c) Register Indirect Data Transfer Instructions

In SAP-III computers, there are some register indirect data transfer instructions in which H-L register pair acts like 'data pointer'. The data pointer is represented by M which denotes the memory location whose address is given in the H-L register pair. It is basically represented by M_{HL}.

For example H L = 2500 H then $M \Rightarrow M_{HL}$ represents the memory location whose address is 2500 H. It indicates the memory location M_{2500}.

The instructions related to register immediate data transfer are given as:

(i) *MOV* ***reg, M*** **(Moves register from memory)**

where *reg = A, B, C, D, E, H or L*

This instruction is indirect read instruction. It moves / copies the data stored in memory location whose address is given in H-L register pair, to the given register.

i.e.

$$[reg] \leftarrow [M_{HL}]$$

The possible combinations of this instruction are:

MOV A, M
MOV B, M
MOV C, M
MOV D, M
MOV E, M
MOV H, M
MOV L, M

For example consider 12H data is stored in the memory location whose address is given in H-L register pair (i.e. H-L = 2500H). After the execution of the instruction *MOV B, M* the data 12H will be stored in B register.

$$[B] \leftarrow 12$$

These are one byte instruction and no flag is affected in these instructions.

(ii) *MOV* ***M, reg*** **(Moves memory from register)**

where *reg = A, B, C, D, E, H or L*

This instruction moves / copies the data in the given register to the memory location addressed by H-L register pair.

i.e.

$$[M_{HL}] \leftarrow [reg]$$

It performs reverse work of *MOV reg, M* instruction. The possible combinations of this instruction are:

MOV M, A
MOV M, B
MOV M, C
MOV M, D
MOV M, E
MOV M, H

MOV M, L

For example let D = 4E H H = 23 H L = 00 H

Then after execution of the instruction *MOV M, D* will produce the result:

$$[M_{2300}] \leftarrow 4E$$

These are also one byte instructions and no flag is affected in these instructions.

(d) Immediate Data Transfer Instructions

The format for immediate data transfer instructions is

*MVI **reg, data*** **(Move Immediate)**

where *reg = A, B, C, D, E, H, L orM*

This instruction transfers the given data to the register (reg).

$$[reg] \leftarrow data$$

The possible combinations of move immediate instruction are given below:

MVI A, data
MVI B, data
MVI C, data
MVI D, data
MVI E, data
MVI H, data
MVI L, data
MVI M, data

For example

MVI E, 1AH

means the given data 1AH will be copied into the register E.

$$[E] \leftarrow 1A$$

Also *MVI M, 2BH*

means the given data 2BH will be copied into the memory location whose address is given in H-L register pair. If H = 21 H L = 00 H then

$$[M_{2100}] \leftarrow 2B$$

All these instructions are two byte instructions and no flag is affected.

4.2.2 Arithmetic Group of Instructions

This group of instructions (given in table 4.2) includes the instructions for addition and subtraction. Some of these instructions are the same as discussed in SAP-II computer.

Table 4.2

Arithmetic Group

Instruction	Op. Code	Addressing modes	No of T-States	No of bytes of instr.	Flags affected	Operation
ADD A	87	Register	4	1	All	$[A] \leftarrow [A]+[A]$
ADD B	80	Register	4	1	All	$[A] \leftarrow [A]+[B]$
ADD C	81	Register	4	1	All	$[A] \leftarrow [A]+[C]$
ADD D	82	Register	4	1	All	$[A] \leftarrow [A]+[D]$
ADD E	83	Register	4	1	All	$[A] \leftarrow [A]+[E]$

ADD H	84	Register	4	1	All	$[A] \leftarrow [A] + [H]$
ADD L	85	Register	4	1	All	$[A] \leftarrow [A] + [L]$
ADD M	86	Register indirect	7	1	All	$[A] \leftarrow [A] + [M_{H-L}]$
ADI data	C6	Immediate	7	2	All	$[A] \leftarrow [A] + data$
ADC A	8F	Register	4	1	All	$[A] \leftarrow [A] + [A] + CY$
ADC B	88	Register	4	1	All	$[A] \leftarrow [A] + [B] + CY$
ADC C	89	Register	4	1	All	$[A] \leftarrow [A] + [C] + CY$
ADC D	8A	Register	4	1	All	$[A] \leftarrow [A] + [D] + CY$
ADC E	8B	Register	4	1	All	$[A] \leftarrow [A] + [E] + CY$
ADC H	8C	Register	4	1	All	$[A] \leftarrow [A] + [H] + CY$
ADC L	8D	Register	4	1	All	$[A] \leftarrow [A] + [L] + CY$
ADC M	8E	Register indirect	7	1	All	$[A] \leftarrow [A] + [M_{H-L}] + CY$
ACI data	CE	Immediate	7	2	All	$[A] \leftarrow [A] + data + CY$
DAD B	09	Register	10	1	CY	$BC \leftarrow BC + BC$
DAD D	19	Register	10	1	CY	$DE \leftarrow DE + DE$
DAD H	29	Register	10	1	CY	$HL \leftarrow HL + HL$
SUB A	97	Register	4	1	All	$[A] \leftarrow [A] - [A]$
SUB B	90	Register	4	1	All	$[A] \leftarrow [A] - [B]$
SUB C	91	Register	4	1	All	$[A] \leftarrow [A] - [C]$
SUB D	92	Register	4	1	All	$[A] \leftarrow [A] - [D]$
SUB E	93	Register	4	1	All	$[A] \leftarrow [A] - [E]$
SUB H	94	Register	4	1	All	$[A] \leftarrow [A] - [H]$
SUB L	95	Register	4	1	All	$[A] \leftarrow [A] - [L]$
SUB M	96	Register indirect	7	1	All	$[A] \leftarrow [A] - [M_{H-L}]$
SUI data	D6	Immediate	7	2	All	$[A] \leftarrow [A] - data$
SBB A	9F	Register	4	1	All	$[A] \leftarrow [A] - [A] - CY$
SBB B	98	Register	4	1	All	$[A] \leftarrow [A] - [B] - CY$
SBB C	99	Register	4	1	All	$[A] \leftarrow [A] - [C] - CY$
SBB D	9A	Register	4	1	All	$[A] \leftarrow [A] - [D] - CY$
SBB E	9B	Register	4	1	All	$[A] \leftarrow [A] - [E] - CY$
SBB H	9C	Register	4	1	All	$[A] \leftarrow [A] - [H] - CY$
SBB L	9D	Register	4	1	All	$[A] \leftarrow [A] - [L] - CY$
SBB M	9E	Register indirect	7	1	All	$[A] \leftarrow [A] - [M_{H-L}] - CY$
SBI data	DE	Immediate	7	2	All	$[A] \leftarrow [A] - data - CY$

INR A	3C	Register	4	1	All but not CY	$[A] \leftarrow [A]+1$
INR B	04	Register	4	1	All but not CY	$[B] \leftarrow [B]+1$
INR C	0C	Register	4	1	All but not CY	$[C] \leftarrow [C]+1$
INR D	14	Register	4	1	All but not CY	$[D] \leftarrow [D]+1$
INR E	1C	Register	4	1	All but not CY	$[E] \leftarrow [E]+1$
INR H	24	Register	4	1	All but not CY	$[H] \leftarrow [H]+1$
INR L	2C	Register	4	1	All but not CY	$[L] \leftarrow [L]+1$
INR M	34	Register indirect	10	1	All but not CY	$[M_{H-L}] \leftarrow [M_{H-L}]+1$
INX B	03	Register	6	1	None	$BC \leftarrow BC+1$
INX D	13	Register	6	1	None	$DE \leftarrow DE+1$
INX H	23	Register	6	1	None	$HL \leftarrow HL+1$
DCR A	3D	Register	4	1	All but not CY	$[A] \leftarrow [A]-1$
DCR B	05	Register	4	1	All but not CY	$[B] \leftarrow [B]-1$
DCR C	0D	Register	4	1	All but not CY	$[C] \leftarrow [C]-1$
DCR D	15	Register	4	1	All but not CY	$[D] \leftarrow [D]-1$
DCR E	1D	Register	4	1	All but not CY	$[E] \leftarrow [E]-1$
DCR H	25	Register	4	1	All but not CY	$[H] \leftarrow [H]-1$
DCR L	2D	Register	4	1	All but not CY	$[L] \leftarrow [L]-1$
DCR M	35	Register indirect	7	1	All but not CY	$[M_{H-L}] \leftarrow [M_{H-L}]-1$
DCX B	0B	Register	6	1	None	$BC \leftarrow BC-1$
DCX D	1B	Register	6	1	None	$DE \leftarrow DE-1$
DCX H	2B	Register	6	1	None	$HL \leftarrow HL-1$
RLC	07	---	4	1	CY	Rotate left through carry
RAL	17	---	4	1	CY	Rotate all left
RRC	0F	---	4	1	CY	Rotate right through carry
RAR	1F	---	4	1	CY	Rotate all right

(a) Add instructions

Following are the add instructions:

(i) *ADD reg* **(Add register)**

where *reg = A, B, C, D, E, H or L.*

It adds the content stored in given register with the accumulator. The result of this addition is stored in accumulator.

$$[A] \leftarrow [A] + [reg]$$

All the flags are affected with this '*ADD reg'* instruction.

The possible combinations of this instruction are as given below:

ADD A
ADD B
ADD C
ADD D
ADD E
ADD H
ADD L

Suppose before the execution of the instruction *ADD E*

A = 10101111
E = 10110101
$CY = 0$, $S = 1$, $Z = 0$ and $P = 1$

then after the execution of the instruction *ADD E* we get the following result:

```
A =        1  0  1  0  1  1  1  1
E =        1  0  1  1  0  1  0  1
           ------------------------
A =   1    0  1  1  0  0  1  0  0
      CY
```

The flags will be affected as;

$CY = 1$, $S = 0$, $Z = 0$ and $P = 0$

$CY = 1$ Since there is a final carry.
$S = 0$ Since Acc content is positive as MSB is zero.
$Z = 0$ Since Acc content is non zero.
$P = 0$ Since Acc content has odd parity.

(ii) *ADD M* **(Add Memory)**

This instruction is one byte instruction and adds the content of memory location whose address is given in H-L register pair with the accumulator and the answer is stored in accumulator.

$$[A] \leftarrow [A] + [M_{HL}]$$

In this instruction too all flags are affected. This instruction is similar to *ADD reg*. For example let A = 40 H H = 21 H L = 00 H

and $M_{2100} = 3AH$

Then after execution of the instruction *ADD M* will produce the result:

A = 7A H

All flag will, however, be affected as per the instruction.

(iii) *ADI data* **(Adds immediately the data)**

It immediately adds the given data with the accumulator and the answer will be stored in Accumulator.

$$[A] \leftarrow [A] + data$$

This is a two-byte instruction. All flags will be affected with this instruction similar to *ADD reg* instruction.

(iv) *ADC* instructions

The format for ADC instruction is

ADC reg **(Add with carry)**

where *reg = A, B, C, D, E, H or L.*

It adds the content stored in given register and content of CY flag with the content of accumulator. The result of this addition is stored in accumulator.

$$[A] \leftarrow [A] + [reg] + [CY]$$

All the flags are affected with this '*ADC reg'* instruction.

The possible combinations of this instruction are as given below:

ADC A
ADC B
ADC C
ADC D
ADC E
ADC H
ADC L

Suppose before the execution of the instruction *ADC D*

A = 10101001
D = 10111101
$CY = 1, \quad S = 1, \quad Z = 0$ and $P = 1$

then after the execution of the instruction *ADD D* we get the following result:

```
CY =                        1
 A =        1  0  1  0  1  0  0  1
 D =        1  0  1  1  1  1  0  1
            ------------------------
 A =   1    0  1  1  0  0  1  1  1
       CY
```

All flags will be affected.

(v) *ADC M* **(Add Memory with Carry)**

This instruction is one byte instruction and adds the content of memory location whose address is given in H-L register pair to the accumulator with carry and the answer is stored in accumulator.

$$[A] \leftarrow [A] + [M_{HL}] + \text{CY}$$

In this instruction too all flags are affected. This instruction is similar to *ADD reg*. For example let A = 50 H CY = 1 H = 25 H L = 00 H

and $M_{2500} = 3BH$

Then after execution of the instruction *ADC M* will produce the result:

A = 8C H

All flag will, however, be affected as per the result.

(vi) *ACI data* **(Adds immediately the data with Carry)**

It immediately adds the given data to the accumulator with carry and the answer will be stored in Accumulator.

$$[A] \leftarrow [A] + data + \text{CY}$$

This is a two-byte instruction. All flags will be affected with this instruction.

(vii) DAD Instruction

The format for this instruction is

DAD rp **(Double Add)**

where rp stands for register pair. It may be B, D or H.
rp as B represents B-C register pair,
rp as D represents D-E register pair,
rp as H represents H-L register pair.

This instruction adds the contents of given register pair with the contents of H-L register pair. The answer will be stored in H-L register pair.

$$[HL] \leftarrow [HL] + [rp]$$

This instruction has the following combinations:

DAD B	$[HL] \leftarrow [HL] + [BC]$
DAD D	$[HL] \leftarrow [HL] + [DE]$
DAD H	$[HL] \leftarrow [HL] + [HL]$

For example before the execution of the instruction DAD D, we have the following data in different registers.

DE = 2AB6 H
HL = 012 7 H

After the execution of this instruction we HL = 2BDD H

Only carry flag will be affected in this instruction.

(b) Subtract Instructions

Following are the subtract instructions:

(i) *SUB reg* **(Subtract Register)**

where *reg = A, B, C, D, E, H or L.*

It subtracts the contents stored in given register from the accumulator. The result of this subtraction is stored in accumulator.

$$[A] \leftarrow [A] - [reg]$$

All the flags are affected with this '*SUB reg*' instruction.

The possible combinations of this instruction are as given below:

SUB A
SUB B
SUB C
SUB D
SUB E
SUB H
SUB L

Suppose before the execution of the instruction *SUB D*

A = 10101111

D = 10110101

$CY = 0$, $S = 1$, $Z = 0$ and $P = 1$

then after the execution of the instruction $SUB\,D$ we get the following result:

A = 1 0 1 0 1 1 1 1

D = 1 0 1 1 0 1 0 1

A = 1 1 1 1 1 1 0 1 0

CY

All flags will be affected as per the accumulator contents.

(ii) *SUB M* **(Subtract Memory)**

This instruction is one byte instruction and subtracts the contents of memory location whose address is given in H-L register pair from the accumulator and the answer is stored in accumulator.

$$[A] \leftarrow [A] - [M_{HL}]$$

In this instruction too all flags are affected.

For example let A = 56 H H = 22 H L = 01 H

and $M_{2201} = 2CH$

Then after execution of the instruction *SUB M* will produce the result:

A = 2A H

All flag will however be affected as per the instruction.

(iii) *SUI data* **(Subtracts immediately the data)**

It immediately subtracts the given data with the accumulator contents and the answer will be stored in Accumulator.

$$[A] \leftarrow [A] - data$$

This is a two-byte instruction. All flags will be affected with this instruction.

(iv) *SBB reg* **(Subtract with Borrow)**

where *reg = A, B, C, D, E, H or L.*

This instruction subtracts the contents stored in given register from the accumulator with borrow. The answer will be stored in accumulator.

$$[A] \leftarrow [A] - [reg] - [CY]$$

All the flags are affected in the operation of this instruction.

The possible combinations of this instruction are as given below:

SBB A
SBB B
SBB C
SBB D
SBB E
SBB H
SBB L

Suppose before the execution of the instruction $SBB\,H$

A = 10101111

H = 10110101

$CY = 0$, $S = 1$, $Z = 0$ and $P = 1$

then after the execution of the instruction $SUB\,H$ we get the following result:

A = 1 0 1 0 1 1 1 1

H = 1 0 1 1 0 1 0 1
CY = 1

A = 1 1 1 1 1 1 0 0 1
CY

All flags will be affected as per the accumulator contents.

(v) *SBB M* **(Subtract memory with Borrow)**

Similar to *SBB reg*, this instruction subtracts the content already stored in memory location addressed by H-L register pair with borrow bit (carry flag) from the accumulator contents and the answer is stored in accumulator.

$$[A] \leftarrow [A] - [M_{HL}] - [CY]$$

All flags will be affected as per the accumulator contents.

(vi) *SBI data* **(Subtract immediate with Borrow)**

SBI data instruction subtracts the given data with borrow bit (carry flag) from the accumulator contents and it stores the answer in accumulator.

$$[A] \leftarrow [A] - data - [CY]$$

All flags will be affected as per the accumulator contents.

(c) Increment Instructions

The increment instructions used in SAP-III computer are given below:

(i) *INR reg* **(Increment Register)**

where *reg = A, B, C, D, E, H or L.*

It increments the contents of given register by one and answer is stored in the given register.

$$[reg] \leftarrow [reg] + 1$$

The possible combinations of this instruction are:

INR A
INR B
INR C
INR D
INR E
INR H
INR L

These instructions do not affect the CY flag but affect all other flags.

For example if Z = 0, S = 1, and CY = 0 and contents of register is

C = 1 1 1 1 1 1 1 1 , then after the execution of the instruction INR C we have:

C = 0 0 0 0 0 0 0 0

Z = 1, S = 0, and CY = 0

(ii) *INR M* **(Increment Memory)**

It increments the contents of memory location addressed by H-L register pair by 1 and stores the answer in the addressed memory location.

$$[M_{HL}] \leftarrow [M_{HL}] + 1$$

Similar to *INR reg*, all flags except carry flag will be affected after the execution of this instruction.

(iii) *INX rp* **(Increment register pair)**

where *rp* is the register pair as B-C register pair, D-E register pair and H-L register pair. This instruction increments the contents given in register pair by one and the result is stored in the given register pair.

$$[rp] \leftarrow [rp] + 1$$

The possible combinations of this instruction are:

INX B
INX D
INX H

No flag is affected with the execution of this instruction.

For example if H = FF H and L = FF H CY = 1, Z = 0

After the execution of the instruction INX H, we have the contents in H-L pair as: H = 25 H and L = 01 H ; the contents of the flags will not be affected. So the carry and zero flag will have the same value as having before the execution of this instruction i.e. CY = 1, Z = 0

(d) Decrement Instructions

The decrement instructions used in SAP-III computer are given below:

(i) DC*R reg* (Decrement Register)

where *reg = A, B, C, D, E, H or L.*

It decrements the contents of given register by one and answer is stored in the given register.

$$[reg] \leftarrow [reg] - 1$$

The possible combinations of this instruction are:

DCR A
DCR B
DCR C
DCR D
DCR E
DCR H
DCR L

These instructions do not affect the CY flag but affect all other flags.

For example if Z = 0, S = 1, and CY = 0 and contents of D register is

D = 0 0 0 0 0 0 0 0 , then after the execution of the instruction DCR D we have:

C = 1 1 1 1 1 1 1 1

Z = 0, S = 1, and CY = 0

(ii) *DCR M* (Decrement Memory)

It decrements the contents of memory location addressed by H-L register pair by 1 and stores the answer in the addressed memory location.

$$[M_{HL}] \leftarrow [M_{HL}] - 1$$

Similar to *DCR reg*, all flags except carry flag will be affected after the execution of this instruction.

(iii) *DCX rp* (Increment register pair)

where *rp* is the register pair as B-C register pair, D-E register pair and H-L register pair. This instruction decrements the contents given in register pair by one and the result is stored in the given register pair.

$$[rp] \leftarrow [rp] - 1$$

The possible combinations of this instruction are:

DCX B
DCX D
DCX H

No flag will be affected with the execution of this instruction like INX rp.

For example if D = 23 H and E = 00 H

After the execution of the instruction DCX D, we have the contents in D-E pair as: D = 22 H and E = FF H

(e) Rotate Instructions

In rotate instructions, the accumulator contents are shifted either left or right. In some instructions shifting may be through CY flag or without CY flag. Following are the rotate instructions.

(i) *RLC* **(Rotate Accumulator Left)**

In this instruction, the bits of the accumulator contents are shifted or rotated left. The LSB of the accumulator is changed as MSB (before the execution). The CY flag is modified as MSB (before the execution).

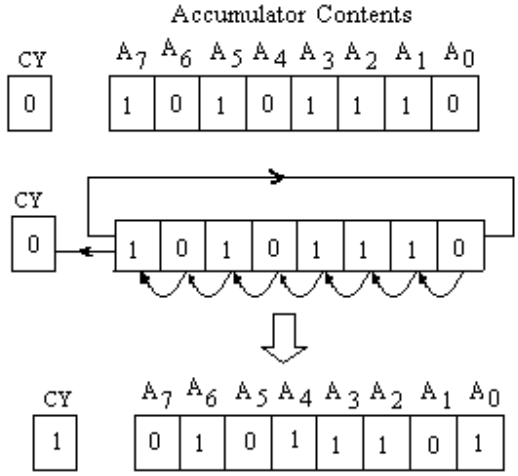


Fig. 4.4

For example A = AE H and CY = 0, before the execution of the instruction RLC. After the execution of the instruction MSB is saved in CY flag and also in the LSB of the accumulator. The other bits are shifted left as shown in figure 4.4.

i.e. $[A_{n+1}] \leftarrow [A_n]$

$[A_0] \leftarrow [A_7]$ also $[CY] \leftarrow [A_7]$

Only carry flag CY will be affected in this instruction and all other flags will be unaffected.

(ii) *RAL* **(Rotate Accumulator Left Through Carry)**

In this instruction, the bits of the accumulator contents will be shifted / rotated left through carry. The content of carry flag CY will be stored in LSB of the accumulator and MSB of the accumulator will be stored in CY flag. All other bits of the accumulator will be shifted to the left.

For example if A = 6A H and CY = 1 before the execution of RAL instruction, then after the execution of this instruction the accumulator contents will be shifted as shown in figure 4.5.

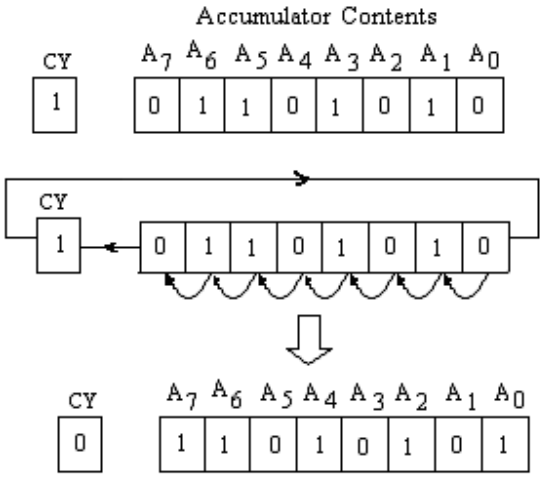


Fig. 4.5

In this instruction too only carry flag will be affected.

(iii) *RRC* **(Rotate Accumulator Right)**

In this instruction, all the bits of accumulator are shifted or rotated right. The MSB of the accumulator is changed as LSB (before the execution). The CY flag is modified as MSB (before the execution).

For example $A = 93$ H and $CY = 0$, before the execution of the instruction RRC. After the execution of this instruction LSB is saved in CY flag and also in the MSB of the accumulator. The other bits are shifted left as shown in figure 4.6.

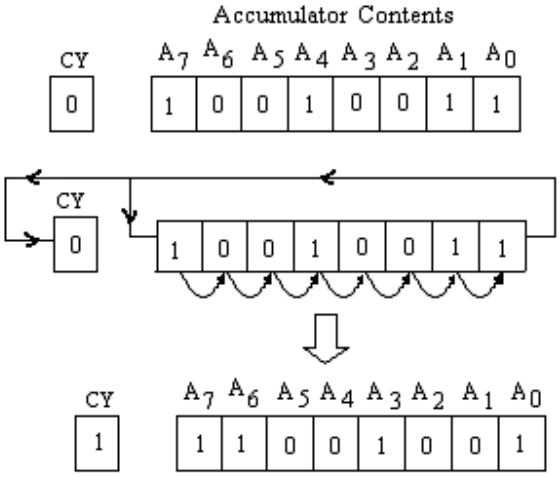


Fig. 4.6

In this instruction only carry flag CY will be affected and all other flags will be unaffected.

(iv)*RAR* **(Rotate Accumulator Right Through Carry)**

In this rotate instruction, all the bits of the accumulator contents will be shifted / rotated right through carry. The content of carry flag CY will be stored in MSB of the accumulator and LSB of the accumulator will be stored in CY flag; and all other bits of the accumulator will be shifted to the right.

For example if A = 76 H and CY = 1 before the execution of RAR instruction, then after the execution of this instruction the accumulator contents will be shifted as shown in figure 4.7.

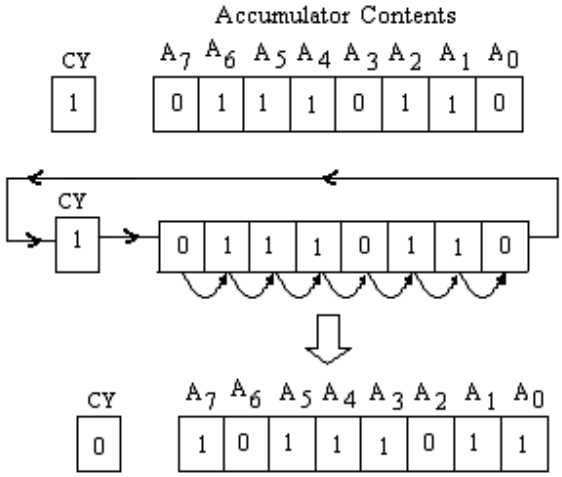


Fig. 4.7

In this instruction too only carry flag will be affected.

4.2.3 Logic Transfer Group

The logic group of instructions operates on registers, memory and conditional flags. This group of instructions contains ANDing, ORing, XORing, complementing and comparing of data (table 4.3).

Table 4.3

Logic Transfer Group

Instruction	Op. Code	Addressing modes	No of T-states	No of bytes of instr.	Flags affected	Operation
ANA A	A7	Register	4	1	All (CY=0)	$[A] \leftarrow [A].AND.[A]$
ANA B	A0	Register	4	1	All (CY=0)	$[A] \leftarrow [A].AND.[B]$
ANA C	A1	Register	4	1	All (CY=0)	$[A] \leftarrow [A].AND.[C]$
ANA D	A2	Register	4	1	All (CY=0)	$[A] \leftarrow [A].AND.[D]$
ANA E	A3	Register	4	1	All (CY=0)	$[A] \leftarrow [A].AND.[E]$
ANA H	A4	Register	4	1	All (CY=0)	$[A] \leftarrow [A].AND.[H]$
ANA L	A5	Register	4	1	All (CY=0)	$[A] \leftarrow [A].AND.[L]$
ANA M	A6	Register Indirect	7	1	All (CY=0)	$[A] \leftarrow [A].AND.[M_{H-L}]$
ANI data	E6	Immediate	7	2	All (CY=0)	$[A] \leftarrow [A].AND.data$

ORA A	B7	Register	4	1	All (CY=0)	$[A] \leftarrow [A].OR.[A]$
ORA B	B0	Register	4	1	All (CY=0)	$[A] \leftarrow [A].OR.[B]$
ORA C	B1	Register	4	1	All (CY=0)	$[A] \leftarrow [A].OR.[C]$
ORA D	B2	Register	4	1	All (CY=0)	$[A] \leftarrow [A].OR.[D]$
ORA E	B3	Register	4	1	All (CY=0)	$[A] \leftarrow [A].OR.[E]$
ORA H	B4	Register	4	1	All (CY=0)	$[A] \leftarrow [A].OR.[H]$
ORA L	B5	Register	4	1	All	$[A] \leftarrow [A].OR.[L]$
ORA M	B6	Register Indirect	7	1	All (CY=0)	$[A] \leftarrow [A].OR.[M_{H-L}]$
ORI data	F6	Immediate	7	2	All (CY=0)	$[A] \leftarrow [A].OR.data$
XRA A	AF	Register	4	1	All (CY=0)	$[A] \leftarrow [A] \oplus [A]$
XRA B	A8	Register	4	1	All (CY=0)	$[A] \leftarrow [A] \oplus [B]$
XRA C	A9	Register	4	1	All (CY=0)	$[A] \leftarrow [A] \oplus [C]$
XRA D	AA	Register	4	1	All (CY=0)	$[A] \leftarrow [A] \oplus [D]$
XRA E	AB	Register	4	1	All (CY=0)	$[A] \leftarrow [A] \oplus [E]$
XRA H	AC	Register	4	1	All (CY=0)	$[A] \leftarrow [A] \oplus .[H]$
XRA L	AD	Register	4	1	All (CY=0)	$[A] \leftarrow [A] \oplus [L]$
XRA M	AE	Register Indirect	7	1	All (CY=0)	$[A] \leftarrow [A] \oplus [M_{H-L}]$
XRI data	EE	Immediate	7	2	All (CY=0)	$[A] \leftarrow [A] \oplus data$
CMP A	BF	Register	4	1	All	
CMP B	B8	Register	4	1	All	
CMP C	B9	Register	4	1	All	
CMP D	BA	Register	4	1	All	
CMP E	BB	Register	4	1	All	
CMP H	BC	Register	4	1	All	
CMP L	BD	Register	4	1	All	
CMP M	BE	Register Indirect	7	1	All	
CPI data	FE	Immediate	7	2	All	

CMA	2F	---	4	1	None	$[A] \leftarrow [\overline{A}]$
CMC	3F	---	4	1	CY	$CY \leftarrow \overline{CY}$
STC	37	---	4	1	CY	$CY \leftarrow 1$

The details of these instructions are given below:

(a) Logical Instructions

(i) ***ANA reg*** **(AND register)**

where *reg = A, B, C, D, E, H or L.*

In this instruction each bit of the given register contents are ANDed with each bit of the accumulator contents (bit by bit). The result is saved in the accumulator. It does not affect the contents of the given register.

$$[A] \leftarrow [A].AND.[reg]$$

The possible combinations of this instruction are:

AND A
ANA B
ANA C
ANA D
ANA E
ANA H
ANA L

The *ANA reg* instruction clears (resets) the CY flag and all other flags are modified according to the data conditions of the result. This instruction is one byte instruction.

Let A = 73 H C = C3 H CY = 1

before the execution of the instruction *ANA C*. Then after the instruction is executed we get:

A = 0 1 1 1 0 0 1 1 CY = 1

C = 1 1 0 0 0 0 1 1

A = 0 1 0 0 0 0 1 1 CY = 0

(ii) ***ANA M*** **(AND Memory)**

In this instruction each bit of the data stored in the memory location addressed by H-L register pair are ANDed with each bit of the accumulator contents (bit by bit). The result is stored in the accumulator like *ANA reg*.

$$[A] \leftarrow [A].AND.[M_{H-L}]$$

Similar to ANA reg instruction data of the memory location addressed by H-L register pair will not be affected; the carry flag will, however, be reset after the execution of this instruction and other flags will be affected as per the data in the accumulator.

Let A = 19 H H = 25 H L = 00 H

$M_{2500} = 37$ H and CY = 1

before the execution of the instruction *ANA M*. Then after the instruction is executed we get:

A = 0 0 0 1 1 0 0 1 CY = 1

M_{2500} = 0 0 1 1 0 1 1 1

A = 0 0 0 1 0 0 0 1 CY = 0

(iii) *ANI data* **(AND Immediate)**

In this instruction each bit of the given data is immediately ANDed with each bit of the accumulator contents (bit by bit). The result is stored in the accumulator. The difference between *ANA reg* and *ANI data* instruction is that in *ANA reg* the data given in the register where as in the *ANI data* instruction, the data is given with the instruction itself.

$$[A] \leftarrow [A].AND.data$$

The carry flag will be reset after the execution of this instruction and other flags will be affected as per the result.

For example, if A = AB H and CY = 1 before the execution of ANI 06 H, then after the execution of this instruction we have:

A = 1 0 1 0 1 0 1 1 CY = 1

data = 0 0 0 0 0 1 1 0

A = 0 0 0 0 0 0 1 0 CY = 0

(iv) *ORA reg* **(OR register)**

where *reg = A, B, C, D, E, H or L.*

In this instruction each bit of the given register contents are ORed with each bit of the accumulator contents (bit by bit). The result is saved in the accumulator. It does not affect the contents of the given register.

$$[A] \leftarrow [A].OR.[reg]$$

The possible combinations of this instruction are:

ORA A
ORA B
ORA C
ORA D
ORA E
ORA H
ORA L

The *ORA reg* instruction clears (resets) the CY flag and all other flags are modified according to the data conditions of the result. This instruction is one byte instruction.

Let A = 73 H C = C3 H CY = 1

before the execution of the instruction *ORA B*. Then after the instruction is executed we get:

A = 0 1 1 1 0 0 1 1 CY = 1

B = 1 1 0 0 0 0 1 1

A = 1 1 1 1 0 0 1 1 CY = 0

(v) *ORA M* **(OR Memory)**

In this instruction each bit of the data stored in the memory location addressed by H-L register pair are ORed with each bit of the accumulator contents (bit by bit). The result is stored in the accumulator.

$$[A] \leftarrow [A].OR.[M_{H-L}]$$

Similar to *ORA reg* instruction data of the memory location addressed by H-L register pair will not be affected; the carry flag will however be reset after the execution of this instruction and other flags will be affected as per the data in the accumulator.

Let A = 19 H H = 21 H L = 00 H

$M_{2100} = 37$ H and CY = 1

before the execution of the instruction *ORA M*. Then after the instruction is executed we get:

A = 0 0 0 1 1 0 0 1 CY = 1

M_{2100} = 0 0 1 1 0 1 1 1

A = 0 0 1 1 1 1 1 1 CY = 0

(vi) *ORI data* **(OR Immediate)**

In this instruction each bit of the given data is immediately ORed with each bit of the accumulator contents (bit by bit). The result is stored in the accumulator.

$$[A] \leftarrow [A].OR.data$$

The carry flag will be reset after the execution of this instruction and other flags will be affected as per the result.

For example, if A = AB H and CY = 1 before the execution of ANI 16 H, then after the execution of this instruction we have:

A = 1 0 1 0 1 0 1 1 CY = 1

data = 0 0 0 1 0 1 1 0

A = 1 0 1 1 1 1 1 1 CY = 0

(vii) *XRA reg* **(Exclusive OR register)**

where *reg = A, B, C, D, E, H or L.*

In this instruction each bit of the given register contents are *XORed* with each bit of the accumulator contents (bit by bit). The result is saved in the accumulator. It does not affect the contents of the given register.

$$[A] \leftarrow [A].XOR.[reg]$$

The possible combinations of this instruction are:

XRA A
XRA B
XRA C
XRA D
XRA E
XRA H
XRA L

The *XRA reg* instruction clears (resets) the CY flag and all other flags are modified according to the data conditions of the result. This instruction is one byte instruction.

Let A = 73 H D = C3 H CY = 1

before the execution of the instruction *XRA D*. Then after the instruction is executed we get:

A = 0 1 1 1 0 0 1 1 CY = 1

D = 1 1 0 0 0 0 1 1

A = 1 0 1 1 0 0 0 0 CY = 0

(viii) *XRA M* **(Exclusive OR Memory)**

In this instruction each bit of the memory location addressed by H-L register pair are XORed with each bit of the accumulator contents (bit by bit). The result is stored in the accumulator.

$$[A] \leftarrow [A].XOR.[M_{H-L}]$$

The carry flag will be reset after the execution of this instruction and other flags will be affected as per the data in the accumulator.

Let A = 19 H H = 22 H L = 00 H

$M_{2200} = 37$ H and CY = 1

before the execution of the instruction *ORA M*. Then after the instruction is executed we get:

A = 0 0 0 1 1 0 0 1 CY = 1

M_{2200} = 0 0 1 1 0 1 1 1

A = 0 0 1 0 1 1 1 0 CY = 0

(ix) *XRI data* **(Exclusive OR Immediate)**

In this instruction each bit of the given data is immediately XORed with each bit of the accumulator contents (bit by bit). The result is stored in the accumulator.

$$[A] \leftarrow [A].XOR.data$$

The carry flag will be reset after the execution of this instruction and other flags will be affected as per the result.

For example, if A = AB H and CY = 1 before the execution of ANI 12 H, then after the execution of this instruction we have:

A = 1 0 1 0 1 0 1 1 CY = 1

data = 0 0 0 1 0 0 1 0

A = 1 0 1 1 1 0 0 1 CY = 0

(x) *CMA* **(Complement Accumulator)**

This is one byte implied addressing instruction as no operand is required with the instruction. The execution of this instruction inverts each bit of the accumulator contents and the result is saved in the accumulator. Basically it produces 1's complement of the accumulator contents. No flag is affected with this instruction.

$$[A] \leftarrow [\overline{A}]$$

For example, if A = 0B H before the execution of *CMA* instruction, then after the execution of this instruction we have:

A = 0 0 0 0 1 0 1 1

= 1 1 1 1 0 1 0 0

A = 1 1 1 1 0 1 0 0 (F4)

(b) Compare Instructions

(i) *CMP reg* (Compare Register)

The contents of the given register are compared with the accumulator contents. In fact the contents of the register are subtracted from the contents of accumulator and the accumulator contents remain unchanged. However, as a result of the subtraction the flags are modified as per the result.

The zero flag Z is set if $[A] = [reg]$ otherwise reset. Similarly, carry flag CY is set if $[A] < [reg]$ otherwise reset.

The possible combinations of *CMP reg* are given below:

CMP A
CMP B
CMP C
CMP D
CMP E
CMP H
CMP L

To illustrate the above instruction let us consider the following assembly language program.

Label	**Mnemonics**	**Operand**
	MVI B,	00 H
	MVI A,	09 H
LOOP	INR B	
	CMP B	
	JNZ	LOOP

In this program before the loop A = 09 H and B = 00 H. The value of B will increase by each go in the loop and each time it is compared with accumulator contents. Till the contents of B register are not equal to the contents of Accumulator, the computer will execute the instructions inside the loop. When the value of B register becomes equal to 09 H, the zero flag will be set and it will allow coming out of the loop to execute the next instructions.

(ii) *CMP M* (Compare Memory)

It is similar to *CMP reg* instruction with the difference that the contents of the addressed memory location will be compared by the accumulator contents. The flags will be modified as per the result.

(iii) CPI data **(Compare immediate with data)**

It is similar to CMP instruction with the difference that the data is directly given with the instruction. In this instruction the given data is compared with the accumulator contents. The flags will be modified as per the result.

(c) Miscellaneous Logical Instructions

(i) *CMC* **(Complement the carry)**

This instruction complements the carry flag.

$$CY \leftarrow \overline{CY}$$

If CY = 1 before the execution of *CMC* instruction, the carry flag will be reset (CY = 0) after the execution of this instruction. Similarly, If CY = 0 before the execution of *CMC* instruction, the carry flag will be set (CY = 1) after the execution of this instruction.

In this only carry flag will be affected and all other flags will not be affected.

(ii) *STC* **(Set the carry)**

It sets the carry flag.

$$CY \leftarrow 1$$

The carry flag will be set (CY = 1) irrespective of the carry flag is set or reset before the execution of this instruction *STC*.

Only carry flag gets affected with this instruction.

4.2.4 Branch Group

This group of instructions changes the normal sequence of the program. The branch instructions include JUMP, CALL and RETURN instructions, which are further sub-divided as unconditional and conditional JUMP, CALL and RETURN instructions.

Table 4.4

Branch Group

Instruction	Op. Code	Addressing modes	No of T-states	No of bytes of instr.	Flags affected	Operation
JMP addr	C3	Immediate	10	3	None	$[PC] \leftarrow [addr]$
JNZ addr	C2	Immediate	10/7	3	None	$[PC] \leftarrow [addr]$ if Z=0.
JZ addr	CA	Immediate	10/7	3	None	$[PC] \leftarrow [addr]$ if Z=1.
JNC addr	D2	Immediate	10/7	3	None	$[PC] \leftarrow [addr]$ if CY=0.
JC addr	DA	Immediate	10/7	3	None	$[PC] \leftarrow [addr]$ if CY=1.
JM addr	FA	Immediate	10/7	3	None	$[PC] \leftarrow [addr]$ if S=1.
JP addr	F2	Immediate	10/7	3	None	$[PC] \leftarrow [addr]$ if S=0.
JPO addr	E2	Immediate	10/7	3	None	$[PC] \leftarrow [addr]$ if P=0.
JPE addr	EA	Immediate	10/7	3	None	$[PC] \leftarrow [addr]$ if P=1.
CALL addr	CD	Immediate	18	3	None	Calls subroutine program.

CNZ addr	C4	Immediate	18/9	3	None	Calls subroutine program if Z=1.
CZ addr	CC	Immediate	18/9	3	None	Calls subroutine program if Z=0.
CNC addr	D4	Immediate	18/9	3	None	Calls subroutine program if CY=1.
CC addr	DC	Immediate	18/9	3	None	Calls subroutine program if CY=0.
CM addr	FC	Immediate	18/9	3	None	Calls subroutine program if S=1.
CP addr	F4	Immediate	18/9	3	None	Calls subroutine program if S=0.
CPO addr	FE	Immediate	18/9	3	None	Calls subroutine program if P=0.
CPE addr	EC	Immediate	18/9	3	None	Calls subroutine program if P=1.
RNZ	C0	Register indirect	12/6	1	None	Returns to main program if Z=1.
RZ	C8	Register indirect	12/6	1	None	Returns to main program if Z=0.
RNC	D0	Register indirect	12/6	1	None	Returns to main program if CY=0.
RC	D8	Register indirect	12/6	1	None	Returns to main program if CY=1.
RM	F8	Register indirect	12/6	1	None	Returns to main program if S=1.
RP	F0	Register indirect	12/6	1	None	Returns to main program if S=0.
RPO	E0	Register indirect	12/6	1	None	Returns to main program if P=0.
RPE	E8	Register indirect	12/6	1	None	Returns to main program if P=1.

(a) Unconditional Jump Instructions

(i) *JMP address* (label) (Jump to Address)

This is an unconditional jump instruction. With the execution of this instruction, the program jump to the address (or label) specified with the instruction. This is a three byte instruction and no flag is affected. In fact during the execution of *JMP address* (label) instruction, the address of the label is copied in the program counter; and whenever the program fetches the next instruction the program counter will send the address of this given label.

$$[PC] \leftarrow [LABEL]$$

(a) Conditional Jump Instructions

In conditional jump instructions the program jumps to the instructions specified by the address (or label) if the given condition is fulfilled. However, if the given condition is not satisfied, the program will not jump to the specified address (or label)

rather it will proceed to the normal sequence. In all these conditional jump instructions, if program jumps to the specified address after the given condition is satisfied, it will take 10 T-states for its execution otherwise it takes 7 T-states.

Following are the conditional Jump instructions. Some of them have also been used in SAP-II computers.

(i) *JNZ address* (label) (Jump if the result is not zero)

When this instruction is executed, the program jumps to the instruction specified by the address (or label), if the result is not zero; otherwise it will proceed to the next instruction. Here the result of the preceding instruction is considered.

In this case the address (or label) is copied in the program counter if zero flag is reset (Z = 0).

$$[PC] \leftarrow LABEL \text{ if } Z = 0.$$

This is illustrated if we consider the following instructions:

Label	Mnemonics	Operand

	DCR	C
	JNZ	NEXT
	MOV A, M	
NEXT	STA	2500 H
	HLT	

For the execution of JNZ instruction, the result of C-register will be considered. If $[C] \neq 0$ (Z = 0), it will jump to the instruction (*STA 2500 H*) specified by the label NEXT, otherwise it will jump to the next instruction (*MOV A, M*).

(ii) *JZ address* (label) (Jump if the result is zero)

The condition of this instruction is reverse to that of *JNZ address*. In this case the program will jump to the instruction specified by the address (or label), if the result of the preceding instruction is zero (Z = 1) otherwise it will proceed to the next instruction in the normal sequence.

The address of the label will be copied in the program counter if Z flag is set (or result is zero).

$$[PC] \leftarrow LABEL \text{ if } Z = 1.$$

(iii)*JNC address* (label) (Jump if no carry)

During the execution of this instruction, it will check up the Carry flag modified by the preceding instruction. If there is no carry (CY = 0 or CY flag is reset), the program will jump to the instruction specified by the address (or label) otherwise it will proceed to the next instruction of the normal sequence.

In this case the address of the label will be copied in the program counter if CY = 0 or CY is reset.

$$[PC] \leftarrow LABEL \text{ , if } CY = 0$$

(iv) *JC address* (label) (Jump if carry)

The program will jump to the instruction specified by the address (label) if the CY flag is set (CY = 1) which is modified by the preceding instruction. However, if the carry

is reset (CY = 0), it will proceed to the next instruction of the normal sequence. The condition of this instruction is opposite to that of the *JNC address*.

In this case the address of the label will be copied in the program counter if CY = 1 or CY is set.

$$[PC] \leftarrow LABEL \text{ , if CY = 1}$$

(v) *JM address* (label) (Jump if Minus)

When this instruction is executed, the program will jump to the instruction specified by the address (label) if the result of the preceding instruction is minus or sign flag is set (S = 1) otherwise it will proceed to the next instruction of the normal sequence.

$$[PC] \leftarrow LABEL \text{ , if S = 1}$$

(vi) *JP address* (label) (Jump if Positive)

If the result of the preceding instruction is positive or sign flag is reset (S = 0), the program will jump to the instruction specified by the address (label). However, if the condition is not satisfied it will proceed to the next instruction of the normal sequence.

$$[PC] \leftarrow LABEL \text{ , if S = 0}$$

(vii) *JPO address* (label) (Jump if Parity is Odd)

If the parity is odd or parity flag is reset (P = 0) as a result of the preceding instruction, the program will jump to the instruction specified by the address (label) otherwise next instruction of the normal sequence will be executed.

$$[PC] \leftarrow LABEL \text{ , if P = 0}$$

(vii) *JPE address* (label) (Jump if Parity is Even)

If the parity is even or parity flag is set (P = 1) as a result of the preceding instruction, the program will jump to the instruction specified by the address (label) otherwise next instruction of the normal sequence will be executed.

$$[PC] \leftarrow LABEL \text{ , if P = 1}$$

(c) CALL Instructions

The call instructions allow calling the subroutine program. The address of the subroutine program is specified with the CALL instruction. During the execution of CALL instruction, the current contents of program counter are saved on the stack and the address of subroutine (specified with the CALL instruction) is copied in the program counter. Like the jump instructions, the CALL instructions are also of two types.

1. Unconditional Call Instructions
2. Conditional Call Instructions

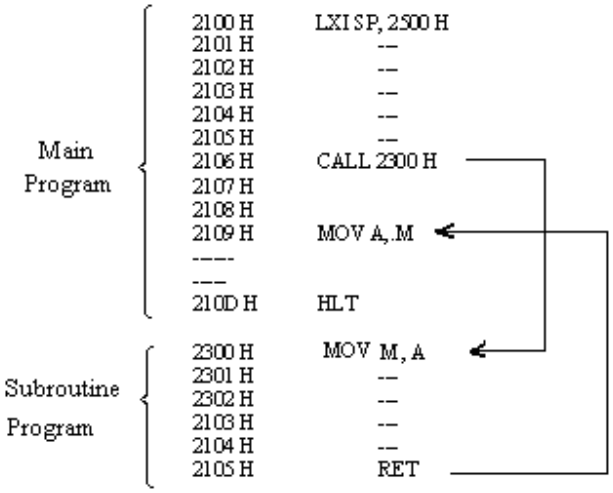
1. Unconditional Call Instructions

CALL address **(Calls the addressed subroutine program)**

This is the format for unconditional call instruction which is of three bytes. The current contents of program counter are saved on the Stack and the address specified with CALL instruction is copied in the program counter.

i.e.
$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and
$$[PC] \leftarrow address$$

To illustrate the CALL address instruction, consider the following assembly language program:



In this program *LXI SP, 2500 H* initializes the stack pointer (stack pointer is represented by 2500 H.) i.e. the area or memory locations less than 2500 H will be used for stack purposes. When the instruction *CALL 2300 H* is executed, the address of the program counter will be 2109 H. The address of the program counter will be saved on the stack. For this stack pointer will be decremented by 1 and the high byte of the program counter (21 H) will be saved on to it; the stack pointer will further be decremented by 1 and lower byte of the program counter (09 H) will be stored on to it. The decremented of the stack pointer will be current position of the stack. This is shown in figure 4.8.

The address given with the CALL instruction i.e. 2300 H will be saved or copied in the program counter.

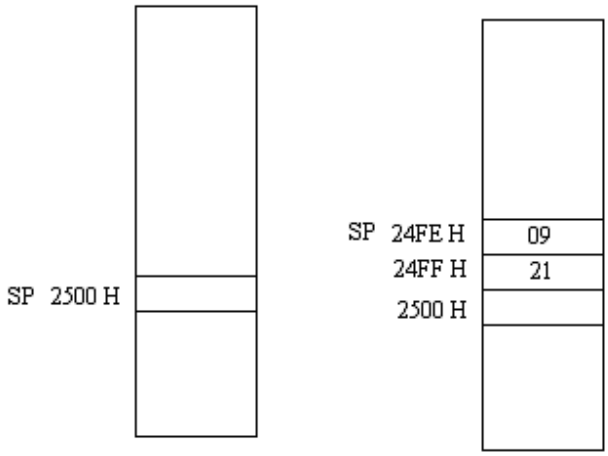


Fig. 4.8

The subroutine program ends at the instruction RET i.e. RET instruction will be the last instruction of the subroutine program. The RET instruction takes the computer back to the main program. The return saved (or pushed) on to the stack will be popped back to the program counter. Further the data stored (or saved) at the top of the stack will be copied as the lower byte of the program counter i.e. $[PCL] \leftarrow [M_{SP}]$. The program

counter will now be incremented by one and data stored in new stack will be saved to higher byte of the program counter i.e. $[PCH] \leftarrow [M_{SP+1}]$.

In the above program, after the execution of RET instruction **PC = 2109 H**.

2. Conditional Call Instructions

Similar to conditional jump instruction SAP-III computer has the following conditional instructions.

(i) CNZ address **(Call if not zero)**

This instruction will call the subroutine program specified by the address provided the zero flag is reset.

i.e. if Z = 0 or the result is not zero then

$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and $$[PC] \leftarrow address$$

(ii) CZ address **(Call if zero)**

It will call the subroutine (specified by the address) if zero fags is set.

i.e. if Z = 1 or the result is 1 zero then

$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and $$[PC] \leftarrow address$$

(iii) CNC address **(Call if no carry)**

This instruction will check the carry flag. If the carry flag is reset (or CY=0) as per the executed preceding instruction, then the computer will call the addressed subroutine program.

i.e. if CY = 0 or there is no carry, then

$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and $$[PC] \leftarrow address$$

(iv) CC address **(Call if carry)**

The addressed subroutine will be called by the computer, there is a carry (or carry flag is set ; CY = 1) as per the result of the preceding instruction.

i.e. if CY = 1 or there is a carry, then

$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and $$[PC] \leftarrow address$$

(v) CM address **(Call if minus)**

During the execution of this instruction the sign flag will be checked. If the result of the preceding instruction is negative or sign flag is set (S = 1), then only the computer will call the subroutine program specified by the address.

So if S = 1, then

$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and $$[PC] \leftarrow address$$

(vi) CP address **(Call if positive)**

In this CALL instruction too the sign flag will be checked. If the result of the preceding instruction is positive or sign flag is reset (S = 0), then the computer will call the subroutine program specified by the address.

So if S = 0, then

$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and $$[PC] \leftarrow address$$

(vii) CPO address **(Call if parity is odd)**

If the parity of the result of the preceding instruction is odd (or parity flag is reset; P = 0), then the computer will call the addressed subroutine program.

So if P = 0, then

$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and $$[PC] \leftarrow address$$

(ix) CPE address **(Call if parity is even)**

The computer will call the addressed subroutine program, if the parity of the result of the preceding instruction is even (or parity flag is set; P = 1), then.

So if P = 0, then

$$[SP-1] \leftarrow [PCH]$$
$$[SP-2] \leftarrow [PCL]$$
and $$[PC] \leftarrow address$$

Return Instructions

As already discussed, the last instruction of the subroutine program is RET (Return instruction). This return instruction takes the computer back to the main program. The return instructions may also be conditional. The following are the conditional return instructions:

RNZ	**Return if no zero**
RZ	**Return if zero**
RNC	**Return if no carry**
RC	**Return if carry**
RM	**Return if minus**
RP	**Return if positive**
RPO	**Return parity is odd**
RPE	**Return parity is even**

All the above return instructions are one byte instructions. If the specified condition is satisfied, it will return to the main program; otherwise the execution of the next instruction in the subroutine will be carried out. Further the return instruction will pop the return address to the program counter.

For example, we consider the instruction ***RNC.***

It will check the carry flag. Only if the carry flag is reset (CY = 0 or no carry), the computer return to main program.

i.e. if CY = 0 , then

$$[PCL] \leftarrow [SP]$$
$$[PCH] \leftarrow [SP+1]$$

4.2.5 Stack and Input / Output Instructions

Stack is a portion of read/write memory, which is primarily used for saving return address and data. As discussed earlier, in SAP-II computers, last two memory locations of the read / write memory FFFF H and FFFE H are exclusively used for return address of a subroutine call. In SAP-III computers, the programmer can specify any area of the memory for the stack purposes. The portion of memory located chosen by the programmer for the stack purposes can not be used for saving the return address of subroutine call. For this purpose a 16 bit register called stack pointer is provided.

Table 4.5

Stack I/O Group

Instruction	Op. Code	Addressing modes	No of T-States	No of bytes of instr.	Flags affected	Operation
LXI SP, addr	31	Immediate	10	3	None	
PUSH B	C5	Register indirect	12	1	None	
PUSH D	D5	Register indirect	12	1	None	
PUSH H	E5	Register indirect	12	1	None	
PUSH PSW	F5	Register indirect	12	1	None	
POP B	C1	Register indirect	10	1	None	
POP D	D1	Register indirect	10	1	None	
POP H	E1	Register indirect	10	1	None	
POP PSW	F1	Register indirect	10	1	None	
IN Port	DB	Immediate	10	2	None	
OUT Port	D3	Immediate	10	2	None	

The stack is initialized by the instruction given below:

LXI SP, address **(Load immediately the stack pointer)**

For example *LXI SP, 2500* H will indicate the stack pointer as 2500 H as

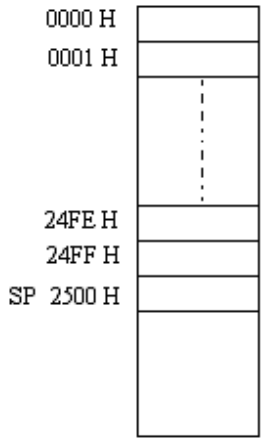


Fig. 4.9

shown in figure 4.9. So the memory locations lower than 2500 H may be used for stack purposes i.e. the data or return addresses may be pushed to stack in the sequence discussed earlier in CALL instructions.

Following are the Stack instructions:

(i) Push Instructions

In the stack not only the return address of the main program during the execution of CALL address instruction is pushed but also the contents of registers may also be pushed to stack. For this following is the format of PUSH instructions:

PUSH rp

where rp stands for register pair. It may be
B for B-C register pair,
D for D-E register pair,
H for H-L register pair,
PSW is known as program status word; it represents the accumulator and the flag contents i.e. **A F**

So we have the following PUSH instructions.

```
PUSH B      ; it pushes or saves the contents of B
              and C registers on the stack.
PUSH D      ; it pushes or saves the contents of D
              and E registers on the stack.

PUSH H      ; it pushes or saves the contents of H
              and L registers on the stack.

PUSH PSW    ; it pushes or saves the contents of
              accumulator (A) and F (flag)
              registers on the stack.
```

Following steps are carried out during the execution of ***PUSH rp*** instructions.

1. The stack pointer is decremented by one to get a new value of stack pointer as SP-1.

2. The contents of higher register of the given register pair (say B register in B-C register pair.

   i.e. $[M_{SP-1}] \leftarrow [rp_H]$

3. The stack pointer is further decremented by one to get new value of stack pointer as SP-2.
4. The contents of lower register (C register in B-C register pair) is saved in the memory location specified by SP-2.

   i.e. $[M_{SP-2}] \leftarrow [rp_L]$

5. SP-2 will now be the stack pointer for other PUSH instructions.

For example we have SP = 2500 H
HL = 5678 H

After the execution of PUSH H instruction 56 H will be loaded to memory location 24FF H and 78 H will be loaded to 24FE H memory location as shown in figure 4.10.

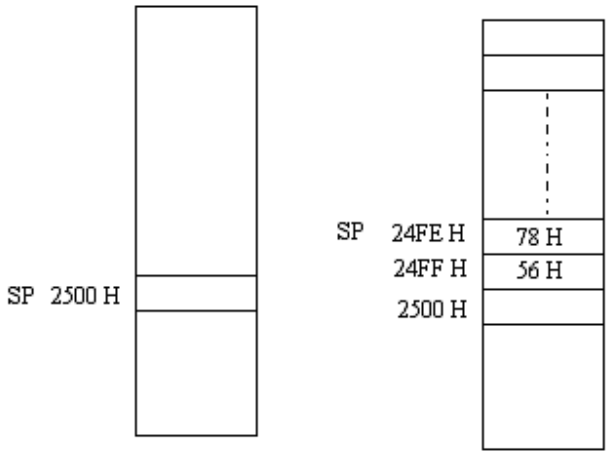


Fig. 4.10

(ii) Pop Instructions

To retrieve the contents of registers from the stack, following POP instructions are used.

POP B	; where B stands for B-C register pair.
POP D	; where D stands for D-E register pair.
POP H	; where H stands for H-L register pair.
POP PSW	; where PSW stands for Program status word i.e. accumulator (A) and F (flag) register.

During the execution of POP instructions following steps are carried out.

1. The byte stored in memory location addressed by stack pointer is saved in the lower register of the given register pair (say C in B-C register pair). The contents of higher register of the given register pair (say B register in B-C register pair.
   i.e. $[rp_L] \leftarrow [M_{SP}]$
2. The stack pointer is incremented by one to get new value of stack pointer as SP+1.
3. The contents stored in the memory location addressed by stack pointer (new value as SP+1) is saved in higher register of the given register pair (B register in B-C register pair).
   i.e. $[rp_H] \leftarrow [M_{SP+1}]$
4. The stack pointer is further incremented by one to get the new value of stack pointer as SP+2 i.e. SP+2 will now work as stack pointer.

Suppose the data saved in stack is shown in figure 4.11. The stack pointer is shown as 24FC H.

Let A = 01 H F = 00 H
D = 12 H E = 6E H

before the execution of

POP PSW

and POP D instructions.

After the execution of these instructions we have the following data in the registers.

F = 59 H
A = 24 H
E = 36 H
D = 9F H

Now 2500 H will be the new value of stack pointer.

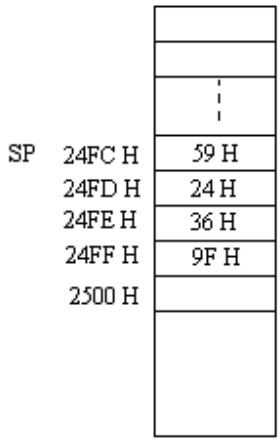


Fig. 4.11

It should be remembered that when a subroutine is called, it should save the contents of any register being used in the main program using PUSH instruction in the stack; and when it returns back to the main program form the subroutine program the

contents of the register should be retrieved from the stack using POP instruction as discussed above.

The sequence of PUSH and POP instructions should be as:

```
PUSH  PSW
PUSH  D
PUSH  B
PUSH  H
----------
----------
POP  H
POP  B
POP  D
POP PSW
```

It should be noted that POP retrieve the data from the stack just in reverse order of the data was PUSHed i.e. as Last In First Out (LIFO) method is used in the stack.

PUSH and POP instructions may also be used in the main program. PUSH instruction is used before the CALL instruction and POP (in the reverse order) after the CALL instruction in the main program as given below:

Main Program:

```
----------
----------
PUSH  PSW
PUSH  H
CALL  LABEL
POP  H
POP PSW
-----------
------------
```

In this program the contents of Accumulator, Flag, H and L registers are saved in the Stack before the execution of CALL instruction. After the execution of subroutine program the contents of the Accumulator, Flag, H and L registers are retrieved and next instructions of the main program will be executed with restoration of earlier values of the registers.

Further, the PUSH and POP instructions may be used in the subroutine program as given below. This is generally done when the CALL instruction is used may times.

Main Program:

```
----------
----------
CALL  LABEL
-----------
-----------
```

Subroutine Progam:

```
LABEL    PUSH  PSW
         PUSH  D
         PUSH  B
         ----------
         ----------
         POP  B
         POP  D
         POP  PSW
         RET
```

Input / Output Instructions

The input / output instructions deal with the input / output operations. These instructions are same used in SAP-II computer.

(i) IN Port **(Inputs the data from the port)**

In this instruction the data available at the specified port address is moved to the accumulator.

$$[A] \leftarrow \textit{data from port}$$

It is two byte instruction and no flag is affected after the execution of this instruction. This instruction is basically used to read the data from the input devices such as data read from key board, switches etc. during the computer run.

(ii) OUT Port **(Outputs the data to the port)**

This two byte output instruction is used to send the contents of accumulator to the specified port.

$$\textit{Output} \leftarrow [A]$$

4.3 TIME DELAY INTRODUCED BY A REGISTER PAIR

The time delay introduced by different registers has been discussed in the last chapter using SAP-II instructions. The time delay can considerably be increased by using a register pair. A 16-bit number may be taken in two registers (register pair).

Consider the following subroutine assembly language program:

Label	Mnemonics	Operand	Comments
	LXI D,	F424 H	;Loads DE register pair with a 16-bit number.
LOOP	DCX	D	;Decrements DE register pair by one.
	MOV A,	E	;Moves the contents of E register to accumulator.
	ORA	D	;ORing of the contents of D and E registers are performed to set the zero flag.
	JNZ	LOOP	;If result $\neq 0$ jump to loop
	RET		;Go back to main program.

Total T-states used for the above program are given as:

Mnemonics	T-states
LXI	10
DCX	5
MOV	5
ORA	4
JNZ LOOP	10/7
RET	10

24 T-states are used for the inner loop and 10+7+10 = 27 T-states are used for outer loop.

In this program the execution of loop is for 62500 times (as F424 H = 62500_{10}). The condition for the check of zero flag can not be applied just after DCX instruction, since no flag gets affected with this instruction. So to check the zero flag ORA instruction affect the zero flag. The zero flag will be set if the contents of both D and E registers are zero.

The time delay introduced by the inner loop is:

$$T_{LOOP} = 62500 \text{ x } 24 \text{ x Time of one T-state.}$$

If the system clock frequency is 3 MHz, then

$$T_{LOOP} = 62500 \text{ x } 24 \text{ x } \frac{1}{3}\mu\text{sec}$$
$$= 500000\ \mu\text{sec}$$
$$= 0.5 \text{ sec.}$$

Delay introduced for outside loop is:

$$T_{out} = 27\text{x}1\text{x}\frac{1}{3}\mu\text{sec}$$
$$= 9\ \mu\text{sec}$$

So the total time delay introduced by the above subroutine program is given by:

$$T_{Delay} = 0.5 \text{ sec} + 9\ \mu\text{sec}$$
$$\approx 0.5 \text{ sec}$$

Example 4.1 *What maximum delay can be introduced by the subroutine program using a register pair? Let the system clock frequency is 3 MHz.*

Solution. The subroutine program for the delay using a register pair is given as:

Label	Mnemonics	Operand
	LXI D,	FFFF H
LOOP	DCX	D
	MOV	A, E
	ORA	D
	JNZ	LOOP
	RET	

The maximum delay can be introduced if the maximum number (FFFF H) is taken in DE register pair.

FFFF H = 65535_{10}

So $$T_{LOOP} = 65535 \text{ x } 24 \text{ x } \frac{1}{3}\mu\text{sec} + 27\text{x}1\text{x}\frac{1}{3}\mu\text{sec}$$
$$= 524280+9\ \mu\text{sec}$$

= 524289 μsec
= 0.52 sec

Example 4.2 *Write a program in assembly language to introduce a time delay of 1 sec using a register pair. Let the system clock frequency is 3 MHz.*

Solution. For one sec delay, the program discussed in section in 4.3 can be executed twice as given below:

Main Program:

Label	Mnemonics	Operand
	MVI C,	02 H
LOOP1	CALL	DELAY
	DCR	C
	JNZ	LOOP1
	HLT	

Subroutine Program:

Label	Mnemonics	Operand
DELAY	LXI D,	F424 H
LOOP	DCX	D
	MOV	A, E
	ORA	D
	JNZ	LOOP
	RET	

In this program the subroutine program is run two times as subroutine program in one go introduces a time delay of 0.5 sec.

Example 4.3 *It is desired to clear the accumulator contents of a SAP-III computer. Explain the possible instructions for this purpose.*

Solution. To clear the accumulator contents, the possible instructions are:

(i)	MVI A, 00 H	Two byte instruction and no flag gets affected.
(ii)	XRA A	One byte instruction, CY flag will be reset and all other flags will be modified as per the result.
(iii)	ANI 00 H	Two byte instruction, CY flag will be reset and all other flags will be modified as per the result.
(iv)	SUB A	One byte instruction and all flags will be affected as per the result.

Example 4.4 *Write a program in assembly language using instructions of SAP-III computers to complement the 256 bytes stored in memory locations 2500 H through 25FF H. Store the complemented data in memory locations starting at 2600 H.*

Solution.

Label	Mnemonics	Operand	Comments
	LXI H,	2500 H	;Loads the address of the first number in H-L register pair.
	MVI C,	FF H	; 256 bytes (FF H) is stored in C-register as index register.
LOOP	MOV A,	M	;Moves the contents of memory location addressed

			by H-L register pair to accumulator.
	CMA		;complements the accumulator contents.
	MVI H,	26 H	;Stores 26 H in H-register
	MOV M,	A	;Complemented data is stored in the destination memory location.
	MVI H,	25 H	;Stores 25 H in H-register
	INX H		;Increments the H-L register pair.
	DCR C		;Decrements the contents of C register.
	JNZ	LOOP	;Go back to loop for the operation of next data.
	HLT		

Example 4.5 *Write a program in assembly language using instructions of SAP-III computers to find the smaller of two numbers stored in memory locations 2501 H and 2502 H. Store the result in 2503 H memory location.*

Solution.

Label	Mnemonics	Operand	Comments
	LXI H,	2501 H	;Loads the address of the first number in H-L register pair.
	MOV A,	M	;Saves the first number in accumulator.
	INX H		;Increments the H-L register pair.
	CMP M		;compares the two numbers.
	JC	NEXT	;if carry smaller number is in accumulator go to NEXT.
	MOV A,	M	;If no carry then move the smaller number to accumulator.
NEXT	STA	2503 H	;Stores the smaller number in 2503 memory location .
	HLT		

Example 4.6 *Write a program in assembly language using instructions of SAP-III computers to add two hexadecimal numbers stored in memory locations 2501 H and 2502 H. The answer should be stored in 2503 H memory location. The carry if any should be stored in 2504 H memory location.*

Solution.

Label	Mnemonics	Operand	Comments

```
                LXI  H,     2501 H    ;Loads the address of the
                                       first number in H-L register
                                       pair.
                MVI C,      00 H      ;Clears C-register for carry
                                       (most significant bit).
                MOV  A,     M         ;Saves the first number in
                                       accumulator.
                INX  H                ;Increments the H-L
                                       register pair.
                ADD  M                ;Adds the two numbers.
                JNC         NEXT      ;if no carry go to NEXT.
                INR  C                ;Else increment the content
                                       of C-register.
NEXT            STA         2503 H    ;Stores the answer .
                MOV  A,     C         ;Moves the contents of C
                                       register (carry contents) to
                                       accumulator
                STA         2504 H    ;Saves the contents of
                                       carry.
                HLT
```

Example 4.7 *Write a program in assembly language for SAP-III computers to find the sum of a series 1+2+3+……+10 (or sum of first 10 natural numbers).*

Solution.

```
Label           Mnemonics   Operand   Comments
                MVI B,      01 H      ;Loads first number of series
                                       to B-register.
                MVI C,      00 H      ;Clears C-register.
LOOP            MOV  A,     C         ;Saves the contents of C-
                                       register to accumulator.
                ADD  B                ;Adds the contents of B-
                                       register with the contents of
                                       accumulator.
                MOV  C,     A         ;Saves the partial sum to C-
                                       register.
                INR  B                ;increments the contents of
                                       B-register.
                CPI  0B H             ;Compares with natural
                                       number 11.
                JNZ         LOOP      ;If not 11 go back to LOOP.
                MOV  A,     C         ;Saves the answer to
                                       accumulator.
                STA         2500 H    ;Saves the answer to
                                       memory location.
                HLT
```

Example 4.8 *Reset all the flags, in a subroutine program, without performing arithmetic or logical instructions. However the contents of other registers should not be changed.*

Solution.

Label	Mnemonics	Operand	Comments
	PUSH H		;Saves the contents of H-L register pair to stack.
	PUSH PSW		;Saves the contents of A-F to stack.
	LXI H,	0000 H	;Loads 0000 H to H-L register pair.
	PUSH H		; Saves the contents of H-L register pair to stack.
	POP PSW		;Saves the contents of stack in A-F registers.
	POP H		; Retrieve the contents of H-L register pair from the stack.
	MOV A,	H	;Moves the contents of H-register to accumulator.
	POP H		;Pops the contents from stack to H-L register pair.
	RET		

Example 4.9 *How will you exchange the contents of BC and HL register pairs?*
Solution. Following program may exchange the contents of BC and HL register pairs:

Label	Mnemonics	Operand	Comments
	LXI SP,	XXX H	;Initializes the stack pointer.
	PUSH B		;Saves the contents of B-C register pair to stack.
	PUSH H		;Saves the contents of H-L register pair to stack.
	POP B		;Pops the contents of stack (H-L register pair contents) to B-C register pair.
	POP H		;Pops the contents of stack (B-C register pair contents) to H-L register pair.
	HLT		

Example 4.10 *Write a subroutine program using SAP-III instructions, which will subtract the contents of D-E register pair from H-L register pair and the result is to be stored in H-L register pair.*
Solution.

Label	Mnemonics	Operand	Comments
	MOV A,	L	;Transfer the contents of L register to accumulator.
	SUB E		;Contents of E register are subtracted from accumulator contents.

	MOV L,	A	;Answer (list significant digit of the answer) is loaded to L register.
	MOV A,	H	;Moves the contents of H register to accumulator.
	SBB D		;Subtracts the content of D register from acc. with carry.
	MOV H,	A	;Most significant digit of the answer is saved in H register.
	HLT		

Example 4.11 *What will be the value of accumulator and flags (CY, S, P and Z), after the execution of the following program.*

Label	***Mnemonics***	***Operand***
	MVI D,	*7F H*
	MVI C,	*3E H*
	MOV A,	*C*
	RLC	
	RLC	
	ANA D	
	HLT	

Solution. After the execution of the above program we have:

	D = 0 1 1 1 1 1 1 1	
	C = 0 0 1 1 1 1 1 0	
	A = 0 0 1 1 1 1 1 0	
After first RLC	A = 0 1 1 1 1 1 0 0	CY = 0
After second RLC	A = 1 1 1 1 1 0 0 0	CY = 0
ANDing of D	= 0 1 1 1 1 1 1 1	

The value of accumulator is	A = 0 1 1 1 1 0 0 0 = 78 H	
The values of Flag register are:	CY = 0 (reset after ANDing)	
	S = 0	
	P = 1	
	Z = 0	

Example 4.12 *Write a program using SAP-III instructions to check the even parity or the odd parity of the number stored in memory location 2010 H. Send 00 H or EE H at the output port 02 H if the parity is odd or even respectively.*

Solution.

Label	**Mnemonics**	**Operand**	**Comments**
	LXI H,	2010 H	;Initializes the H-L register pair with the address of the location.
	MOV A,	M	;Moves the number to accumulator.

	ORA A		;ORing of A with A will load the same number to accumulator. The parity flag will be affected with this operation.
	JPO	ODD	;Jump to ODD if parity is odd.
	MVI A,	EE H	;Load EE H to accumulator for even parity.
	OUT	02 H	;EE is sent to output port 02H.
	JMP	END	;Jump to end.
ODD	MVI A,	00 H	;Load 00 H to accumulator for odd parity.
	OUT	02 H	;00 is sent to output port 02H.
END	HLT		

Example 4.13 (a) *What will be the value of B-register after the following program is executed.*

Label	***Mnemonics***	***Operand***	***Comments***
	MVI A,	*07 H*	
	STC		
	CMC		
	RAL		
	MOV B,	*A*	
	STC		
	CMC		
	RAL		
	STC		
	CMC		
	RAL		
	ADD B		
	MOVB,	*A*	
	HLT		

(b) *Suggest some alternative program that can perform the same operation.*

Solution. (a) In the beginning accumulator is loaded with a hexadecimal number 07 H.

The instruction *STC* and *CMC* resets the carry flag and RAL instruction moves the accumulator contents left through carry giving us the contents 0E H ($=14_{10}$) i.e. *STC*, *CMC* and *RAL* instructions multiplies the accumulator contents by a factor of two. The accumulator contents are now saved in register B.

Three instructions *STC*, *CMC* and *RAL* are further used twice so that accumulator contents are multiplied by a factor of 4. This way after the third *RAL* in the program we have A = 8 X 07 H = 38 H = 56_{10}.

After *ADD B* we get:

A = 38 H + 0E H
= 46 H
= 70_{10}

So this program multiplies the accumulator contents by a factor of 10 (decimal number) which is stored in B-register.

So B = 46 H

$= 70_{10}$

(b) The program given in (a) multiplies the accumulator contents by a factor of 10 (decimal number) which is stored in B-register. The following program also perform the same operation.

Label	Mnemonics	Operand	Comments
	MVI A,	07 H	;Loads 07 H to accumulator.
	ANI	FF H	;It clears the carry flag.
	RAL		;Shifts left by one bit or multiplies accumulator contents by a factor or two.
	MOV B,	A	;Moves the answer to B-register.
	ANI	FF H	;It clears the carry flag.
	RAL		;Shifts left by one bit or multiplies accumulator contents by a factor or two. So A = 4 x A.
	ANI	FF H	;It again clears the carry flag.
	RAL		;Shifts left by one bit or multiplies accumulator contents by a factor or two. So A = 8 x A.
	ADD B		; $[A] \leftarrow [A] + [B]$ So it gives A = 10 x A
	MOV B,	A	;Result is stored in B register.
	HLT		

This program does the same operation as given in (a), however, less number of instructions is used in this program.

Example 4.14 *A byte of data is stored in memory location 2500 H. Display at the output port 02 H the number of 1's present in the data byte stored in memory location 2500 H.*

Solution.

Label	Mnemonics	Operand	Comments
	LDA	2500 H	;Loads the data byte in accumulator.
	MVI C,	08 H	;08 H is loaded to C-register to use it as a counter.
	MVI B,	00 H	; 00 H is loaded to B-register.
LOOP	RLC		;MSB is shifted to carry flag.
	JNC	NEXT	;Checks the carry flag, if not zero go to next (i.e. no 1 is present at the MSB).
	INR B		;Increments the B-register by one.

```
NEXT    DCR  C          ;Decrements C-register.
        JNZ     LOOP    ;Jumps to LOOP if Z = 0.
        MOV  A, B       ;No. of 1's present is loaded
                        to accumulator.
        OUT     02 H    ;Sends the data to the output
                        port.
        HLT
```

PROBLEMS

1. Give the programming model (Software model) of SAP-III computers.
2. How many flags are used in SAP-III computers? How are these flags affected with the result?
3. Explain with the help of block diagram how the carry flag gets affected with the result.
4. In how many groups the instruction set of SAP-III computers are classified. Explain with examples the instructions of arithmetic group.
5. Discuss the rotate instructions used in SAP-III computers. How the flags are affected with these instructions.
6. What is the difference between the stack and stack pointer? Discuss PUSH and POP instructions.
7. Explain what operations are performed when the following instructions are executed. Give at least one example for each operation.
   (i) LXI H, address
   (ii) ADD M
   (iii) CMP reg
8. What is difference between *CMP reg* and *SUB reg* instructions? Explain with suitable examples.
9. If carry flag is zero, then show that RAL instruction produces a multiplication of accumulator contents by a factor of 2.
10. If carry flag is zero, then show that RAR instruction produces a division of accumulator contents by a factor of 2.
11. Show that the instruction DAD H shifts left the data in H-L register pair by one bit.
12. Discuss various addressing modes of SAP-III computers instructions and give at least two examples of each addressing mode.
13. What do you understand by subroutine? Explain the use of stack and stack pointer in calling a subroutine. Explain any other use of stack.
14. Write a program for SAP-III computers introduce a time delay of 0.5 sec if the system clock frequency is 6 MHz.
15. Explain the following instruction of SAP-III computer. Also discuss which flags get affected with the execution of these instructions.
    ADD M, CMP B, POP PSW and MOV A, M.
16. An *ORA reg* or *ORI data* instruction can be used to make a selected bit of a specified register as 1. Justify this statement. Also write the mnemonic of an instruction that will set bit 6 of the accumulator without changing any of the other bits in the register. **(Ans.: ORI 40 H)**

17. What will be contents of accumulator and flags (CY, S, P and Z), after the execution of ADD D instruction; if A = C3 H and D = 3D H.

(Ans: A= 00 H, CY = 1, S = 0, P = 1 and Z = 1)

18. What will be contents of accumulator and flags (CY, S, P and Z), after the execution of SUB D instruction; if A = C3 H and D = 3D H.

(Ans: A= 85 H, CY = 0, S = 1, P = 0 and Z = 0)

19. What will be contents of memory location 2500 H and flags (CY, S, P and Z), after the execution of the following program:

```
MVI C, C8 H
MVI A, 11 H
ADD C
STA 2500 H
HLT
```

(Ans: M_{2500} = D9 H, CY = 0, S = 1, P = 0 and Z = 0)

20. What will be the value of stack pointer, after the following program written in SAP-III instructions is executed.

```
LXI SP, 27FF H
PUSH D
CALL SUBROUT
POP D
ADD D
PUSH D
HLT
```

What will be the value of stack pointer after this program?

(Ans: SP = 27FD H)

21. What will be the contents in B-register, after the execution of the following assembly language program?

Label	Mnemonics	Operand
	MVI A,	08 H
	STC	
	CMC	
	RAL	
	MOV B,	A
	HLT	

(Ans.: B = 10 H = 16_{10})

5

The 8085 Microprocessor

After the study of conceptual computers SAP-I, SAP-II and SAP-III, we are now in the position to understand the fundamentals of 8-bit microprocessor 8085. As discussed in the preceding chapters, the basic components of all the computers are Keyboard, Display devices, Memory devices and Central Processing unit. The central processing unit (CPU) is the heart of the computer and also called microprocessor.

The technological revolution brought in recent years, the invention of micro-programmable computer on microprocessor chip. First four bit microprocessor chip INTEL-4004 was developed by Intel Corporation of America in 1971. Intel introduced in 1972 an 8-bit microprocessor 8008 and in 1973 another 8-bit microprocessor 8080. The microprocessor 8080 was the most popular microprocessor of the early 70s. In the year 1974, Intel developed a 40 pin microprocessor chip 8085, which was the enhanced version of 8080. Intel 8080 microprocessor was not in fact a complete CPU on a chip because the clock and controller were on separate chip. Further, it utilizes two separate power supplies. The 8085 microprocessor has the advantages over 8080 that it has on-chip clock and control circuit. It needs only one power supply of +5 volts.

5.1 ARCHITECTURE OF 8085 MICROPROCESSOR

Intel 8085, an 8-bit NMOS microprocessor is available in the form of 40 Pin dual in line IC package. It is fabricated on a single LSI chip. It operates on +5 V d.c. supply. The clock speed used in this microprocessor is about 3 MHZ. General Purpose 8-bit microprocessor is capable of addressing up to 64 K bytes (i.e. $2^{16} = 65536$ bytes) of memory. The functional block diagram is shown in figure 5.1.

The main functional components of 8085A microprocessor are as given below:

(i) Register Section
(ii) Arithmetic and Logic Unit
(iii) Timing and Control Section
(iv) Interrupt Control
(v) Serial Input / Output Control

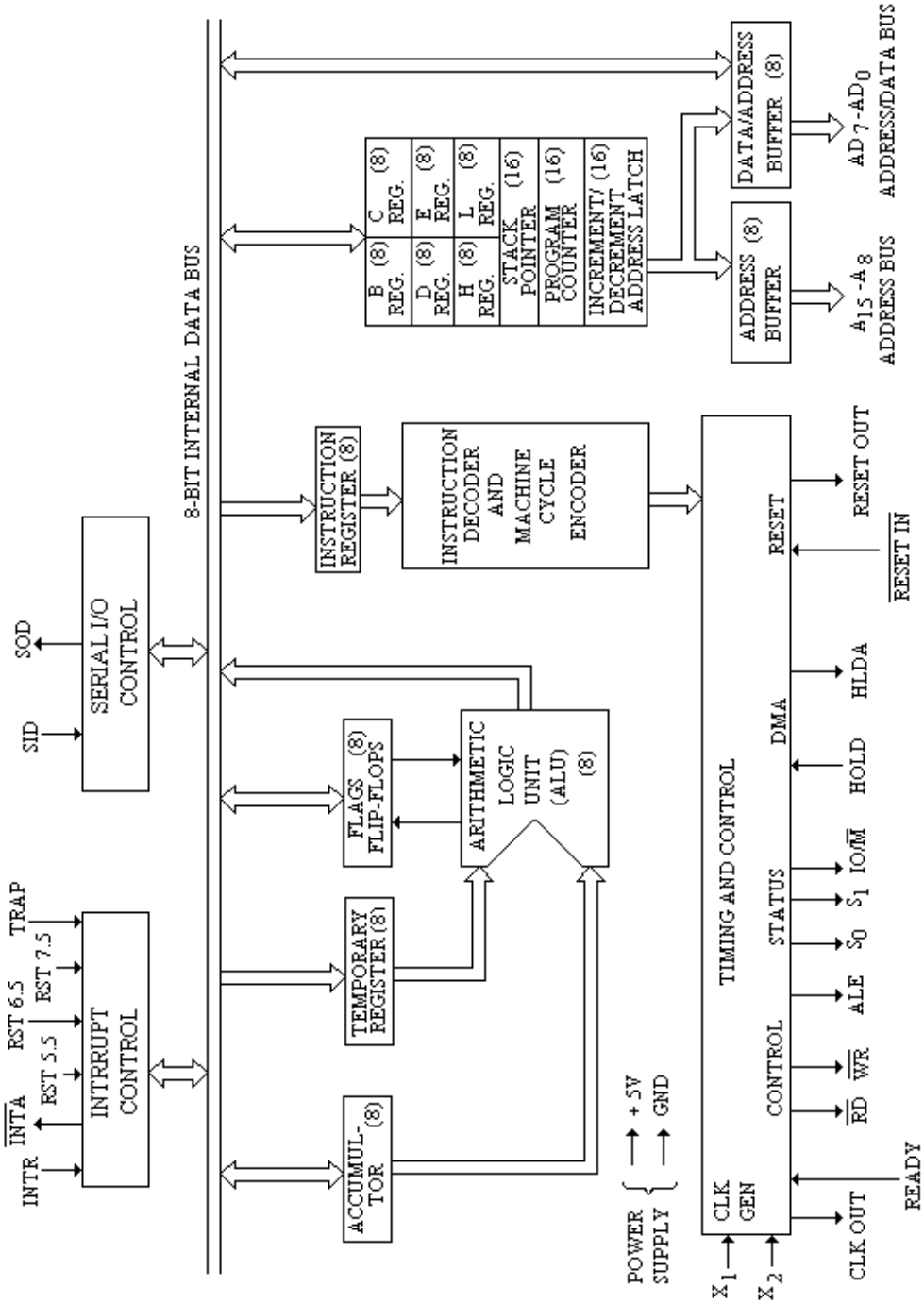

8-BIT INTERNAL DATA BUS
SERIAL I/O CONTROL
SOD
SID
INTERRUPT CONTROL
INTR
INTA
RST 5.5
RST 6.5
RST 7.5
TRAP
ACCUMULATOR (8)
TEMPORARY REGISTER (8)
FLAGS (8) FLIP-FLOPS
ARITHMETIC LOGIC UNIT (ALU) (8)
INSTRUCTION REGISTER (8)
INSTRUCTION DECODER AND MACHINE CYCLE ENCODER
B REG. (8)
C REG. (8)
D REG. (8)
E REG. (8)
H REG. (8)
L REG. (8)
STACK POINTER (16)
PROGRAM COUNTER (16)
INCREMENT/ DECREMENT ADDRESS LATCH
ADDRESS BUFFER (8)
DATA/ADDRESS BUFFER (8)
A_{15}-A_8 ADDRESS BUS
AD_7-AD_0 ADDRESS/DATA BUS
POWER SUPPLY
+5V
GND
TIMING AND CONTROL
CLK GEN
X_1
X_2
CLK OUT
READY
CONTROL
RD
WR
ALE
STATUS
S_0
S_1
IO/M
DMA
HOLD
HLDA
RESET
RESET OUT
RESET IN

Fig. 5.1

5.1.1 Register Section

The 8085 microprocessor contains eight addressable 8-bit registers namely:

A (Accumulator) register
F Flag register (Flag flip-flops)
B register
C register
D register
E register
H register
L register

Out of these registers B, C, D, E, H and L registers are 8-bit general purpose registers. These registers can either be used as single register or a combination of two registers as 16 bit register pair. As discussed in SAP-III computers, the valid register pairs are B-C, D-E or H-L register pairs. The higher order byte of 16 bit data is stored in first register (B in B-C register pair), and low order byte in the second register (C in B-C register pair).

The H-L register pair can also be used for register indirect addressing; since this register pair can also function as data pointer.

The large number of general purpose registers gives more flexibility and ease in the programming. However, the general purpose registers are limited as registers occupy more space on the silicon chip.

Beside these general purpose registers, the 8085 has remaining two 8-bit registers Accumulator (A) and Flag (F) as special purpose registers and two 16 bit registers namely Program counter (PC) and stack pointer (SP).

Accumulator (A)

As discussed in preceding chapters, accumulator is a 8 bit buffer register extensively used in arithmetic, logic, load and store operations as well as in input / output instructions. All the arithmetic and logical operations are performed on the accumulator contents; i.e. one of the operand is always taken into the accumulator.

Flag (F) Register

It is an 8-bit register associated with the execution of instructions in the microprocessor. Out of the 8 bits of flag register, 5 bits contains significant information in terms of status flags. The five flags are:

(i) Sign flag (S)
(ii) Zero flag (Z)
(iii) Carry flag (CY)
(iv) Parity flag (P)
(v) Auxiliary Carry flag (AC)

All the flags except the Auxiliary carry (AC) flag have been discussed in SAP-III computers.

The bit positions reserved for these flags in the flag register (F) is shown in figure 5.2.

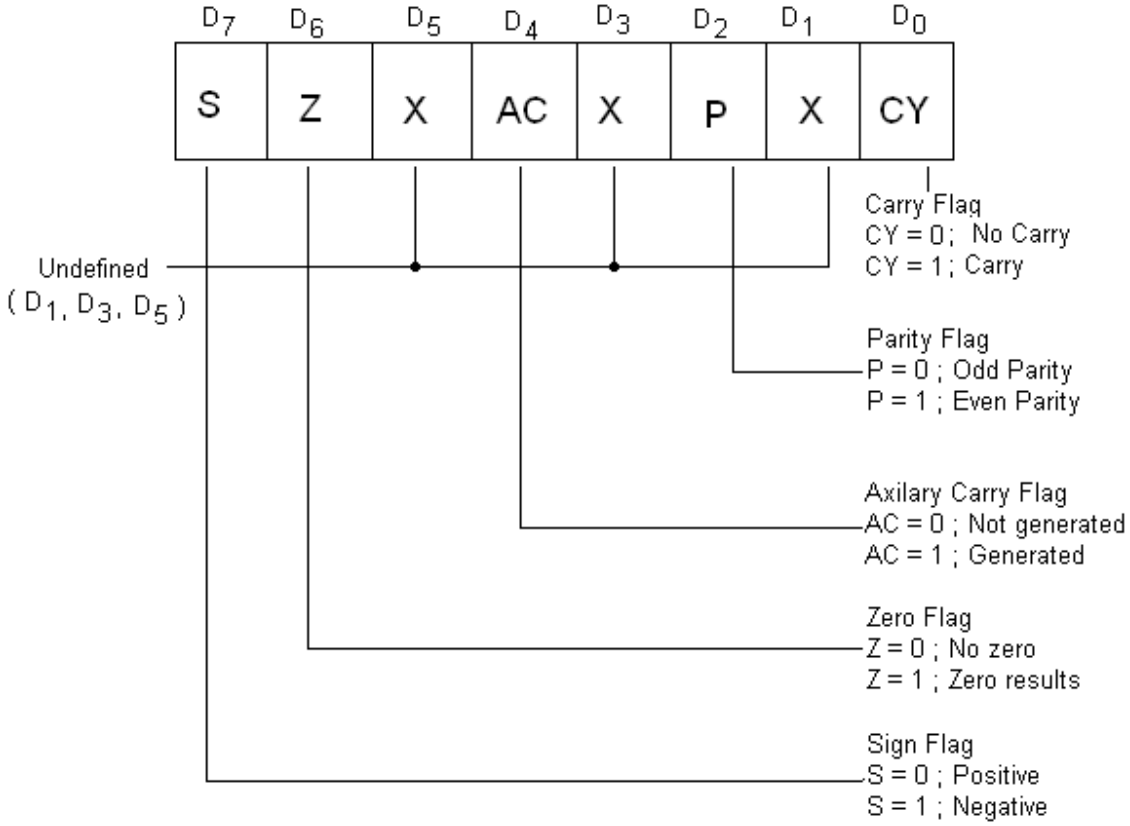


Fig. 5.2

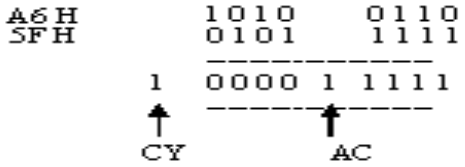
(i) Sign flag (S) The sign flag is set (S = 1), if the result of the operation of the instruction is negative (MSB of the result is 1); otherwise it is reset (S = 0) for the positive result (MSB is zero).

(ii) Zero flag (Z) The zero flag is set (Z = 1) if the result of the operation of the instruction is zero otherwise this flag is reset (Z = 0).
i.e. Z = 1 if the result is zero,
and Z = 0 if the result is not zero.

(iii) Carry flag (CY) The carry flag is set to 1, if there exist a carry (or borrow) to the highest order bit (non-existent 9^{th} position) as a result of the execution of addition or subtraction instructions. If there is no carry (or borrow) to the higher order bit, the carry flag is reset.
i.e. CY = 1 if there is a carry to the highest order bit (or overflow), and CY = 0 if there is no carry to the highest order bit (or no overflow).

(iv) Parity flag (P) After an arithmetic and logic operation, if the result has even number of 1s, then parity bit is set. If on the other hand the result has odd number of 1s, the parity flag is reset.
i.e. P = 1, if the result has even number of 1s,
and P = 0, if the result has odd number of 1s.

(v) Auxiliary carry (AC) This is a new flag in 8085 microprocessor. This flag (AC) is set to 1, if there is an overflow at bit 3 of the accumulator. AC flag is used in BCD arithmetic. This is illustrated as given below:



As shown in figure 5.2, the five bits in flag register are defined. The three bits are undefined. The Accumulator and 8 bits (including three undefined bits) of flag register form a Program Status Word (PSW). The accumulator and flag registers are treated as a 16 bit unit for stack operation.

Program Counter – 16 bit register

The program counter is a 16 bit register. It is used to send 16 bit address to fetch the instruction from the memory. It acts as a pointer which indicates the address of the next instruction to be fetched and executed. The program counter is updated after an instruction has been fetched by the processor. If an instruction is one byte instruction, then the program counter will be updated by one (i.e. PC = PC + 1). Similarly, for two and three byte instructions, the program counter will be updated by two (i.e. PC = PC + 2) or three (i.e. PC = PC + 3) locations respectively.

Stack Pointer – 16 bit register

The stack is an area of RAM (random access memory or read / write memory) in which temporary information is stored. It is stored on First-In-Last-Out (FILO) basis. An address in the RAM area is assigned to the stack pointer where the first information is stored as the first stack entry. This is done by initializing stack pointer by an instruction. Higher stack entries are made at the progressively decreasing addresses.

5.1.2 Address Buffer and Address-Data Buffer

It has already been discussed that 8085 requires 8-bit data bus and 16-bit address bus, as the memory address is of 16 bits. More number of IC pins are required if separate address and data bus are introduced. To restrict the number of pins of 8085 to only 40, lower address lines A_0-A_7 and data lines D_0-D_7 are used in multiplexed mode. The multiplexed lines are designed as Address/Data Bus (AD_0-AD_7). So whenever 16-bit address is transmitted by the microprocessor 8 MSBs of the address lines are sent on the Address Bus (A_{15}-A_8) and 8 LSBs of the lines are sent on the Address/Data Bus (AD_7-AD_0). The 8 LSBs of the address are then latched either into memory or external latch so that the complete address remains available for further operation. The 8-bit Address/Data Bus will now be free for the data transmission.

5.1.3 Arithmetic and Logical Unit (ALU)

The arithmetic and logical unit (ALU) basically consists of accumulator (A), flag register (F) and a temporary register (which is inaccessible by the programmer or user). This unit carries out the arithmetic and logic calculations of the data stored in general purpose registers or in memory locations. The arithmetic operations are ADD, SUB, compare, increments, decrements and complements etc.; while logical operations are AND, OR, XOR and Rotate. The result of these operations could be placed in the accumulator or elsewhere through the internal bus. Arithmetic operations are usually carried out in 2's complement adder / subtrator discussed in the preceding chapter. For these operations, ALU receives the control signals from the timing and control unit.

5.1.4 Timing and Control Unit

This unit consists of the following sections:

1. Instruction Register and Decoder
2. Control signals

1. Instruction Register and Decoder

As discussed in the preceding chapter, the CPU fetches an instruction from the memory for its execution. This instruction can be of 1-3 byte long. The first byte contains the op code of instruction which basically specifies the nature of operation to be performed indicating the length of the instruction. The first byte (op code of the instruction) transferred to 8-bit instruction register through the internal bus of the CPU, becomes available at the instruction decoder. The decoder decodes the op code and directs the control unit to produce the necessary control signals.

2. Control Signals

Following are the control signals of 8085 microprocessor needed for the operation of CPU.

(i) X_1, X_2 and CLK Out

Two pins X_1 and X_2 are provided to be externally connected to a quartz crystal. The clock signal of fixed frequency is generated through the internal circuitry of the processor. The frequency at which the microprocessor 8085 works is half of the crystal frequency. The quartz crystal of 6.144 MHz is used in this processor. This gives the clock frequency of 3.072 MHz (half of the crystal frequency) of 50% duty cycle. The clock period is of about 320 nsec. The output of the clock frequency is also available at CLK out terminal.

(ii) Address Latch Enable (ALE)

The 16 bit address bus is basically divided into two sets. The most significant bits A_7-A_{15} of the address bus are used separately and the least significant bits of the address AD_0-AD_7 are time multiplexed with the bits of bidirectional data bus (D_0-D_7). The AD_0-AD_7 bus serves the dual purpose as they can be used as low-order address bus as well as bidirectional data bus at different times. This is used as address bus, during the first clock cycle of the machine cycle involving memory; and during the remaining clock cycle of the machine cycle, it acts as the data bus. This is accomplished by address latch enable (ALE) signal provided in the processor. During the first clock cycle of the machine cycle ALE is high which enables the lower 8-bit of the address to be latched either into the memory or external latch.

(iii) $\overline{R_D}$ (Read) Signal

This is an active low signal to be connected to memory read input (output enable signal to memories) or to input / output read signal to enable input / output buffer.

(iv) $\overline{W_R}$ (Write) Signal

Similar to read signal ($\overline{R_D}$), write signal ($\overline{W_R}$) is also active low. This signal is used to write to the memory or input / output devices.

(v) *IO/* $\overline{M}$ (Input Output / Memory)

This signal IO/$\overline{M}$ distinguishes that the address and data is meant for either I/O devices or memory. Whenever this signal is high (1), microprocessor will communicate to the I/O devices and whenever it is low (0), microprocessor will communicate to the memory.

(vi) Status Signals (S_0, S_1)

The status signals (S_0, S_1) along with IO/$\overline{M}$ signal indicate the type of machine cycle in progress. The type of machine cycle are op code fetch cycle, memory read cycle, memory write cycle, I/O read cycle or I/O write cycle.

Various types of status codes are given in table 5.2.

Table 5.2

Machine Cycle	IO/$\overline{M}$	S_1	S_0
Op code fetch Cycle	0	1	1
Memory Read Cycle	0	1	0
Memory Write Cycle	0	0	1
I/O Read Cycle	1	1	0
I/O Write Cycle	1	0	1
INTR Acknowledge	1	1	1
Halt	Hi-Z	0	0

(vi) Hold and HLDA

HOLD and HLDA (Hold Acknowledge) signals are used for DMA (Direct Memory Access) operation. In a microprocessor, the data transfer between I/O devices and memory will take place through the microprocessor. The involvement of the processor slows down the data transfer between I/O devices and memory. The transfer of data directly from I/O devices to memory without involvement of microprocessor is called DMA. The DMA will save the time as CPU relinquishes the control of Buses. In this way DMA transfers the large amount of data in a relatively short time. The HOLD and HLDA signals are used in the operation. Whenever HOLD signal is high, CPU temporarily relinquishes its operation by floating the address, data and control buses; and DMA operation is started. A high HLDA (Hold Acknowledge) signal is also sent to DMA controller, indicating that CPU has received the hold request. Whenever the data transfer is complete, then the control to CPU is returned back by sending a HOLD signal. Further the HLDA signal goes low.

(viii) READY signal (Input)

Some peripheral devices connected to 8085 microprocessor operate at much slower speed than the processor. To synchronize the speed of CPU and peripheral devices or to slow down the speed of 8085, the READY signal is used. If the READY signal is high the peripheral device is ready and the processor can complete the data transfer. If

this signal is low the microprocessor waits (by generating a number of NOP T-states) till it goes high.

(ix) $\overline{\text{RESET IN}}$ and RESET OUT

The signal $\overline{\text{RESET IN}}$ may be low from the operator Reset button or from the processor. When the signal is low, the CPU will reset the program counter, instruction register and other circuits. It also sends a high **RESET OUT**. The **RESET OUT** signal goes to peripheral devices to reset or initialized. When $\overline{\text{RESET IN}}$ signal goes high and **RESET OUT** goes low, the data processing may begin.

5.1.5 Interrupt Control

Sometimes it is necessary to interrupt the execution of the main program. For this an interrupt request is obtained from the I/O devices. After receiving the interrupt request (*INTR*), processor temporarily stops what it was doing and attends to the I/O device. $\overline{INTA}$ is an interrupt acknowledge signal which is sent by the microprocessor after INTR signal is received. After the work of the I/O device is complete it returns to what it was doing earlier.

Basically 8085A has five hardware interrupts namely:

INTR
RST 5.5
RST 6.5
RST 7.5
and **TRAP**

If two or more of these interrupts are active at the same time, the 8085 takes them in order of priority level. The priority levels of these interrupts are given in table 5.3.

Table 5.3

Interrupts	Priority
TRAP	**1**
RST 7.5	**2**
RST 6.5	**3**
RST 5.5	**4**
INTR	**5**

The details of these instructions will be discussed in chapter 7.

5.1.6 Serial I/O Control

Serial input / output control circuit incorporated in this microprocessor is used for the data transmission. For this purpose two pins SID and SOD are provided in the serial input/output control unit. The SID (Serial Input Data) terminal receives the serial data stream from an input device, the control unit converts serial data stream to parallel data before it is used by the computer. After the conversion 8-bit parallel data is stored in the accumulator. Similarly, SOD (Serial Out Data) terminal outputs the 8-bit parallel available with the accumulator into serial form to the peripheral device connected with the computer.

5.2 PIN DESCRIPTION OF 8085

The pin details and logical schematics of the 40 pin dual line package (DIP) IC 8085 are shown in figures 5.3(a) and (b) respectively. Fig. 5.3 (c) shows the shape of the microprocessor. The descriptions of various pins of the microprocessor are given below:

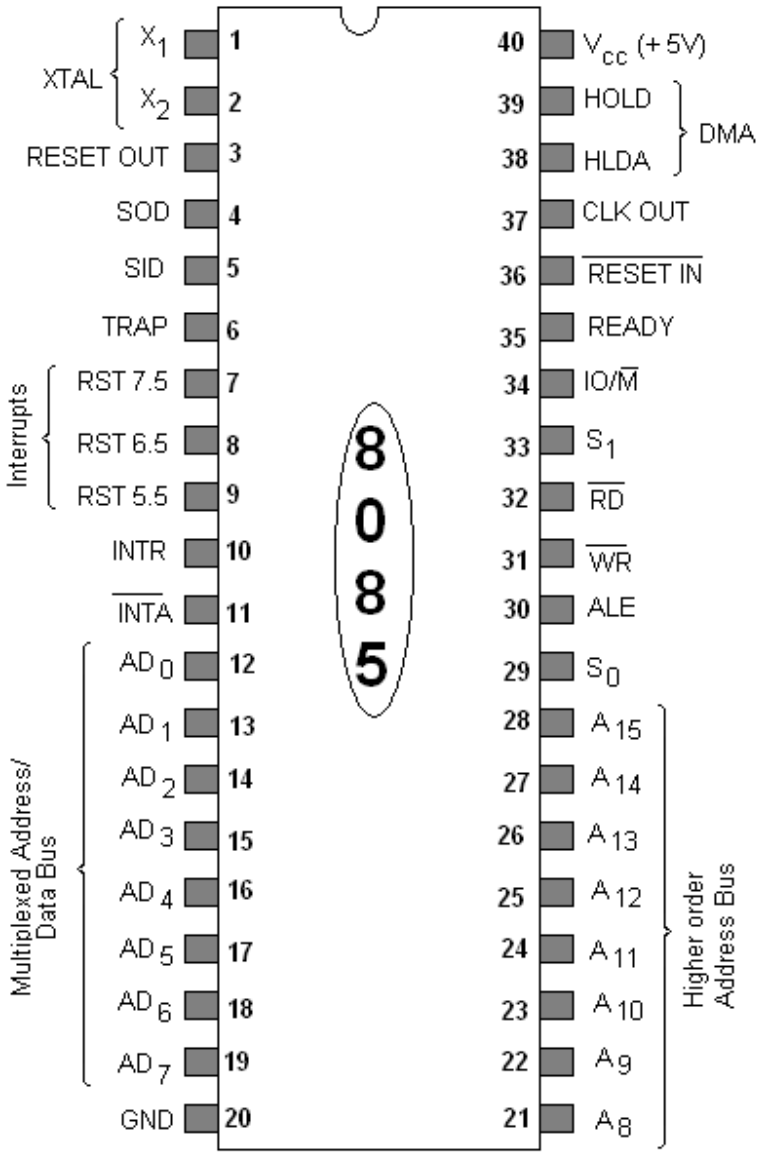


Fig. 5.3 (a)

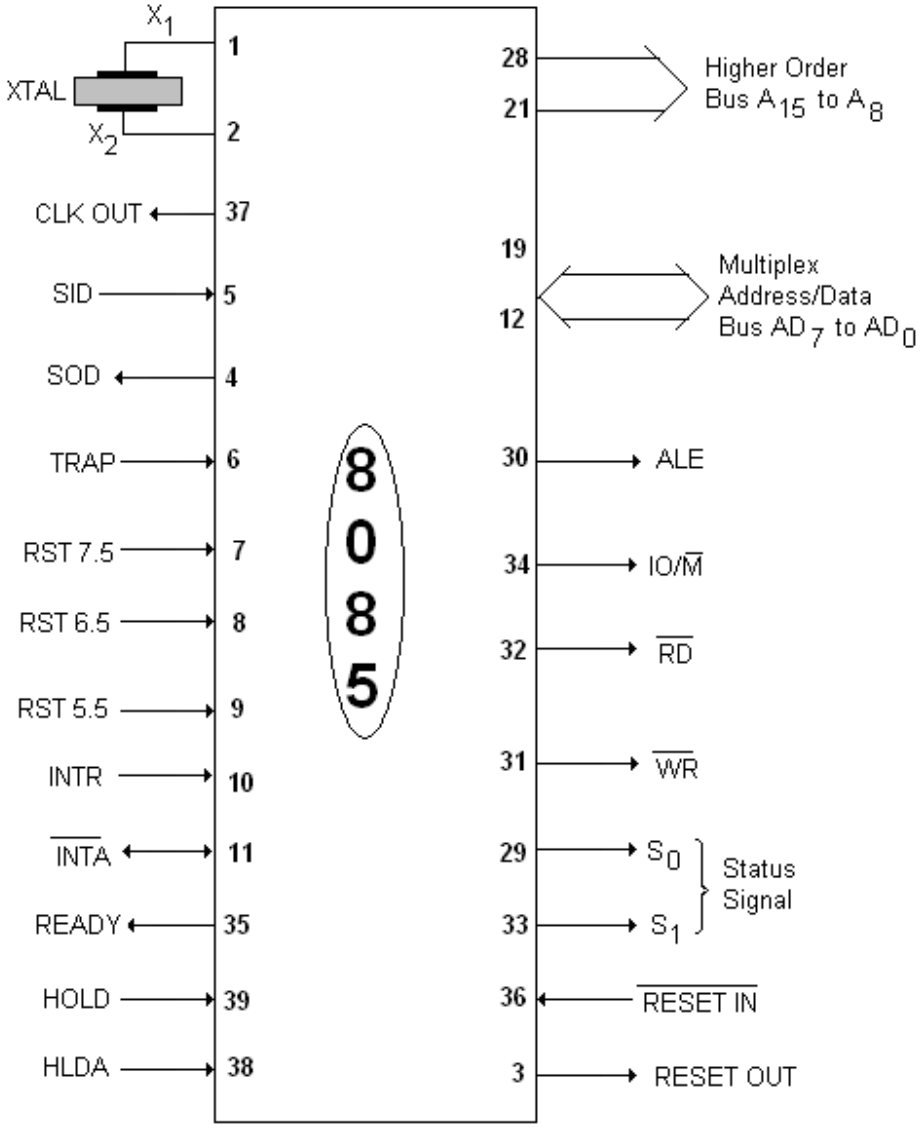


Fig. 5.3 (b)

Fig. 5.3 (c)

PIN NOS. 1 and 2:

These X_1 and X_2 pins are to be connected to an external quartz crystal, L-C or R-C network which drives the internal clock generator. The clock signal of appropriate frequency is determined when a quartz crystal is connected to the on-chip oscillator as shown in figure 5.4(a). The oscillator output from the Schmitt trigger drives a flip-flop which divides the frequency by a factor of two. The circuit produces two clock signals Φ1 (CLK) and Φ2 ($\overline{CLK}$) to derive the internal circuit of the microprocessor. A 6.25 MHz crystal is used to provide 3.125 MHz internal clock frequency.

Generally, quartz crystal is used for the On-chip oscillator for the accurate and stable clock frequency, though a parallel resonant L-C circuit may be used for the frequency determining network as shown in figure 5.4(b). The network produces a signal whose frequency tolerance is about $\pm 10\%$. The component values may be chosen from the following formula:

$$f = \frac{1}{2\pi\sqrt{L(C + C_{in})}}$$

The input capacitance Cin is approximately 15 Pf. To minimize the variations in frequency, it is recommended to choose C as 30 Pf.

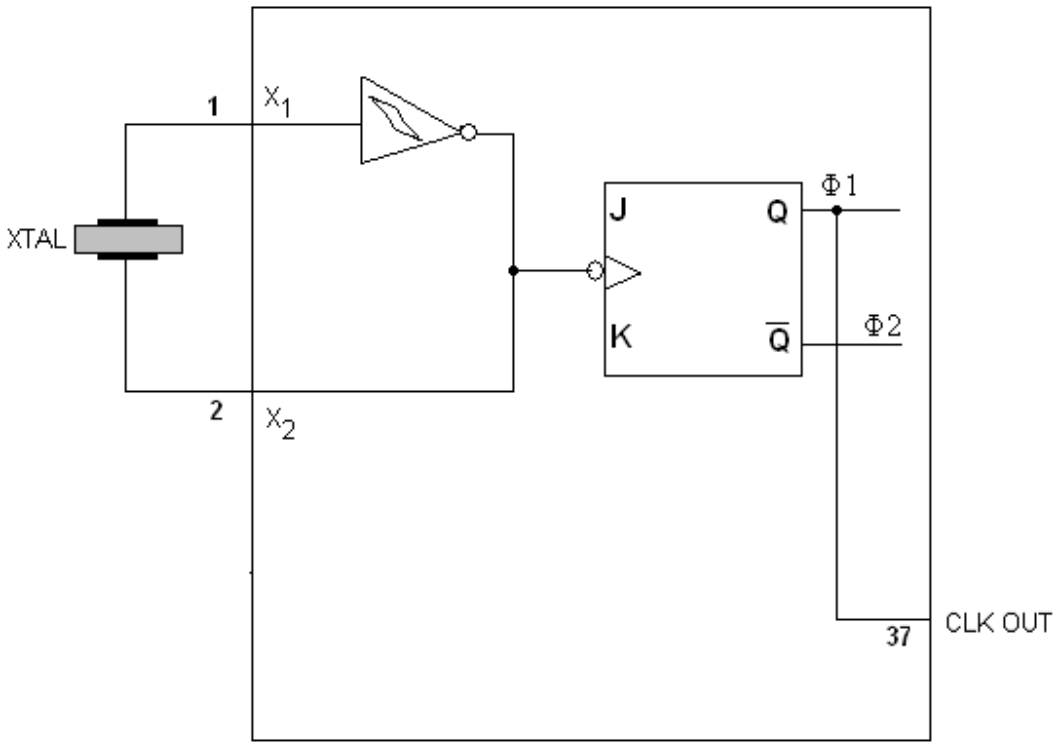
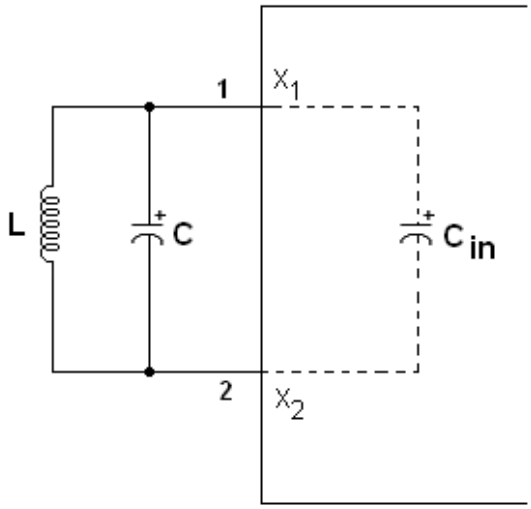


Fig. 5.4(a)



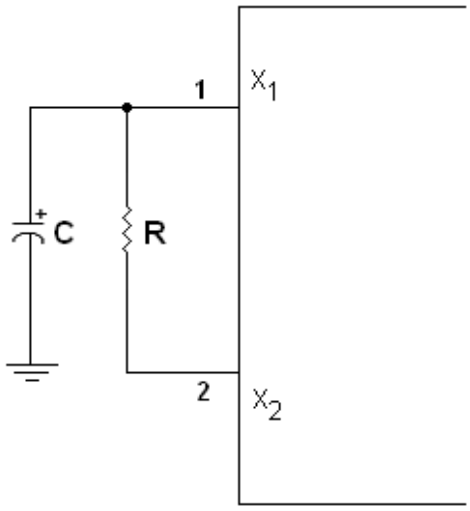


Fig. 5.4 (b) **Fig. 5.4 (c)**

An R-C network may also be used as the frequency determining network for the on-chip oscillator of the microprocessor as shown in figure 5.4(c). The driving frequency generated by this circuit is approximately 3MHz. It is not recommended to use the frequencies higher or lower than this.

PIN NO. 3

This is **RESET OUT** signal, which indicates that CPU is being reset. When it is high, system is reset. The signal is synchronized to the processor clock and lasts for an integral number for clock periods. When the **RESET OUT** signal goes low, the processing begins.

PIN NOS. 4 and 5

Pin Nos. 4 and 5 indicate SOD (Serial Out Data) and SID (Serial In Data) terminals respectively. These pins are associated with Serial Input/Output control unit for 8085 microprocessor. As already discussed these pins are used for the serial data transmission. The SOD output pin can deliver a serial data stream to a peripheral device. On executing SIM (Set Interrupt Mask) instruction, if bit D_6 is set to 1, the content of D_7 bit (set or reset) of the accumulator is latched on the SOD pin as shown in figure 5.5(a).

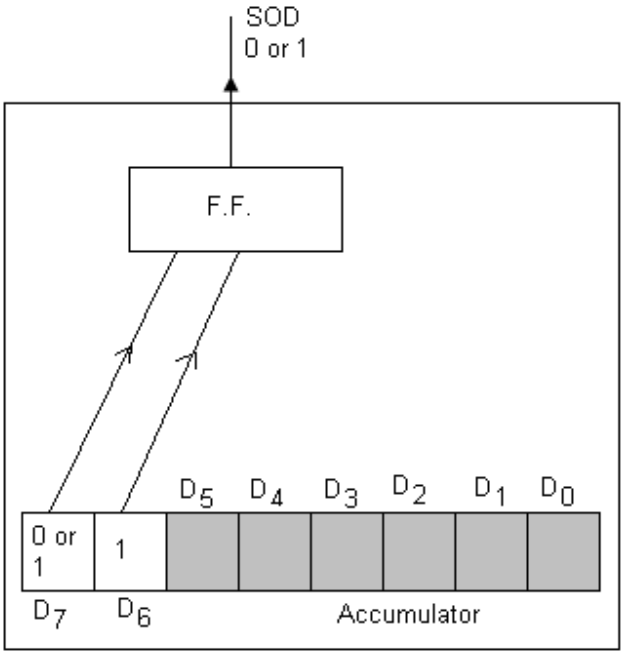


Fig. 5.5 (a)

The data on the SID line (PIN 5) loads into accumulator at bit D_7 whenever a RIM instruction is executed as shown in figure 5.5(b).

The details of SIM and RIM instructions will be discussed in chapter 7.

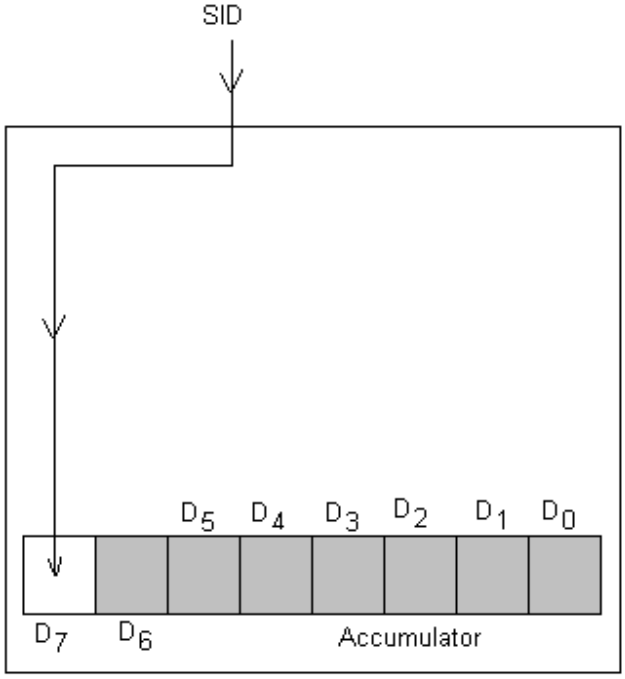


Fig. 5.5 (b)

PIN NOS. 6 to 11

The interrupt control unit of the microprocessor contains these pins. The Pins 6 to 11 are restart interrupts named as:

TRAP (Pin No.6)	I Priority
RST 7.5 (Pin No. 7)	II Priority
RST 6.5 (Pin No. 8)	III Priority
RST 5.5 (Pin No. 9)	IV Priority
INTR (Pin No. 10)	V Priority

The TRAP has the highest priority and INTR has the lowest priority. The priority level is of importance if two or more interrupts become active at the same time. The TRAP is non-maskable interrupt. It is both edge and level sensitive.

The interrupts (TRAP, RST 7.5, RST 6.5 and RST 5.5) are also called vector interrupts, as each interrupt has fixed memory location (vector location) for the transfer of control from the normal execution of the routine. The vector locations of these interrupts are given in table 5.4. As soon as any of these pins 6 to 10 are active (high), the internal circuit of 8085 stops the normal execution of program and the program control is transferred to the corresponding memory location (vector location).

Table 5.4

Interrupts	Memory locations
TRAP	**0024 H**
RST 7.5	**003C H**
RST 6.5	**0034 H**
RST 5.5	**002C H**

INTR (Pin No. 10) is a general purpose interrupt and has the lowest priority. As soon as Pin No. 10 is high, the microprocessor stops the execution of normal program and after completing the instruction at hand, it goes to CALL instruction. The INTR is enabled or disabled by the instructions ET (Enable Interrupts) or DI (Disable Interrupts) respectively.

The Pin No. 11 is an Interrupt Acknowledge ($\overline{\text{INTA}}$) signal. A low (logic 0) to this pin indicate that the microprocessor has acknowledged the request from the peripheral device. It is also used to activate the interrupt controller.

PIN NOS. 12 to 19

Pin Nos. 12 to 19 (AD_0-AD_7) form bi-directional multiplexed Address/Data Bus. The least significant 8 bits of the memory address (or I/O Address) appear on the bus during the first T-states of a machine cycle. It then becomes the data bus during the next T-states.

PIN NO. 20

Pin No. 20 is the ground terminal.

PIN NOS. 21 to 28

The Pin Nos. 21 to 28 (A8-A15) form unidirectional most significant 8 bits of memory address or 8 bits of the I/O address. It remains in the high impedance state during HOLD, HALT and RESET modes.

PIN NOS. 29 to 33

The Pin Nos. 29 to 33 labled as S_0 and S_1 respectively are known as status signals. These status signals along with $IO/\overline{M}$ signal indicate the various operations as indicated in table 5.5.

Table 5.5

Machine cycle	**$IO/\overline{M}$**	**Status**		**Control signals**
		S_1	**S_0**	
Op code Fetch	**0**	**1**	**1**	$\overline{RD} = 0$
Memory Read	**0**	**1**	**0**	$\overline{RD} = 0$
Memory Write	**0**	**0**	**1**	$\overline{WR} = 0$
I/O Read	**1**	**1**	**0**	$\overline{RD} = 0$
I/O Write	**1**	**0**	**1**	$\overline{WR} = 0$
Interrupt Ack.	**1**	**1**	**1**	$\overline{INTA} = 0$
HALT	**HI-Z**	**0**	**0**	
HOLD	**HI-Z**	**X**	**X**	$\overline{RD}, \overline{WR} = Z$
RESET	**HI-Z**	**X**	**X**	$\overline{INTA} = 1$

HI-Z = High Impedance State
X = Unspecified

PIN NO. 30

The Pin No. 30 is known as ALE (Address Latch Enable) terminal. When this signal is high the information carried on the multiplexed address/data bus (AD_0-AD_7) is the lower 8 bits of the address. It also enables the low order address (AD_0-AD_7) from the multiplexed address/data bus to latch either into the memory or the external latch. The ALE signal separates the low order address and data from the multiplexed Address/data Bus. This is illustrated in figure 5.6.

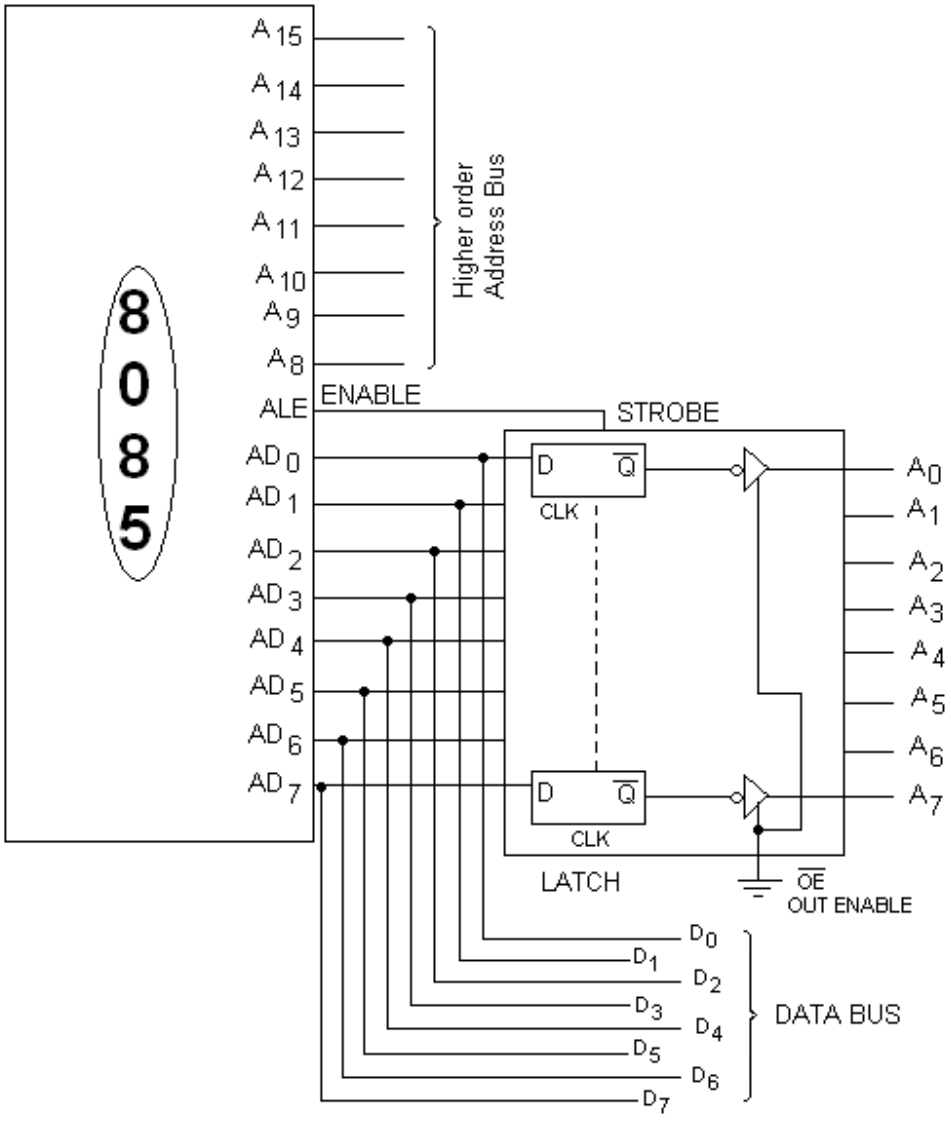


Fig. 5.6

PIN NOS. 31, 32 and 34

The Pin Nos. 31 and 32 are the two control signals $\overline{WR}$ (Write bar) and $\overline{RD}$ (Read bar) respectively. The pin 34 carries $IO/\overline{M}$ signal which is one of the status signals. The other status signals are S0 and S1 discussed earlier. A low $\overline{WR}$ signal generated by the microprocessor sends (writes) data into I/O devices or memory. Similarly, a low $\overline{RD}$ signal generated by the microprocessor reads (receives) the data from the I/O devices or memory locations. The $IO/\overline{M}$ signal indicates whether the address on the address bus is meant for I/O devices. However, a low to this signal indicates that the address on the address bus is meant for memory location. The $\overline{RD}$, $\overline{WR}$ and $IO/\overline{M}$ signals function together.

PIN NO. 35

The Pin No. 35 is known as READ signal which forces the microprocessor to wait till the data become available from the memory or input/output devices. This signal is needed to synchronize the speed of the microprocessor with I/O devices or memory as the memory or I/O devices are not as fast as the microprocessor. When the READ signal is low, the microprocessor waits till the READY signal is 1. As soon as READY signal is 1, the microprocessor knows that the data are available from the memory or I/O devices.

PIN NO. 36

This pin is $\overline{\text{RESET IN}}$ signal. This input carrying signal may be operated by the operator using the RESET button provided externally or it may be operated directly from the other source. When this signal is low (momentarily), the CPU will reset the program counter, instruction register, all interrupts (except TRAP) are disabled, SOD signal becomes low and Data, address and control buses are floated. When this signal goes high, the data processing begins.

PIN NO. 37

This pin carries CLK OUT signal. It is derived from the on-chip oscillator, which goes to peripherals to synchronize their timings.

PIN NOS. 38-39

The Pin Nos. 38 and 39 are the HOLD and HLDA (Hold Acknowledge) signals respectively. These signals are used in DMA (Direct Memory Access) operations. As shown in figure 5.7, when any I/O device indicates that the data are ready for DMA transfer, a high HOLD signal is sent by the DMA controller to the 8085 microprocessor. It is in fact a request signal from the DMA controller to the microprocessor. The microprocessor then sends a high signal to DMA controller indicating that the microprocessor has received the request from the I/O devices and will relinquish the address, data and control bus after completing the current instruction. The DMA controller thus carries out the data transfer. A low HOLD signal will return the control to the microprocessor.

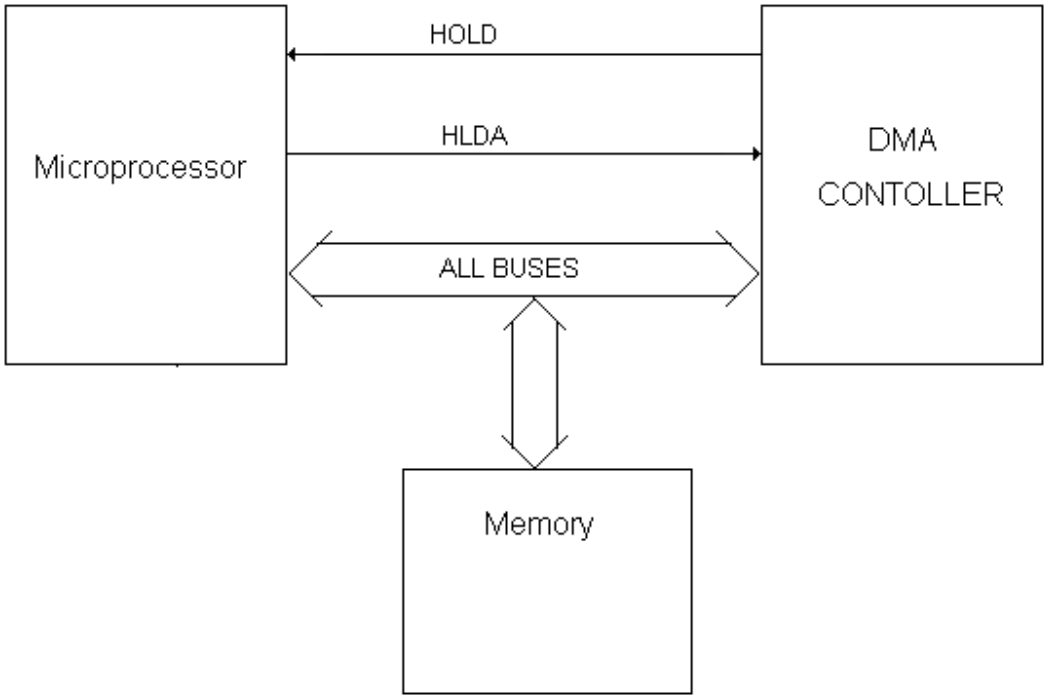


Fig. 5.7

PIN NO. 40

The pin 40 is +VCC, which is to be externally connected to +5 volt d.c. supply.

5.3 INSTRUCTION SET OF 8085 MICROPROCESSOR

The 8085 includes all the instructions of SAP-III. In addition there are few more instructions which will be discussed below. These new instructions were not considered in SAP-3 because of its architecture.

LHLD address **(Loads the H-L pair direct)**

This instruction loads the H-L pair direct with two bytes already stored in two consecutive memory locations starting at the specified memory address. The contents stored in the memory location whose address is given with the instruction will be loaded to the L-register; and the contents stored in the next memory location (address + 1) will be loaded to the H-register.

i.e. $[L] \leftarrow [M_{address}]$

and $[H] \leftarrow [M_{address+1}]$

For example, let 2A H is stored in the memory location 2100 H and 2B H is stored in the memory location 2101 H, then after the execution of the instruction LHLD 2100 H , the L-register will have 2A H and H-register will have 2B H.

None of the flags is affected with this instruction.

SHLD address **(Stores the H-L pair direct)**

This instruction does the reverse operation of LHLD. The instruction *SHLD address* stores the contents of L-register to memory location whose address is given with the instruction; and the contents of H-register are stored in the next consecutive memory location (address + 1).

i.e. $[M_{address}] \leftarrow [L]$

and $[M_{address+1}] \leftarrow [H]$

No flag is affected with this instruction too.

For example, if $[L] = 3A$ H and $[H] = 3B$ H, then after the execution of the instruction SHLD 2200 H will result.

$[M_{2200H}] \leftarrow 3A$

and $[M_{2201H}] \leftarrow 3B$

LDAX rp **(Loads the Accumulator Indirect)**

This instruction loads the accumulator, the contents already stored in the memory location addressed by the register pair (rp). Here rp represents B-C or D-E register pair. The H-L register pair is not included in this instruction.

i.e. $[A] \leftarrow [M_{rp}]$

The possible combinations of the instruction are:

LDAX B

LDAX D

No flag is affected with the execution of this instruction.

For example, if [D] = 25 H, [E] = 00 H

and $M_{2500\ H}$ = 34 H,

then after the execution of the instruction LDAX D, the accumulator will have:

$[A] \leftarrow [M_{2500H}]$

i.e. A = 34 H

It is worth while to mention that the instruction LDAX H does not exist, because the contents stored in the memory location addressed by H-L register pair may be loaded to accumulator by the instruction *MOV A, M*.

STAX rp **(Stores the Accumulator Indirect)**

The *STAX rp* instruction does the reverse operation of *LDAX rp*. This instruction stores the accumulator contents in the memory location addressed by the register pair (rp). Here too rp represents B-C or D-E register pair. The H-L register pair is not included in this instruction.

i.e. $[M_{rp}] \leftarrow [A]$

The possible combination of this instruction are:

STAX B

STAX D

No flag is affected with the execution of this instruction.

For example, if B = 21 H, C = 00 H

and A = 3A H,

then after the execution of the instruction STAX B, 3A H will be stored in the memory location 2100 H.

i.e. $[M_{2100H}] = 3AH$

The combination *STAX H* is not included in this instruction as *MOV A, M* performs the same operation.

XCHG **(Exchange the contents of H-L register with D-E register)**

This is one byte instruction and no operand is needed with it. It exchanges the contents of H and L register with D and E registers respectively.

i.e. $[\text{H}] \leftrightarrow [\text{D}]$ and $[L] \leftrightarrow [E]$

For example:

If H = 25 H , L = 32 H

and D = 12 H, E = 1B H

then after the execution of *XCHG* instruction, we have:

H = 12 H, L = 1B H

and D = 25 H, E = 32 H

The instruction *XCHG* is generally used to keep track of more than one memory location at a time without using LDAX and STAX instructions.

Let us write a program to add two numbers stored in memory locations 2100 H and 2201 H without using *LDAX* and *STAX* instructions. The answer is to be loaded in the memory location 2100 H.

```
LXI H, 2201 H  ; Loads H = 22 H and L = 01 H
LXI D, 2100 H  ; Loads D = 21 H and E = 00 H
MOV A, M       ; [A] ← [M2201H]
XCHG           ; H = 21 H, L = 00 H and D = 22 H, E = 01 H
ADD M          ; [A] ← [A] + [M2100H]
MOV M, A       ; [M2100H] ← [A]
HLT
```

In this program no LDAX and STAX instructions are used.

DAA **(Decimal Adjust the Accumulator)**

The *DAA* is one byte instruction and no operand is needed with this instruction. It adjusts the accumulator to packed BCD (Binary Coded Decimal) after addition of two BCDs. In other words, after addition of two hexadecimal numbers if this instruction is used then the result in decimal form is obtained. For this Auxiliary Carry Flag (AC) and Carry Flag (CY) take care of this instruction.

It functions in two steps:

1. If the lower nibble (lower 4-bits) of the accumulator is greater than 9 or Auxiliary carry flag is set, then it adds 06 H to the accumulator.
2. Subsequently, if the higher nibble (higher 4-bits) of the accumulator is now greater than 9 or the carry flag (CY) is set, it adds 60 H to the accumulator.

All the flags are affected with this instruction.

Example 5.1 *What will be the value of accumulator, CY and AC flags after the execution of the following program:*

```
MVI A, 38 H
ADI 87 H
DAA
HLT
```

Solution.

```
A = 38 H      0 0 1 1  1 0 0 0
Adds 87 H     1 0 0 0  0 1 1 1
              ---------------------
```

```
  1 0 1 1  1 1 1 1
  0 0 0 0  0 1 1 0      Lower nibble is more than 9
-----------------------
  1 1 0 0  0 1 0 1      AC = 1
  0 1 1 0  0 0 0 0      Upper nibble is more than 9
-----------------------
1 0 0 1 0  0 1 0 1
                        Result = 125 (Decimal)
```

A = 0 0 1 0 0 1 0 1 CY = 1 AC = 1

PCHL **(Copies H-L to PC)**

This is one byte instruction and no operand is needed with this instruction. It copies the contents of H-register to high-order byte of the program counter (PC) and the contents of L-register to low order byte of the program counter.

$[PC] \leftarrow [HL]$ i.e. $[PCH] \leftarrow [H]$ and $[PCL] \leftarrow [L]$

For example if PC = 2106 H and HL = 2500 H

then after the execution of the instruction *PCHL* will result:

PC = 2500 H

No flag is affected with the instruction.

This instruction is basically an unconditional jump instruction as is evident from the above example.

SPHL **(Copies HL to Stack Pointer SP)**

This is also one byte instruction as no operand is used. The *SPHL* instruction copies the contents of H-register to high order byte stack pointer (SP) and the contents of L-register to low order byte of stack pointer (SP).

$[SP] \leftarrow [HL]$ i.e. $[SPH] \leftarrow [H]$ and $[SPL] \leftarrow [L]$

None of the flags is affected with this instruction.

For example if H = 23 H and L = 45 H

and SP = 2501 H

then after the execution of the instruction *SPHL* will result:

SP = 2501 H

This is another way of initialization the stack pointer.

XTHL ***(*****Exchanges the top of the stack with H-L register pair)**

The *XTHL* is one byte instruction and does not require any operand. The top byte of the stack is exchanged with L-register and next byte of the stack is exchanged with H-register.

i.e. $[L] \leftrightarrow [M_{SP}]$

and $[H] \leftrightarrow [M_{SP+1}]$

For example if H = 21 H L = 02 H

and M_{SP} =1A H M_{SP+1} = 2C H

as shown in figure 5.8(a), then after the execution of the instruction *XTHL* will result:

H = 2C H L = 1A H

and M_{SP} = 02 H M_{SP+1} = 21 H

as shown in figure 5. 8(b).

No flag is affected with the instruction.

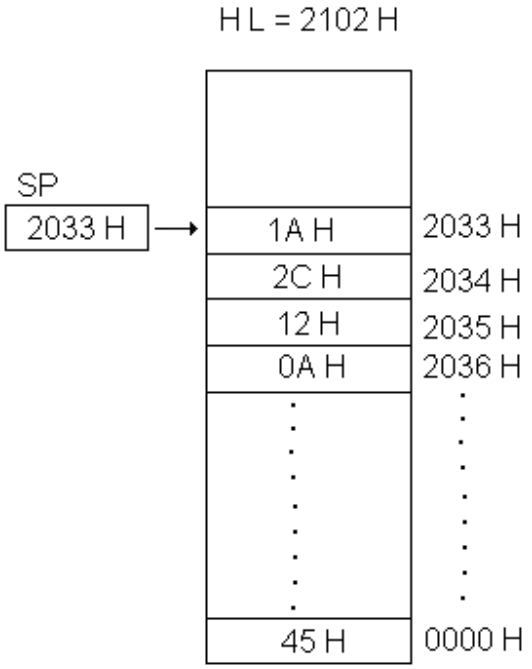


Fig. 5.8 (a)

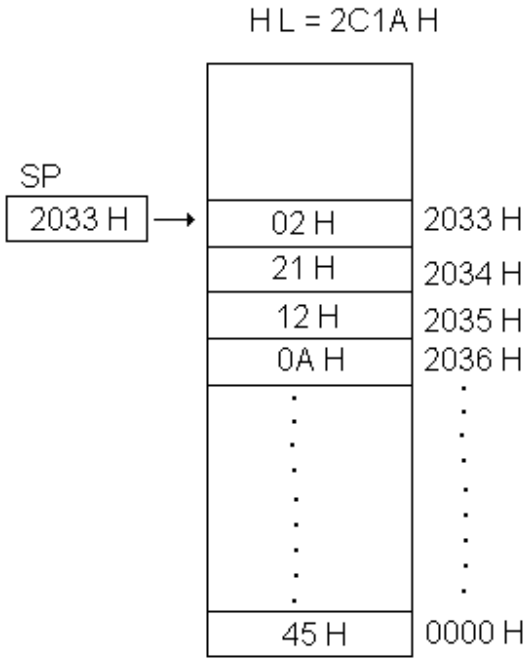


Fig. 5.8 (b)

5.4 TIMING DIAGRAM FOR 8085 INSTRUCTIONS

The working of 8085 microprocessor can best be understood by considering the timing diagrams of its few instructions. The graphical representation depicting the necessary steps carried out in a machine cycle is known as Timing Diagram. As is well known that the total time required for the execution of an instruction is the time required to fetch and execute an instruction.

i.e. **Instruction Cycle = Instruction fetch + Instruction execution**

The instruction cycle may be of one, two or three bytes. During the fetch cycle, an instruction of the program (op code) is extracted from the memory locations and copied in the instruction register (IR) of C.P.U.

The op code of the instruction copied in the instruction register (IR) is decoded during the execution cycle to perform the specific activities.

It may further be mentioned that an instruction cycle may take one to five machine cycles. Generally, first machine cycle known as op code fetch cycle, has either four or six T-states and the remaining machine cycles called execution cycles, have two or three T-states. A T-state represents one clock cycle.

The NOP is the shortest instruction which takes only one machine cycle with four T-states. However, the CALL is the longest instruction which takes five machine cycles with 18 T-states.

Here the timing diagrams of a few instructions will be discussed.

5.4.1 Timing Diagram of *MOV reg, M*

The timing diagram of the instruction

MOV reg, M

is shown in figure 5.9.

This is an indirect read instruction and takes two memory cycles (M_1 and M_2). The first machine cycle (M_1) is known as instruction fetch cycle, during which the op code of the instruction is fetched from the memory. This machine cycle takes four T-states. The second machine cycle M_2 is known as the execution cycle during which the data from the memory location addressed by H-L register pair is transferred to the given register. The second machine takes three states.

During the first T-state (T_1) of M_1 cycle, microprocessor sends the address of the memory location where the op code of the instruction ***MOV reg, M*** is stored, to the address lines. The high order byte of the PC (PCH) is placed on A_8-A_{15} lines and it stays on till T_4. The low order byte of PC (PCL) is placed on the address data lines (AD_0-AD_7) which stays on only during T_1-state. For this purpose ALE (Address Latch Enable) signal gives a positive pulse midway through first T-state (T_1), which latches the low order byte of the address into the memory chips. The $IO/\overline{M}$ signal goes low at the beginning of T_1 state; this enables the peripheral chips for a memory operation rather than input/output operation. It is customary to represent the address lines by double sided waveforms as the address bits may be high or low (ref. fig. 5.9).

MOV reg, M

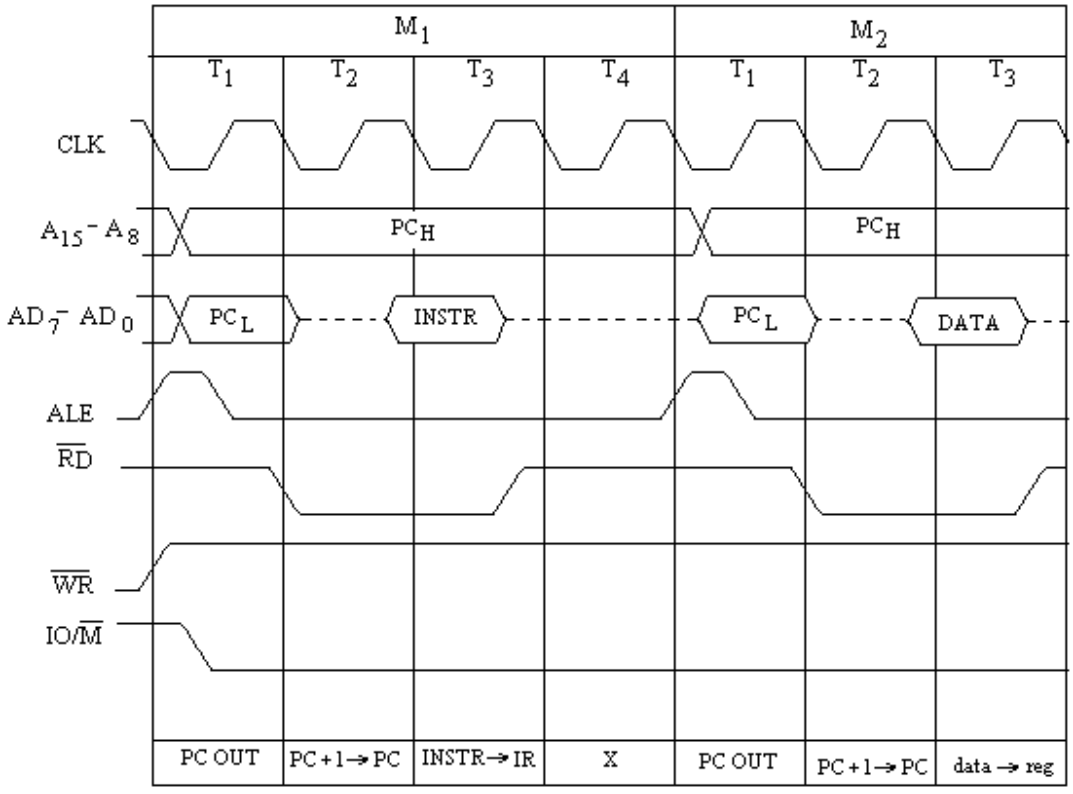


Fig. 5.9

During the second T-state (T_2) of this op code fetch cycle, program counter (PC) is incremented (PC = PC + 1). The address disappears from the address data bus (AD_7-AD_0) at the beginning of T_2 state. This is shown by dashed line indicating the data on the

bus is invalid or meaningless. At the beginning of T_2 state $\overline{RD}$ signal goes low and remains low till the middle of the T_3 state.

During the third T-state (T_3) of M_1 machine cycle, the op code of the instruction is read out from the memory which is sent to the instruction register (IR) i.e. $INSTR \rightarrow IR$.

The fourth T-state (T_4) of M_1 machine cycle is denoted by **X** which indicates that this T-state is needed for the instruction decoding and other internal operations before the execution cycle.

Now the second machine cycle M_2 known as execution cycle starts. During the first T-state of this execution cycle the contents of H-L register pair are placed on the address bus and address data bus. Basically, it performs the same operation as the T_1-state of M_1 cycle. During T_2 and T_3 states of M_2 machine cycle the data is read from the memory and copied in the specified register.

This completes the instruction ***MOV reg, M***. It may be noted that this instruction needs two machine cycles with 7 T-states.

5.4.2 Timing Diagram of *MOV M, reg*

This is an indirect write instruction. The timing diagram of MOV M, reg instruction is shown in figure 5.10. The first three states of memory fetch cycle M_1 are the same as for ***MOV reg, M***. During fourth T-state (T_4) of M_1, the contents of specified register are copied in the temporary register.

During the T_1-state of second machine cycle M_2, the contents of H-L pair are placed on the address and address data bus. As usual during this state ALE sends a high pulse. During T_2 and T_3 states of machine cycle M_2, contents of temporary register are transferred (copied) to the specified address. Since it is memory write instruction, so at the beginning of T_2-state $\overline{WR}$ signal activates (becomes low) and it remains low till half way through its T_3-state.

This instruction also takes two machine cycles with 7 T-states.

MOV M, reg

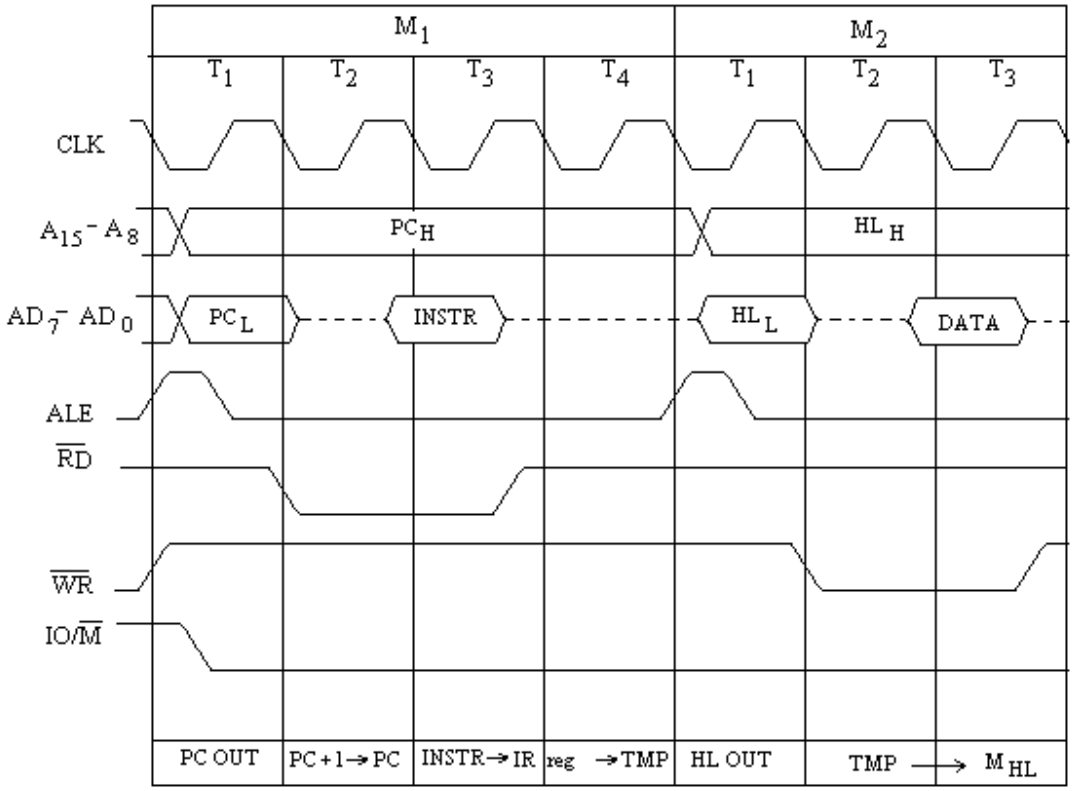


Fig. 5.10

5.4.3 Timing Diagram of *MVI reg, data*

Figure 5.11 illustrates the timing diagram of the instruction ***MVI reg, data***. This is an immediate move instruction. It is two byte instruction; one byte is used for the op code of the instruction and the other byte is used for the data. During T_1, T_2 states of op code fetch cycle M_1 the op code of the instruction (INSTR) is loaded in the instruction register (IR). The operation for these T-states is similar to other instructions discussed earlier. The T_4-state of this cycle M_1 is for decoding.

In the second machine cycle M_2, the contents of the program counter (PC) is placed on the address and address data buses during its T_1 state. This is the address of the second byte. During T_2 of M_2 cycle, the PC is incremented. At the same time memory is accessed and immediate read data is loaded to the specified register during T_3-state of M_2 machine cycle. This instruction takes two machine cycles with 7 T-states.

MVI reg, data

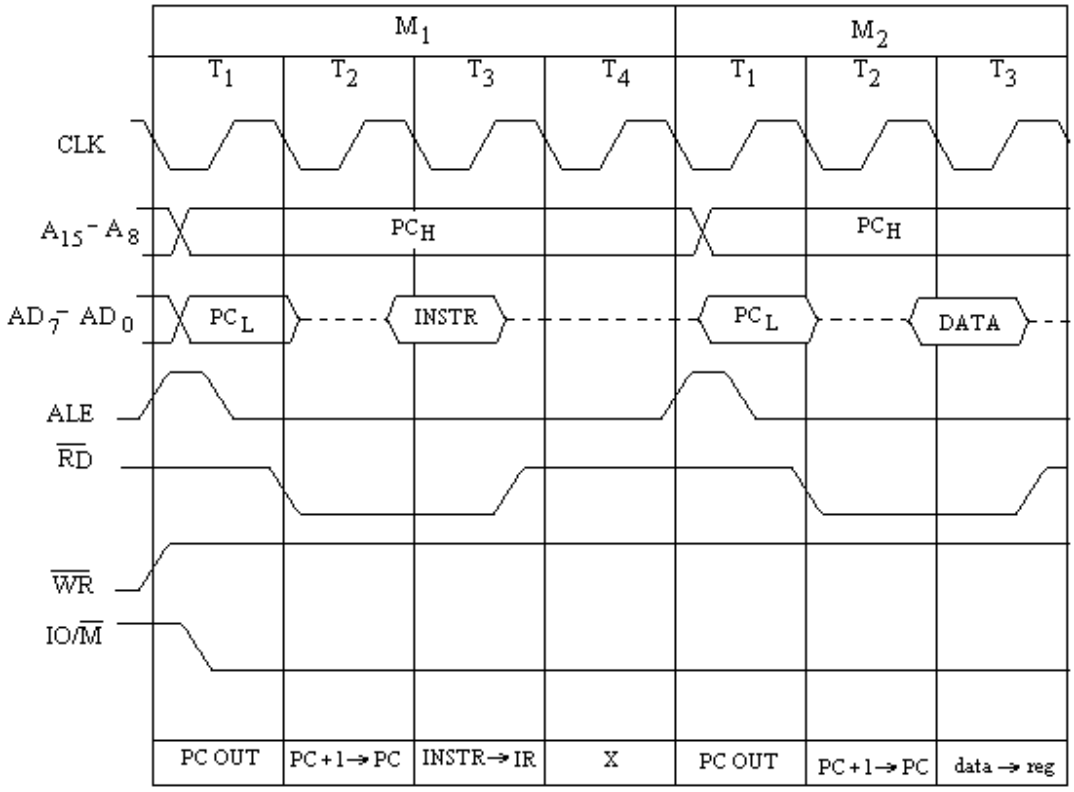


Fig. 5.11

5.4.4 Timing Diagram of *MOV reg2, reg1*

The timing diagram of ***MOV reg2, reg1*** is shown in figure 5.12. This instruction copies the contents of reg2 to reg1. So T_1, T_2 and T_3 states of the op code fetch cycle M_1 as usual indicate PC OUT, increment state ($PC+1 \rightarrow PC$), copying of the op code on the instruction to the instruction register ($INSTR \rightarrow IR$) operations respectively. Accordingly, ALE sends a high pulse at the beginning of T_1-state, $\overline{RD}$ activates (low) at the beginning of T_2 state till the middle of T_3 state till the middle of T_3 state of M_1 cycle. The $IO/\overline{M}$ signal becomes low at the beginning of T_1 till the end of T_3 of M_2 machine cycle. During T_4 state the contents of reg2 is copied in Temporary register ($reg2 \rightarrow TMP$).

This instruction does not need address bus and address data bus during the second machine cycle M_2. The only thing to happen during M_2 cycle is the transfer of the contents of Temporary register to reg1, for which address and address data buses are not required. So time process starts i.e. during the second machine cycle M_2 of the instruction ***MOV reg2, reg1,*** fetching operation of the next instruction will take place. This is called Fetch Execute Overlap (FEO). Because of FEO, the new address of PC is placed on the address and address data buses during T_1 state of M_2 machine cycle. Thus at the trailing

edge of ALE signal, this address is latched into the memory chips. The contents of temporary register are copied into reg1 at the T2 state of this machine cycle. This completes the execution ***MOV reg2, reg1*** instruction. However at the same T-state, program counter is incremented. The op code of the next instruction is copied in IR during T_3-state of M_2.

It may be mentioned here that as shown in figure 5.12, this instruction takes two machine cycles with 7 T-states; but because of FEO only one machine cycle with 4 T-states are counted to fetch and execute this instruction.

MOV reg2, reg1

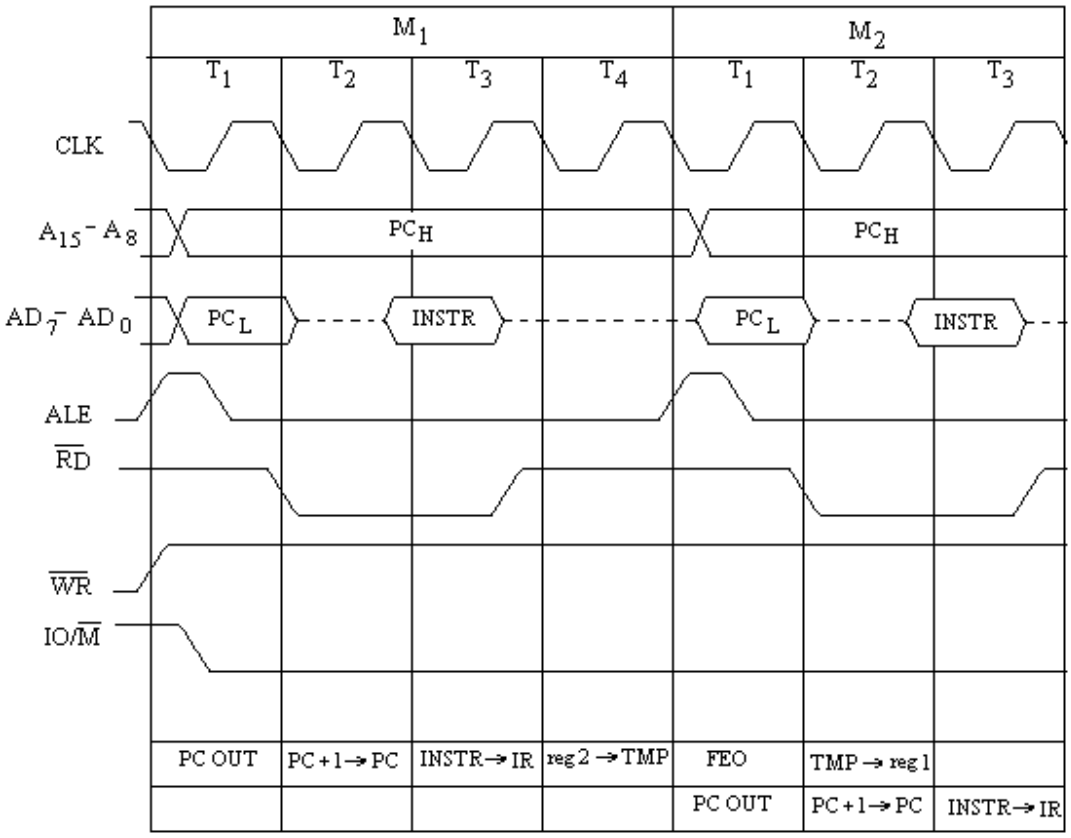


Fig. 5.12

5.4.5 Timing Diagram of *MVI M, data*

The timing diagram of ***MVI M, data*** instruction is shown in figure 5.13. This is a two byte instruction which takes 3 machine cycles with 10 T-states. The machine cycle M_1 is the op code fetch cycle, whose operation has already been discussed. In the T_1 state of M_2 cycle, the incremented address of PC is latched into memory chips. The T_2-state is the increment state and during the T_3 state of M_2 data is transferred to temporary register. This machine cycle is memory read cycle.

The third machine cycle is basically memory write cycle during which the contents of temporary register are copied in the memory location addressed by H-L register pair. In the T_1 state of the machine cycle M_3, the contents of H-l register pair is placed on the address bus and address data bus. It is then latched in the memory chips. In T_2 state the $\overline{WR}$ signal is low till the middle of T_3 state of this cycle. So the contents of

temporary register are copied in M_{H-L} location during T_2 and T_3 state of M_3 machine cycle.

MVI M, data

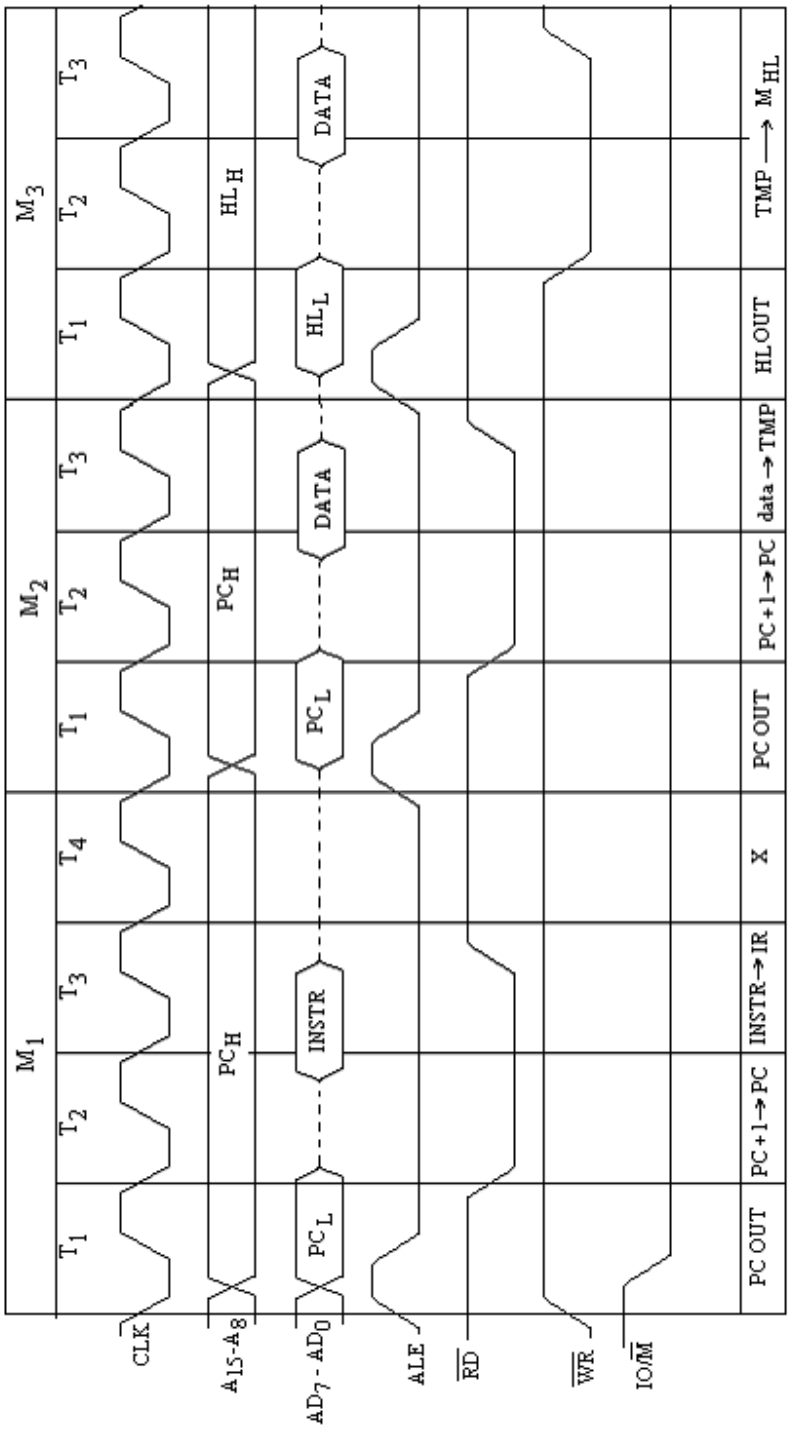


Fig. 5.13

5.4.6 Timing Diagram of *XCHG*

Let us discuss the timing diagram of one byte instruction XCHG. Its timing diagram is shown in figure 5.14. It exchanges the contents of H-L register pair with contents of D-E register pair.

The operation of M_1 machine cycle of this instruction is similar to other instructions discussed above. In this instruction the contents of H-L and D-E register pairs are to be exchanged, so address bus and address data bus are not needed. For this the operation FEO (Fetch Execute Overlap) occurs and as usual the next instruction is fetched from the memory for the M_2 machine cycle. However, during T_2 state of this machine cycle the contents of H-E and D-E pairs are exchanged ($H-L \leftrightarrow D-E$).

Because of FEO, XCHG instruction is considered to take one machine cycle with 4 T-states.

XCHG

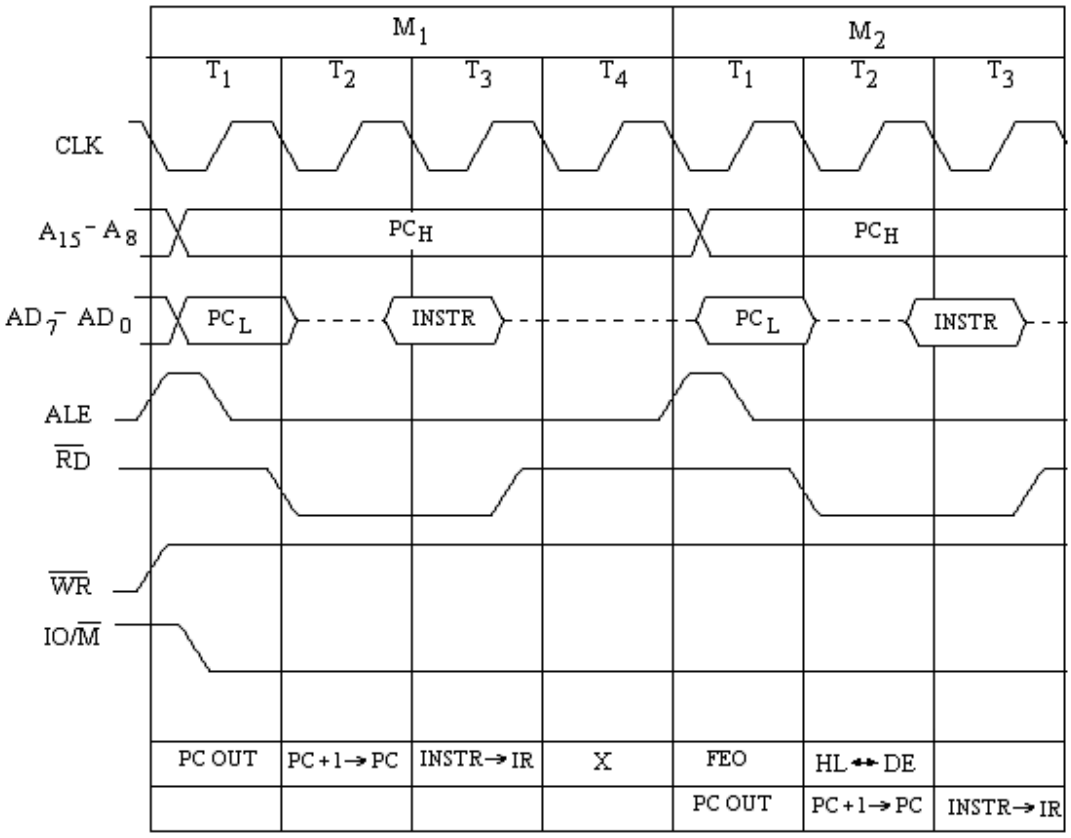


Fig. 5.14

5.4.7 Timing Diagram of *LXI rp, dbyte*

The timing diagram of 3-byte instruction ***LXI rp, dbyte*** is shown in figure 5.15. It loads the low byte of dbyte (double byte data) to the lower register of the given register pair *rp* and high byte to the high register of *rp*. This instruction takes three memory cycles with 10 T-states. First machine cycle is the op code fetch cycle which fetches the

op code of the instruction using the same procedure as discussed above for the other examples.

LXI rp, data

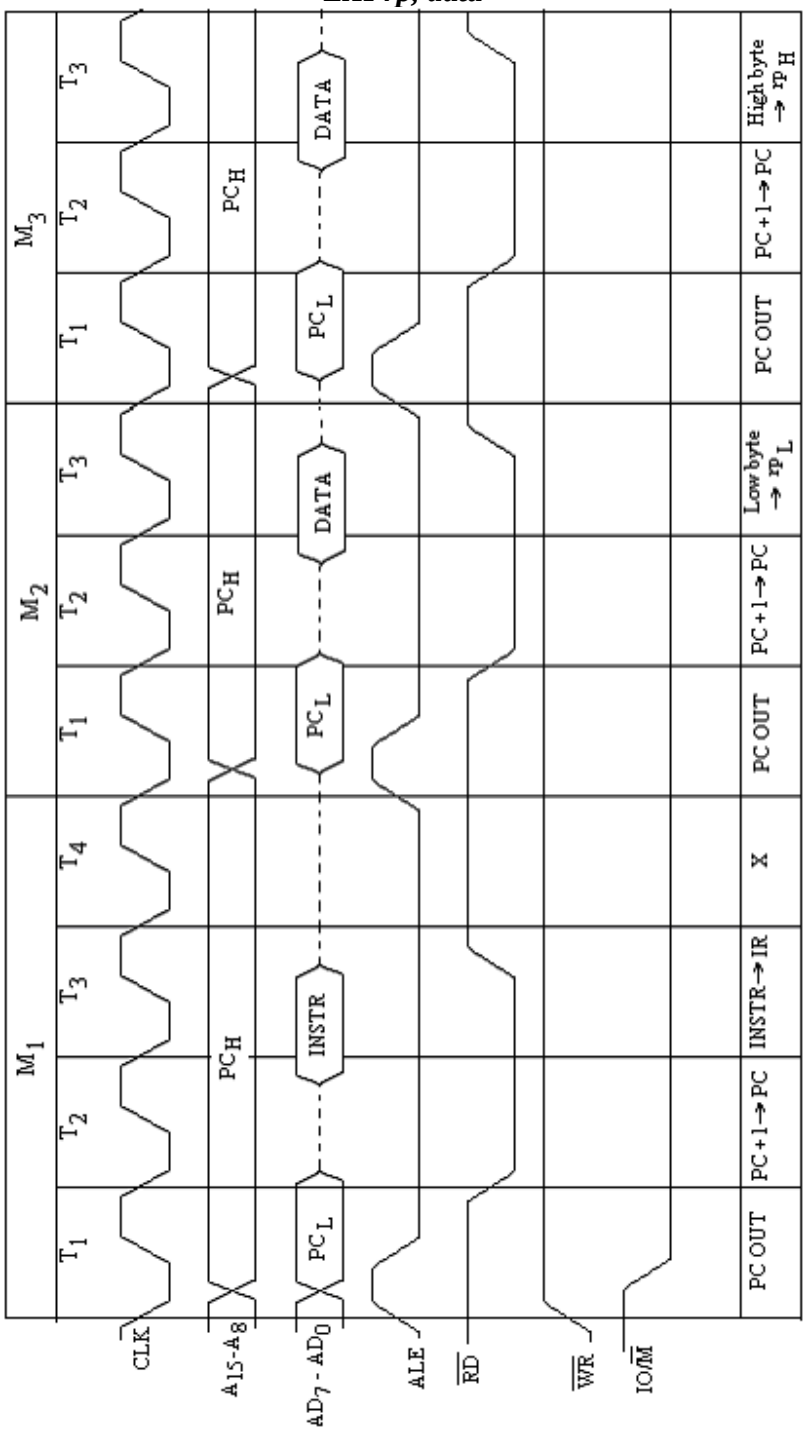


Fig. 5.15

In the T_1-state of the second machine cycle, address bus and address data bus are used to fetch the second byte (low byte of dbyte), for which ALE is enable at the beginning to the middle of this T-state. During T_2 state of M_2, PC is incremented and at T_3-state low byte of the dbyte is stored in lower register of *rp* (Low byte $\rightarrow rp_L$). This is a read cycle so $\overline{RD}$ is low at the beginning of this T-state to the middle of T_3 state. Same operation as for M_2 is, therefore, performed during M_3 machine cycle so that high byte of dbyte is stored in high register of register pair rp (High byte $\rightarrow rp_H$).

5.4.8 Timing Diagram of *IN byte*

The timing diagram of ***IN byte*** shown in figure 5.16 will now be discussed. This is an I/O read cycle and the microprocessor reads the data available at an input port or input device. The address of the port is of one byte given with the instruction. The data read out from the output port will be placed in the accumulator (A). This instruction is a two byte long and takes three memory cycles with 10 T-states. First machine cycle is an op code fetch cycle, the second is memory read cycle and the third is output read cycle.

We are familiar with the op code fetch cycle M_1. So the discussion will be made for the second and third machine cycles. In the T_1-state of second machine cycle M_2, the address of PC (incremented address) will be latched into the memory chips. At the beginning of T_1 state ALE sends a high pulse. During the T_2-state PC is incremented ($PC+1 \rightarrow PC$). Near the end of T_2-state of M_2, a byte appears on address data bus. During the T_3-state of M_2 machine cycle, this byte is copied in the W and Z registers (byte $\rightarrow$ Z, W), i.e. W and Z registers have the same byte. This byte is port address which is of 8 bits.

During T_1-state of the third machine cycle M_3, $IO/\overline{M}$ signal goes high which enables the peripheral chips for an I/O operation rather than memory operation. During this T-state, the contents of W-register are placed on the address bus and the contents of Z-register on address data bus (WZ OUT). In case of I/O device or I/O port the address is only of 8 bit long and therefore, the address of I/O device is duplicated on both address bus and address data bus. During T_2-state of M_3 cycle, $\overline{RD}$ signal goes low which indicates that it is I/O read operation. The data read out from the input port (input device) will be copied in the accumulator during T_2 and T_3 states.

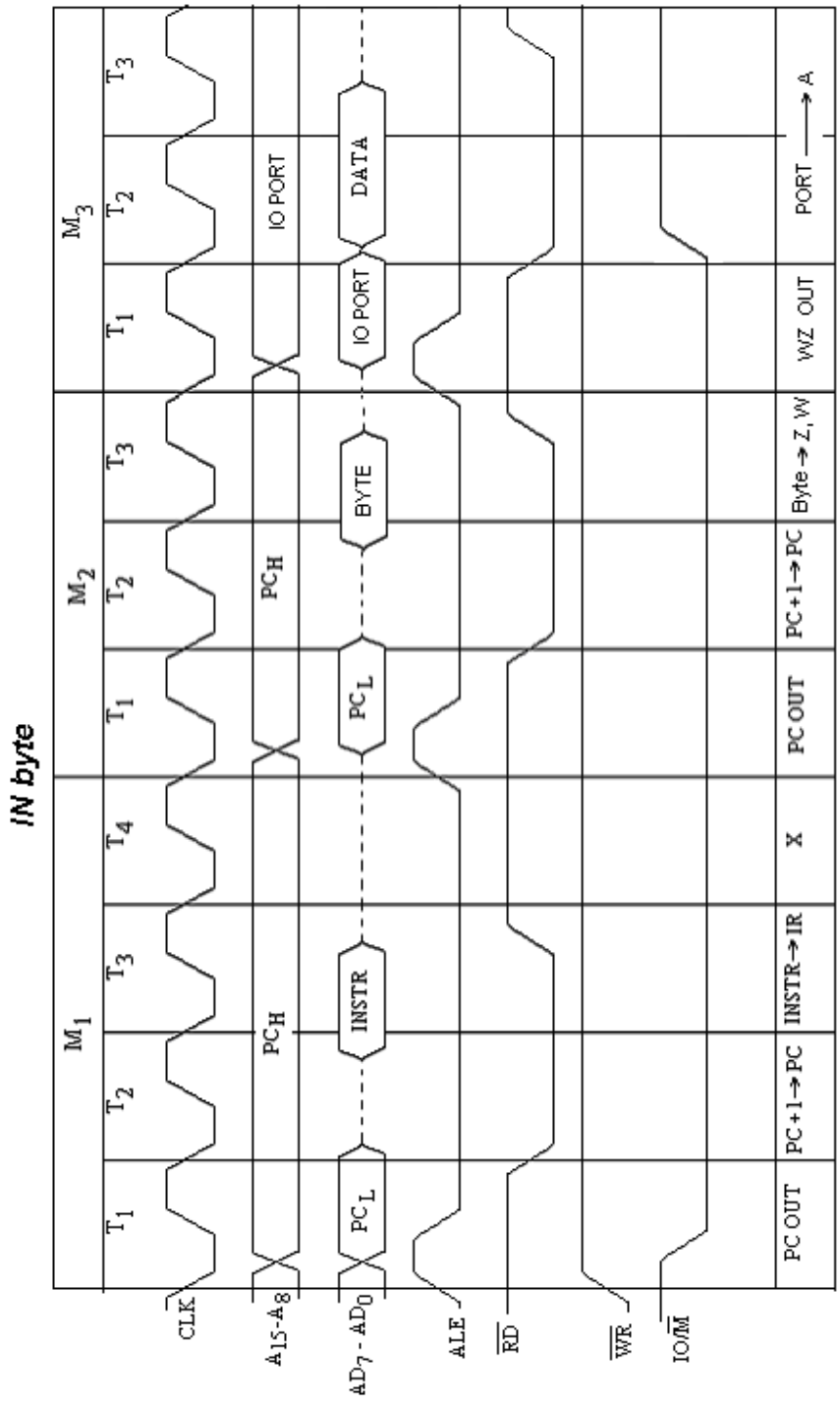

IN byte
M1
M2
M3
T1
T2
T3
T4
T1
T2
T3
T1
T2
T3
CLK
A15-A8
AD7 - AD0
ALE
RD
WR
IO/M
PCH
PCL
INSTR
PCH
PCL
BYTE
IO PORT
IO PORT
DATA
PC OUT
PC+1→ PC
INSTR→ IR
X
PC OUT
PC+1→ PC
Byte→ Z, W
WZ OUT
PORT ⟶ A

Fig. 5.16

PROBLEMS

1. Draw and discuss the architecture of 8085 microprocessor.
2. Mention various flags provided in 8085 microprocessor and discuss their roles.
3. Discuss the functions of program counter, Stack pointer and status flags in the architecture of 8085 microprocessor.
4. Discuss the role of address buffer and address data buffer in the architecture of 8085 microprocessor.
5. Discuss the functions of the following signals of 8085 microprocessor:
   ALE, $\overline{WR}$, $\overline{RD}$, S_0 and S_1.
6. Discuss *LHLD address*, *SHLD address*, *LDAX rp* and *STAX rp* instructions of 8085.
7. Discuss the following instructions of 8085
   XCHG, XTHL, SPHL and PCHL
   Which flags are affected by these instructions?
8. Discuss DAA instructions of 8085. Explain how flags are affected with this instruction.
9. Draw and discuss the timing diagram of *MOV reg2, reg1*.
10. Draw and discuss the timing diagram of *MOV reg, M.*
11. Draw and discuss the timing diagram of *MOV M, reg.*
12. Draw and discuss the timing diagram of *MVI reg, data.*
13. Draw and discuss the timing diagram of *MVI M, data.*
14. Draw and discuss the timing diagram of *LXI rp, dbyte.*
15. Draw and discuss the timing diagram of *XCHG.*
16. Draw and discuss the timing diagram of *IN byte.*
17. Discuss *LDAX rp* instruction of 8085. What instruction should be used to move the contents stored in memory location whose address is stored in H-L register pair.
18. What instruction should be used to store the accumulator contents to the memory location addressed by D-E register pair? Discuss that instruction in detail.
19. Discuss PCHL instruction of 8085. Explain how this instruction is said to be used as unconditional jump instruction.
20. What will be contents of accumulator and flag register, after the execution of following program:

```
MVI A, 47 H
MVI B, 37 H
ADD B
DAA
HLT
```

6

Programming of 8085

In this chapter assembly language programming of different programs will be discussed. It will help readers to go into the details of 8085 programming operations and their applications. For this, one is supposed to be well acquainted with 8085 instruction set discussed in the preceding chapters. Though some programming examples have also been discussed in these chapters, yet this chapter will exclusively deal with some more programming examples. The assembly language program written by the programmer in mnemonic form is also called as source program. The instructions, operand and data may be converted to binary (machine language) either by hand assembler or machine assembler. In case of hand assembler the hexadecimal data (op code / operand and data) are fed to the microprocessor kit through the hexadecimal key board. But in machine assembler, instructions (in mnemonic form) with data or operand are directly fed to the microprocessor kit through Alpha-numeric key board (computer key board). Here we are concerned with the assembly language program (source program).

In computers high level language like C, C++, Pascal, BASIC etc are used, which are easier than the assembly language. The computer is used for the conversion of high level language to assembly language. The advantage of using assembly language is that it can directly go into the details of the microprocessor's register and manipulate as the requirement.

6.1 SIMPLE PROGAMS

The simple programs based on the instruction set of 8085 microprocessor are given in the following examples.

Example 6.1. *Write an assembly language program of 8085 to find 1's complement of the data stored in memory location 2500 H and the result is to be stored in memory location 2501 H.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LDA	2500 H	; Load the data from 2500 H memory location to accumulator.
	CMA		; Complement the contents of accumulator (Converts in 1's complement).
	STA	2501 H	; Store the result in memory location 2501 H.
	HLT		; Stop processing.

Example 6.2. *Write an assembly language program of 8085 to find 2's complement of the data stored in memory location 2100 H and the result is to be stored in memory location 2101 H.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LDA	2100 H	; Load the data from 2100 H memory location to accumulator.
	CMA		; Complement the contents of accumulator (Converts in 1's complement).
	ADI	01 H	; 01 is added to the accumulator contents to get the 2's complement.
	STA	2101 H	; Store the result in memory location 2101 H.
	HLT		; Stop processing.

Example 6.3. *Write an assembly language program of 8085 to find 1's complement of n (decimal number) bytes of data stored in memory location starting at 2501 H. The number n (decimal number) is stored in memory location 2500 H. Store the result in memory locations starting at 2601 H.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI D,	2601 H	; Get first address of the destination in D-E register pair.
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	MOV C,	M	; Data in 2500 H is loaded to C-register which will be used as counter.
	INXH		; Increment H-L register pair.
LOOP	MOV A,	M	; Accumulator is loaded with the data.
	CMA		; Complement the contents of accumulator (Converts in 1's complement)
	STAX D		; Store the result in memory location addressed by D-E register pair.
	INX D		; Increment D-E register pair.
	INX H		; Increment H-L register pair.
	DCR C		; Decrement the C-register data.
	JNZ	LOOP	; If data in C-reg. is not zero jump to LOOP for next conversion.
	HLT		; Stop processing.

Example 6.4. *Write an assembly language program of 8085 to find 9's complement of n bytes (in BCD) of data stored in memory location starting from 2501 H. The number n*

(decimal number) is stored in memory location 2500 H. Store the result in memory locations starting from 2601 H.

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI D,	2601 H	; Get first address of the destination in D-E register pair.
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	MOV C,	M	; Data in 2500 H is loaded to C-register which will be used as counter.
	INX H		; Increment H-L register pair.
LOOP	MVI A,	99 H	; Get 99 H in accumulator.
	SUB M		; Subtract each byte by 99 H to get 9's complement.
	STAX D		; Store the result in memory location addressed by D-E register pair.
	INX D		; Increment D-E register pair.
	INX H		; Increment H-L register pair.
	DCR C		; Decrement the C-register data.
	JNZ	LOOP	; If data in C-reg. is not zero jump to loop for next conversion.
	HLT		; Stop processing.

It is clear from this example that for 9's complement each BCD byte is subtracted from 99.

Example 6.5. *Write an assembly language program of 8085 to find 2's complement of a 16 bit number stored in memory locations 2101 H and 2102 H. The least significant byte is in 2101 H. The result is to be stored in memory locations 2103 H and 2104 H.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2101 H	; Get H-L pair with 2101 H.
	MVI B,	00 H	; Store 00 to register B which will be used to store carry.
	MOV A,	M	; Accumulator is loaded with the data.
	CMA		; Complement the contents of accumulator (Converts in 1's complement).
	ADI	01 H	; Add 01 H to the accumulator content to get 2's complement.
	STA	2103 H	; Store the result in memory location 2103 H.
	JNC	NXT	; If there is no carry jump to NXT.
	INR B		; Increment 1 to B-register as carry.

```
NXT     INX H                   ; Increment H-L register pair for the
                                  second byte.
        MOV A,      M           ; Move the second byte to
                                  accumulator.
        CMA                     ; Complement the contents of
                                  accumulator.
        ADD B                   ; Add carry stored in register B.
        STA         2504 H      ; Store in 2504 H.
        HLT                     ; Stop processing.
```

From this example it is clear that 1 is added to the 1's complement of LS byte and to the 1's complement of MS byte 1 is added if there is a carry from the previous byte.

Example 6.6. *Write an assembly language program of 8085 to find 10's complement of a 16 bit number (BCD) stored in memory locations 2101 H and 2102 H. The least significant byte is in 2101 H. The result is to be stored in memory locations 2103 H and 2104 H.*

Solution.

Program:

```
Label   Mnemonics   Operand     Comments
        LXI  H,     2101 H      ; Get H-L pair with 2101 H.
        MVI B,      00 H        ; Store 00 to register B which will be
                                  used to store carry.
        MVI A,      99 H        ; Store 99 in accumulator
        SUB M                   ; Subtract each byte by 99 H to get
                                  9's complement.
        ADI         01 H        ; Add 01 H to the accumulator
                                  content to get 10's complement.
        STA         2103 H      ; Store the result in memory location
                                  2103 H.
        JNC         NXT         ; If there is no carry jump to NXT.
        INR B                   ; Increment 1 to B-register as carry.
NXT     INX H                   ; Increment H-L register pair for the
                                  second byte.
        MVI A,      99 H        ; Store 99 in accumulator
        SUB M                   ; Subtract each byte by 99 H to get
                                  9's complement.
        ADD B                   ; Add carry stored in register B.
        STA         2104 H      ; Store in 2104 H.
        HLT                     ; Stop processing.
```

From this example it is clear that 1 is added to the 9's complement of LS byte and to the 9's complement of MS byte 1 is added if there is a carry from the previous byte.

Example 6.7. *Write an assembly language program of 8085 to find 2's complement of N bytes ($N \geq 2$). The number of bytes N in hexadecimal is stored in 2100 H. The bytes are stored in memory locations starting at 2101 H. The least significant byte is in 2101 H. The result is to be stored in memory locations starting at 2201 H.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2100 H	; Get H-L pair with 2100 H.
	LXI D,	2201 H	; Get D-E pair with 2201 H.
	MOV C,	M	; Get the number N in c-register to be used as counter.
	INX H		; Increment H-L pair.
	MOV A,	M	; Accumulator is loaded with first data.
	CMA		; Complement the contents of accumulator (Converts in 1's complement).
	ADI	01 H	; Add 01 H to the accumulator content to get 2's complement.
	STAX D		; Store the result in memory location addressed by D-E pair.
	JNC	NXT	; If there is no carry jump to NXT.
	MVI B,	01 H	; Store 01 to B-register as carry.
	JMP	NXT1	; Jump to NXT1.
NXT	MVI B,	00 H	; Store 00 to B-register as carry.
NXT1	DCR C		; Decrement C-reg.
LOOP	INX H		; Increment H-L pair.
	INX D		; Increment D-E pair.
	MOV A,	M	; Move the next byte to accumulator.
	CMA		; Complement the contents of accumulator.
	ADD B		; Add carry stored in register B.
	STAX D		; Store in the memory location addressed by D-E pair.
	JNC	NXT2	; If there is no carry jump to NXT2.
	MVI B,	01 H	; Store 01 to B-register as carry.
	JMP	NXT3	; Jump to NXT3.
NXT2	MVI B,	00 H	; Store 00 to b-register as carry.
NXT3	DCR C		; Decrement C-reg.
	JNZ	LOOP	; If not zero jump to LOOP.
	HLT		; Stop processing.

Example 6.8. *Write an assembly language program of 8085 to find 10's complement of N bytes ($N \geq 2$). The number of bytes N in hexadecimal is stored in 2100 H. The bytes are stored in memory locations starting at 2101 H. The least significant byte is in 2101 H. The result is to be stored in memory locations starting at 2201 H.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2100 H	; Get H-L pair with 2100 H.
	LXI D,	2201 H	; Get D-E pair with 2201 H.

```
        MOV C,    M       ; Get the number N in c-register to be
                            used as counter.
        INX  H              ; Increment H-L pair.
        MVI A,    99 H      ; Accumulator is loaded with 99 H.
        SUB M               ; Get the 9's Complement of the
                            number stored in location
                            addressed by H-L pair.
        ADI       01 H      ; Add 01 H to the accumulator
                            content to get 2's complement.
        STAX D              ; Store the result in memory location
                            addressed by D-E pair.
        JNC       NXT       ; If there is no carry jump to NXT.
        MVI B,    01 H      ; Store 01 to B-register as carry.
        JMP       NXT1      ; Jump to NXT1.
NXT     MVI B,    00 H      ; Store 00 to b-register as carry.
NXT1    DCR C               ; Decrement C-reg.
LOOP    INX  H              ; Increment H-L pair.
        INX  D              ; Increment D-E pair.
        MVI A,    99 H      ; Get 99 in accumulator.
        SUB M               ; Get 9's Complement of the
                            contents of accumulator.
        ADD B               ; Add carry stored in register B to
                            get 10's complement.
        STAX  D             ; Store in the memory location
                            addressed by D-E pair.
        JNC       NXT2      ; If there is no carry jump to NXT2.
        MVI B,    01 H      ; Store 01 to B-register as carry.
        JMP       NXT3      ; Jump to NXT3.
NXT2    MVI B,    00 H      ; Store 00 to b-register as carry.
NXT3    DCR C               ; Decrement C-reg.
        JNZ       LOOP      ; If not zero jump to LOOP.
        HLT                 ; Stop processing.
```

Example 6.9. *Write an assembly language program of 8085 to combine two hex nibbles stored in 2500 H and 2501 H memory locations, to form a byte. The least significant nibble is stored in 2500 H and most significant nibble is stored in 2501 H. The byte thus combined should be stored in 2502 H. (Let 08 H is stored in 2500 H and 09 H is stored in 2501 H, after the program is executed 2502 H should be loaded with the combined byte 89 H).*

Solution.

Program:

```
Label   Mnemonics  Operand   Comments
        LXI  H,    2500 H    ; Get H-L pair with 2500 H.
        MOV A,     M         ; Data in 2500 H is loaded to
                             Accumulator.
        RLC                  ; Rotate left
```

```
        RLC                     ; Rotate left
        RLC                     ; Rotate left
        RLC                     ; Rotate left (Rotated left four times
                                  so that it is moved to MS nibble).
        INX H                   ; Increment H-L register pair.
        ORA M                   ; Oring of two nibbles combines
                                  them to form a byte in
                                  accumulator.
        INX H                   ; Increment H-L register pair.
        MOV M,      A           ; Move the accumulator content
                                  (required byte) to the memory
                                  location addressed by H-L register
                                  pair.
        HLT                     ; Stop processing.
```

Example 6.10. *Write an assembly language program of 8085 to separate a hexadecimal number into two nibbles. The hexadecimal number is stored in 2501 H memory location. The least significant nibble of the byte is to be stored in 2502 memory location and the most significant nibble is to be stored in 2503 H memory location. (Suppose 3A H is a byte stored in memory location 2501 H and the lower nibble 0A should be stored in 2502 H and 03 should be stored in 2503 H)*

Solution.

Program:

```
Label   Mnemonics   Operand     Comments
        LXI  H,     2501 H      ; Get H-L pair with 2501 H.
        MOV A,      M           ; Data in 2501 H is loaded to
                                  Accumulator.
        MOV B,      A           ; Accumulator content are also
                                  loaded to B register.
        ANI         0F H        ; Mask off the first four digits
                                  (higher nibble).
        INX H                   ; Increment H-L register pair.
        MOV M,      A           ; Lower nibble is loaded to the
                                  memory location addressed by H-L
                                  register pair.
        MOV A,      B           ; Get the byte again in the
                                  accumulator.
        ANI         F0 H        ; Mask off the lower nibble.
        INX  H                  ; Increment H-L register pair.
        MOV  M,     A           ; Higher nibble is loaded to the
                                  memory location addressed by H-L
                                  register pair.
        HLT                     ; Stop processing.
```

Example 6.11. *Write an assembly language program of 8085 to check a hexadecimal number stored in 2500 H memory location for odd or even parity. If the parity is even store data EE H to memory location 2501H otherwise store 00 H in 2501 H.*
Solution.
Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	MOV A,	M	; Data in 2500 H is loaded to Accumulator.
	ORA A		; Set the flag.
	JPE	EVEN	; Check for even parity, if parity is even jump to EVEN.
	INX H		; Increment H-L register pair.
	MVI M,	00 H	; Load 00 H to the memory location addressed by H-L register pair.
	JMP	DONE	; Jump to DONE.
EVEN	INX H		; Increment H-L register pair.
	MVI M,	EE H	; Load EE H to the memory location addressed by H-L register pair.
DONE	HLT		; Stop processing.

Example 6.12. *n (decimal number) data bytes are stored in the memory locations starting at 2501 H. Write an assembly language program of 8085 to check if 12 H is stored in any of the given locations. If any of the locations has 12 H then store 12 H in that location else load 00 H in that memory location.*
Solution.
Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	MOV C,	M	; Data in 2500 H is loaded to C reg. which is used as the counter.
LOOP	INX H		; Increment H-L register pair.
	MOV A,	M	; Load the content of M_{HL} to accumulator.
	CPI	12 H	; Compare if the accumulator data are 12 H.
	JZ	NXT	; If it is 12 H then jump to NXT.
	MVI M,	00 H	; Else load 00 H to the memory location addressed by H-L register pair.
	JMP	DONE	; Jump to DONE.
NXT	MVI M,	12 H	; Load 12 H to the memory location addressed by H-L register pair.

```
DONE    DCR C              ; Decrement the content of C-reg.
                             for next byte.
        JNZ     LOOP       ; If the contents of C-reg. are not
                             zero then jump to LOOP.
        HLT                ; Stop processing.
```

Example 6.13. *16 data bytes are stored in memory locations 2001 H to 2010 H. Write an assembly language program of 8085 to transfer this block of data bytes to memory locations 2501 to 2510 H.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2001 H	; Get H-L pair with 2001 H.
	LXI D,	2501 H	; Get the address of the memory location (destination) in D-E register pair
	MVI C,	10 H	; Get 10 H (16_{10}) Data in C register which is used as the counter.
LOOP	MOV A,	M	; Load the content of M_{HL} to accumulator.
	STAX D		; Load the content of accumulator to the memory locations addressed by D-E register pair.
	INX H		; Increment H-L register pair.
	INX D		; Increment D-E register pair.
	DCR C		; Decrement the content of C-reg. for next byte.
	JNZ	LOOP	; If the contents of C-reg. are not zero then jump to LOOP.
	HLT		; Stop processing.

Example 6.14. *16 data bytes are stored in memory locations 2001 H to 2010 H. Write an assembly language program of 8085 to transfer this block of data bytes to memory locations 2501 to 2510 H in the reverse order (i.e. the data of 2001 is to transferred to 2510 H and data of 2002 H to 250F H and so on).*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2001 H	; Get H-L pair with 2001 H.
	LXI D,	2510 H	; Get the address of the memory location (destination) in D-E register pair
	MVI C,	10 H	; Get 10 H (16_{10}) Data in C register which is used as the counter.

```
LOOP    MOV A,    M       ; Load the content of M_HL to accumulator.
        STAX D            ; Load the content of accumulator to the memory locations addressed by D-E register pair.
        INX H             ; Increment H-L register pair.
        DCX D             ; Decrement D-E register pair.
        DCR C             ; Decrement the content of C-reg. for next byte.
        JNZ       LOOP    ; If the contents of C-reg. are not zero then jump to LOOP.
        HLT               ; Stop processing.
```

Example 6.15. *Write an assembly language program of 8085 to clear the memory locations starting at 2001 (i.e. each location should have 00 H). The length of data bytes is given in 2000 H memory location.*

Solution.

Program:

```
Label   Mnemonics   Operand   Comments
        LXI  H,     2000 H    ; Get H-L pair with 2000 H.
        MOV C,      M         ; Use C register as counter.
        XRA A                 ; Clear the accumulator.
LOOP    INX H                 ; Increment H-L register pair.
        MOV M,      A         ; Load the accumulator content to the memory location.
        DCR C                 ; Decrement the content of C-reg. for next byte.
        JNZ         LOOP      ; If the contents of C-reg. are not zero then jump to LOOP.
        HLT                   ; Stop processing.
```

Example 6.16. *Write an assembly language program of 8085 to add five bytes of data stored in memory locations starting at 2000 H. If the sum generates a carry, stop addition and store 01 H to the memory location 2501 H; else continue addition and store the sum at 2501 memory location..*

Solution.

Program:

```
Label   Mnemonics   Operand   Comments
        LXI  H,     2000 H    ; Get H-L pair with 2000 H.
        MVI C,      05 H      ; Store the number of data bytes in C register as counter.
        XRA A                 ; Clear the accumulator and CY flag.
LOOP    ADD M                 ; Add the byte in accumulator.
```

	JC	NXT	; If there is a carry jump to NXT.
	INX H		; Increment the H-L register pair.
	DCR C		; Decrement the content of C-reg. for next byte.
	JNZ	LOOP	; If the contents of C-reg. are not zero then jump to LOOP.
	STA	2501 H	; Store the sum in memory location 2501 H.
	HLT		; Stop processing.
NXT	MVI A,	01 H	; If carry occurs load 01 to accumulator.
	STA	2501 H	; Store 01 H to 2501 H.
	HLT		; Stop processing.

Example 6.17. *Write an assembly language program of 8085 to add five bytes of data stored in memory locations starting at 2000 H. The sum may be more than one byte. Store the result at two consecutive memory locations2500 H and 2501 H.*

Solution.

Program:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI H,	2000 H	; Get H-L pair with 2000 H.
	MVI C,	05 H	; Store the number of data bytes in C register as counter.
	XRA A		; Clear the accumulator and CY flag.
	MOV B,	A	; Store the accumulator contents to B-register also for the carry.
LOOP	ADD M		; Add the byte in accumulator.
	JNC	NXT	; If there is a no carry jump to NXT.
	INR B		; Increment B as carry is generated.
NXT	INX H		; Increment the H-L register pair.
	DCR C		; Decrement the content of C-reg. for next byte.
	JNZ	LOOP	; If the contents of C-reg. are not zero then jump to LOOP.
	LXI H,	2500 H	; Store H-L pair with 2500 H (destination address).
	MOV M,	A	
	INX H		; Store the sum to 2500 H.
	MOV M,	B	; Content of B are stored in 2501 H.
	HLT		; Stop processing.

Example 6.18. *Write a program in assembly language of 8085 to test 6^{th} bit (D_6 bit) of a byte stored at 2500 H memory location. If the bit D_6 is zero then store 00 H at 2501 H else store the same number at 2501 H.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	MOV A,	M	; Store the byte in accumulator
	ANI	40 H	; Reset all the bits except D_6.
	JZ	END	; If this bit is zero jump to END.
	MOV A,	M	; Store the byte in accumulator again.
	INX H		; Increment H-L pair.
	MOV M,	A	; The byte is stored in 2501 H.
	HLT		; Stop processing.
END	XRA A		; Clear accumulator.
	INX H		; Increment H-L pair.
	MOV M,	A	; 00 H is stored in 2501 H.
	HLT		; Stop processing.

Example 6.19. *Write an ALP (Assembly Language Program) of 8085 to find logical OR and exclusive OR of two numbers stored at 2501 H and 2502 H memory locations. The results should be stored at 2503 H (logical OR operation) and 2504 H (XOR operation) memory locations.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2501 H	; Get H-L pair with 2501 H.
	MOV B,	M	; Store the byte in B-register.
	MOV A,	B	; Store this byte in accumulator also.
	INX H		; Increment H-L pair.
	MOV C,	M	; Store second byte to C register.
	ORA C		; Logical OR of two bytes.
	INX H		; Increment H-L pair.
	MOV M,	A	; Store the result (OR) in 2503 H.
	MOV A,	B	; Load the first number in accumulator.
	XRA C		; Ex-OR the two number.
	INX H		; Increment H-L pair.
	MOV M,	A	; Store the result (XOR)
	HLT		; Stop processing.

Example 6.20. *Write an ALP (Assembly Language Program) for 8085 to shift an 8-bit number left by one bit. The number is stored in 2101 H memory location. The result is to be stored in 2102 H memory location.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LDA	2101 H	; Get the number is accumulator.
	ADD A		; Shift it left by one bit.
	STA	2102 H	; Store the result in 2502 H.
	HLT		; Stop processing.

Example 6.21. *Write an ALP 8085 to shift a 16-bit number left by one bit. The number is stored in 2101 H and 2102 H memory locations. The result is to be stored in 2103 H and 2104 H memory locations.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LHLD	2101 H	; Get the 16 bit number in H-L register pair.
	DAD H		; Shift left by one bit.
	SHLD	2103 H	; Store the result in 2103 H and 2104 H memory locations.
	HLT		; Stop processing.

6.2 PROGAMS ON CODE CONVERSION

The programs on code conversion will now be discussed.

6.2.1 BCD to Binary Conversion

Suppose we have 0 to 99 decimal numbers (BCD numbers) and we wish to convert any of these numbers to its equivalent binary number, we proceed in the following way:

For example 49 (4 x 10 + 9) is the given decimal number, its binary equivalent is ($00011\ 0001_2$ = 31 H).

Step I — Unpack the number in MSD and LSD form.
- 0000 1001 LSD in four least significant nibble in a register.
- 0000 0100 MSD in four least significant nibble in anther register.

Step II — Multiply MSD by decimal number 10.
- or MSD is added 10 times with 0000.
- So we get (4 x 10 = 0010 1000).

Step III — ADD LSD to the result of step II.
- i.e. 00101000 + 00001001 = 00110001
- Hence we get the result.

Example 6.22. *A BCD number (0 to 99) is stored in a memory location 2500 H, write an ALP of 8085 to convert the BCD number into its equivalent binary number. Store the result in 2600 H memory location.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LXI SP,	XXXX H	; Initialize the stack pointer.
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	LXI D,	2600 H	; Get D-E pair with 2600 H.
	MOV A,	M	; Load the byte to accumulator.
	CALL	CONV	; Call conversion subroutine program.
	STAX D		; Store the result in 2600 H.
	HLT		

Subroutine Program:

Label	Mnemonics	Operand	Comments
CONV	PUSH D		; Push the contents of D-E register pair to stack.
	PUSH H		; Push the contents of H-L register pair to stack.
	ANI	0F H	; Separate LSD
	MOV C,	A	; Store it to C-register.
	MOV A,	M	; Load the byte to accumulator again.
	ANI	F0 H	; Separate MSD.
	RRC		; Rotate right four times
	RRC		; to shift MSD to four
	RRC		; least significant nibble
	RRC		; of accumulator.
	MOV D,	A	; Store MSD to D register.
	XRA A		; Clear accumulator.
	MVI E,	0A H	; Get 0A (decimal number 10) to E-register.
SUM	ADD D		; Add MSD ten times to 00 H.
	DCR E		; Decrement C
	JNZ	SUM	; If addition not complete then move to SUM.
	ADD C		; Add LSD to 10 X MSD
	POP H		; Pop the contents of H-L pair.
	POP D		; Pop the contents of D-E pair.
	RET		; Go back to main program.

6.2.2 Binary to BCD (Unpacked) Conversion

For example binary number is 1111 1111 (FF H). Its decimal equivalent is 255_{10} having the unpacked BCD numbers as

05 (0000 0101) BCD 1
05 (0000 0101) BCD 2
02 (0000 0010) BCD 3

The procedure for converting the 8 bit binary number (0000 0000 to 1111 1111) into unpacked BCD number is given in the following steps.

Step I If the number is less than 100 go to step II. If the number is more than 100, divide the number by 100 or subtract 100 repeatedly till the reminder is less than 100. The quotient is MS BCD (BCD 3).

Step II If the number (or remainder) is less than 10, go to step III. If number is >10 < 100, then subtract 10 repeatedly till the remainder is less than 10. The quotient is BCD 2.

Step III The remainder from step II is BCD 1.

Example 6.23. *A binary number (between 00 H to FF H) is stored in a memory location 2500 H, write an ALP of 8085 to convert this number into BCD number. Store each BCD as unpacked BCD digit in the memory locations at 2601 H to 2603 H.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LXI SP,	XXXX H	; Initialize the stack pointer.
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	MOV A,	M	; Load the byte to accumulator.
	CALL	CONV	; Call conversion subroutine program.
	HLT		

Subroutine Program 1:

Label	Mnemonics	Operand	Comments
CONV	LXI H,	2601 H	; Get H-L pair with 2601 H.
	MVI B,	64 H	; Load 100 in B-reg.
	CALL	CONV1	; Call another subroutine program for the division by 100.
	MVI B,	0A H	; Load 10 in B-reg.
	CALL	CONV1	; Call the subroutine again for the division by 10.
	MOV M,	A	; Load the accumulator contents to the memory location.
	RET		; Go back to main program.

Subroutine Program 2:

Label	Mnemonics	Operand	Comments
CONV1	MVI M,	FF H	; Get FF H in the $M_{H\text{-}L}$.
NXT	INR M		; Increment $[M_{H-L}]$
	SUB B		; Subtract the content of B from accumulator.
	JNC	NXT	; Jump to NXT if no carry.
	ADD B		; Add the contents of B to accumulator.
	INX H		; Increment H-L register pair.
	RET		; Go back.

Example 6.24. *Write ALP of 8085 for the following statement:*

A set of three packed BCD numbers (six digits or three bytes) are stored in memory locations starting at 2500 H. The seven segment codes of the digits 0 to 9 for common cathode LED are stored in memory locations starting at 2050 H; and the answer is to be stored in memory locations starting at 2070 H.

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LXI SP,	XXXX H	; Initialize the stack pointer.
	LXI H,	2500 H	; Get H-L pair with 2500 H.

	MVI D,	03 H	; Number of bytes to be converted is placed in D-register.
	CALL	CONV	; Call conversion subroutine program.
	HLT		

Subroutine Program 1:

Label	**Mnemonics**	**Operand**	**Comments**
CONV	LXI B,	2070 H	; Get B-C pair with 2070 H.
NXT	MOV A,	M	; Move the content of $[M_{H-L}]$to accumulator.
	ANI	F0 H	; Separate MSD.
	RRC		; Rotate right four times
	RRC		; to shift MSD to four
	RRC		; least significant nibble
	RRC		; of accumulator.
	CALL	SEGMENT	; Call subroutine program for the conversion of unpacked BCD to seven segment.
	INX B		; Increment B-C register pair.
	MOV A,	M	; Get the byte again.
	ANI	0F H	; Separate LSD.
	CALL	SEGMENT	; Call subroutine program for the conversion of unpacked BCD (LSD) to seven segment.
	INX B		; Increment B-C register pair.
	INX H		; Increment H-L register pair.
	DCR D		; Decrement the contents of D register.
	JNZ	NXT	; If the content in D register is not then jump to NXT for next byte.
	RET		; Return to main program.

Subroutine Program 2:

Label	**Mnemonics**	**Operand**	**Comments**
SEGMENT	PUSH H		; Push the contents of H-L register pair to stack.
	LXI H,	2050 H	; Get H-L pair with 2050 H.
	ADD L		; Add the content of L register to accumulator to get right location for the digit from the look-up table.
	MOV L,	A	; Store it to L-register.
	MOV A,	M	; Load the corresponding data from the look-up table to accumulator.
	STAX B		; Store the result in the memory location addressed by B-C register pair.

```
        POP H                   ; Get the contents of H-L register
                                  pair from the stack.
        RET                     ; Go back
```

DATA (in the form of look-up table) is stored in the following memory locations:

2050 H	3F H	(Digit 0)
2051 H	06 H	(Digit 1)
2052 H	5B H	(Digit 2)
2053 H	4F H	(Digit 3)
2054 H	66 H	(Digit 4)
2055 H	6D H	(Digit 5)
2056 H	7D H	(Digit 6)
2057 H	07 H	(Digit 7)
2058 H	7F H	(Digit 8)
2059 H	6F H	(Digit 9)

The 3F H data is given for the digit 0. It is clear from the figure 6.1.

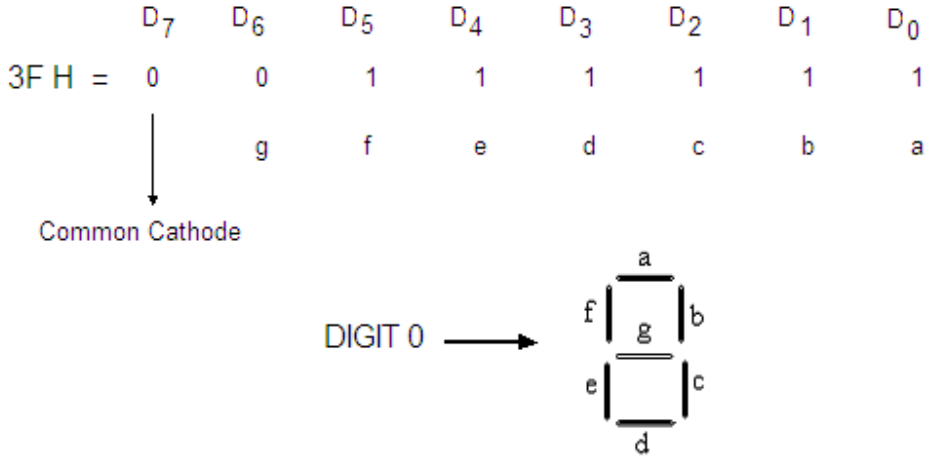


Fig. 6.1

6.2.3 Binary to ASCII Conversion

ASCII (American Standard Code for Information Interchange) is a most commonly used alphanumeric code. It is a seven bit code. The hexadecimal numbers 30 to 39 (0011 0000 to 0011 1001) represent 0 to 9 ASCII decimal numbers. Similarly, 41 H to 5A H (0100 0001 to 0101 1010) represent capital letters A to Z in ASCII.

For example, the ASCII representation of a binary number 8E is 38 H and 45 H (as 8 is represented by 0011 1000 or 38 H; and E is represented by 0100 0101 or 45 H).

The following steps are carried out for the conversion of Binary to ASCII

Step I First separate the LSD and MSD of the given binary number (or byte). Each digit is then converted to ASCII.

Step II If the digit is less than 10 (0A H) then add 30 H (0011 0000) to the binary number else add 37 H (0011 0111).

For binary number 8E H, 30 is added to 08 and we get 38 H (the ASCII Code for 8) and 37 is added to 0E H to get 45 H (the ASCII Code for E).

Example 6.25. *An eight bit binary number is stored in 2500 H. Write an ALP for 8085 to convert the binary number to its equivalent ASCII code. The ASCII code for most*

significant binary digit should be stored to 2601 H location; and the code for least significant binary digit should be stored to 2602 H location.

Solution.

Main Program:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	; Initialize the stack pointer.
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	LXI D,	2601 H	; Get D-E pair with 2601 H.
	MOV A,	M	; Move the content of $[M_{H-L}]$to accumulator.
	MOV B,	A	; Move the accumulator content to B-register.
	RRC		; Rotate right four times
	RRC		; to shift MSD to four
	RRC		; least significant nibble
	RRC		; of accumulator.
	CALL	ASCII	; Call the subroutine to convert MSD to ASCII.
	STAX D		; Store the ASCII code of MS digit to the memory location addressed by D-E register pair.
	INX D		; Increment D-E register pair.
	MOV A,	B	; Move the binary number again to accumulator.
	CALL	ASCII	; Call the subroutine to convert LSD to ASCII.
	STAX D		; Store the ASCII code of LS digit to the memory location addressed by D-E register pair.
	HLT		; Stop processing.

Subroutine Program:

Label	**Mnemonics**	**Operand**	**Comments**
ASCII	ANI	0F H	; Isolate MS digit.
	CPI	0A	; Compare with 10 (0A H).
	JC	NXT	; If it less than 10, jump to NXT.
	ADI	07 H	; Else add 07 H.
NXT	ADI	30 H	; Add 30 H.
	RET		; Go back to main program.

6.2.4 ASCII to Binary Conversion

The following steps are carried out to convert the ASCII code to binary (Hex):

Step I	Subtract 30 H (0011 0000) from the given binary (Hex) number.
Step II	If the difference after subtraction in step I is less than 10, then it is the required binary (Hex) number.

Step III If the difference after subtraction in step II is more than 10 then subtract 07 also from it. This will then give the required binary (Hex) number.

Example 6.26. *An ASCII number is stored in 2500 H memory location. Write an ALP for 8085 to convert this number into its equivalent binary (Hex) number and store it to 2501 memory location.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LXI SP,	XXXX H	; Initialize the stack pointer.
	LXI H,	2500 H	; Get H-L pair with 2500 H.
	MOV A,	M	; Move the content of $[M_{H-L}]$ to accumulator.
	CALL	CONV	; Call the subroutine to convert ASCII to binary.
	INX H		; Increment H-L register pair.
	MOV M,	A	; Store the binary number to required memory location.
	HLT		; Stop processing.

Subroutine Program:

Label	Mnemonics	Operand	Comments
CONV	SUI	30 H	; Subtract 30 H from accumulator.
	CPI	0A	; Compare with 10 (0A H).
	RC		; If it less than 10 go back to main program.
	SUI	07 H	; Else subtract 07 H.
	RET		; Go back to main program.

6.3 PROGAMS ON ADDITION AND SUBTRACTION

Now we shall take up the problems on BCD or decimal addition and subtraction of two bytes or multibyte. These problems are self explanatory and can be understood by considering proper logic.

Example 6.27. *Write an ALP for 8085 to add two 16-bit numbers. The augend's LS byte is stored in 2101 H and MS byte in 2102 H. The addend's LS and MS byte are stored in 2103 H and 2104 H respectively. The result is to be stored in 2105 H to 2107 H.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LHLD	2101 H	; Load the H-L pair direct with the contents of 2101 H and 2102 H Locations.
	XCHG		; Exchange the contents of H-L and D-E pair.
	LHLD	2103 H	; Load the H-L pair direct with the contents of 2103 H and 2104 H Locations.

```
        MVI  C,     00 H    ; Store 00 H to C register for carry.
        DAD  D              ; Add the contents of H-L and D-E
                              register pair and result is in H-L
                              pair.
        JNC         NXT     ; If there is no carry after the
                              addition jump to NXT.
        INR  C              ; Increment the carry.
NXT     SHLD        2105 H  ; Store the contents in the required
                              locations.
        MOV  A,     C       ; Move carry to accumulator.
        STA         2107 H  ; Store carry in 2107 H.
        HLT                 ; Stop processing.
```

Example 6.28. *Write an ALP for 8085 to add two N byte numbers. The augend's bytes are stored in memory locations starting at 2101 H and the addend's bytes are stored in memory locations starting at 2201 H. The number N is stored in memory location 2100 H. The result is to be stored in the memory locations starting at 2201 H.*

Solution.

Main Program:

```
Label   Mnemonics  Operand     Comments
        LXI  H,    2100 H    ; Get H-L pair with 2100 H.
        MOV C,     M         ; Move the number N to C-register
                               for counter.
        INX  H               ; Increment the H-L pair.
        LXI  D,    2201 H    ; Get D-E pair with 2201 H.
        XRA  A               ; Clear the accumulator and carry
                               flag.
LOOP    LDAX D               ; Get the addend byte to
                               accumulator.
        ADC  M               ; Add with carry the two bytes.
        MOV  M,    A         ; Store the result in the location
                               addressed by H-L pair.
        INX  D               ; Increment the D-E pair.
        INX  H               ; Increment the H-L pair.
        DCR  C               ; Decrement C.
        JNZ        LOOP      ; If C is not zero, then jump to
                               LOOP.
        HLT                  ; Stop processing.
```

Example 6.29. *Write an ALP for 8085 to add two N byte numbers. The augend's bytes are stored in memory locations starting at 2101 H and the addend's bytes are stored in memory locations starting at 2201 H. The number N is stored in memory location 2100 H. The result should in decimal form and is to be stored in the memory locations starting at 2201 H.*

Solution.

Main Program:

```
Label   Mnemonics  Operand     Comments
```

```
        LXI  H,     2100 H      ; Get H-L pair with 2100 H.
        MOV C,      M           ; Move the number N to C-register
                                  for counter.
        INX  H                  ; Increment the H-L pair.
        LXI  D,     2201 H      ; Get D-E pair with 2201 H.
        XRA  A                  ; Clear the accumulator and carry
                                  flag.
LOOP    LDAX D                  ; Get the addend byte to
                                  accumulator.
        ADC  M                  ; Add with carry the two bytes.
        DAA                     ; Decimal adjust the accumulator for
                                  the result in decimal form.
        MOV  M,     A           ; Store the result in the location
                                  addressed by H-L pair.
        INX  D                  ; Increment the D-E pair.
        INX  H                  ; Increment the H-L pair.
        DCR  C                  ; Decrement C.
        JNZ         LOOP        ; If C is not zero, then jump to
                                  LOOP.
        HLT                     ; Stop processing.
```

It may be noted from this example that DAA (decimal adjust the accumulator) instruction is used after ADC M for the conversion of the result in decimal form.

Example 6.30. *Write an ALP for 8085 for multibyte (N bytes) subtraction. The number of bytes (N) is stored in 2100 H location. The minuend bytes are stored in the memory locations starting at 2101 H and the subtrahend bytes are stored in the memory locations starting at 2201 H. The answer should be stored in the memory locations starting at 2101 H.*

Solution.

Main Program:

```
Label   Mnemonics  Operand     Comments
        LXI  H,     2100 H      ; Get H-L pair with 2100 H.
        MOV C,      M           ; Move the number N to C-register
                                  for counter.
        INX  H                  ; Increment the H-L pair.
        LXI  D,     2201 H      ; Get D-E pair with 2201 H.
        XRA  A                  ; Clear the accumulator and carry
                                  flag.
LOOP    LDAX D                  ; Get the addend byte to
                                  accumulator.
        SBB  M                  ; Subtract with borrow the contents
                                  of [M_{H-L}] from the accumulator
                                  contents.
        MOV  M,     A           ; Store the result in the location
                                  addressed by H-L pair.
        INX  D                  ; Increment the D-E pair.
        INX  H                  ; Increment the H-L pair.
```

```
        DCR  C                  ; Decrement C.
        JNZ      LOOP           ; If C is not zero, then jump to
                                  LOOP.
        HLT                     ; Stop processing.
```

Decimal Subtraction

It may be noted from the above example that SBB M (subtract with borrow) instruction is used for the subtraction. The result will not be obtained in the decimal form as DAA instruction can not be used after Subtract instructions. For decimal subtractions, the subtrahend number should be converted to its equivalent 10's complement which will then be added to minuend.

Example 6.31. *Write an ALP for 8085 to subtract 78 from 96. The answer in decimal form should be stored in 2100 H memory location.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	MVI C,	96 H	; Get 96 in C register.
	MVI B,	78 H	; Get 78 in B register.
	MVI A,	99 H	; Get 99 in accumulator.
	SUB B		; Subtract the content of B-register from 99 to get the 9's complement of 78.
	INR A		; Increment accumulator to get 10's complement.
	ADD C		; Add the content of C to the accumulator.
	DAA		; Decimal adjust the accumulator, to get the answer in decimal for.
	STA	2100 H	; Store the answer in decimal form at 2100 H location.
	HLT		; Stop processing.

Example 6.32. *Write an ALP for 8085 for multibyte (N bytes) decimal subtraction. The number of bytes (N) is stored in 2100 H location. The minuend bytes are stored in the memory locations starting at 2101 H and the subtrahend bytes are stored in the memory locations starting at 2201 H. The answer obtained in decimal form should be stored in the memory locations starting at 2101 H.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LXI SP,	XXXX H	; Initialize the stack pointer.
	LXI H,	2100 H	; Get H-L pair with 2100 H.
	MOV C,	M	; Move the number N to C-register for counter.
	INX H		; Increment the H-L pair.
	LXI D,	2201 H	; Get D-E pair with 2201 H.
	LDAX D		; Get the subtrahend byte in accumulator.

```
        MOV  B,     A       ; Move the accumulator content to B-
                              register.
        MVI  A,     99 H    ; Store 99 to accumulator.
        SUB  B              ; Get 9's complement of the
                              subtrahend.
        INR  A              ; Get 10's complement of the
                              subtrahend.
        ADD  M              ; Add 10's complement of the
                              subtrahend to minuend.
        DAA                 ; Give the answer in decimal form.
        PUSH PSW            ; Push the carry to stack.
        MOV  M,     A       ; Store the result in the location
                              addressed by H-L pair.
LOOP    INX  D              ; Increment the D-E pair.
        INX  H              ; Increment the H-L pair.
        LDAX D              ; Get the second number to
                              accumulator.
        MOV  B,     A       ; Store it to B-register.
        MVI  A,     99 H    ; Store 99 to accumulator.
        SUB  B              ; Get 9's complement.
        MOV  B,     A       ; Store it to B-register.
        POP  PSW            ; Get the carry of the previous
                              addition from the stack.
        MOV  A,     B       ; Get the B-register content to
                              accumulator.
        ADC  M              ; Add with carry.
        DAA                 ; Give the answer in decimal form.
        PUH  PSW            ; Store the carry to stack for further
                              addition.
        MOV  M,     A       ; Store the result in the memory
                              location addressed by H-L register
                              pair.
        DCR  C              ; Decrement the content of C-
                              register.
        JNZ         LOOP    ; If C is not zero, then jump to
                              LOOP.
        HLT                 ; Stop processing.
```

6.4 PROGAMS TO FIND LARGEST OR SMALLEST NUMBER

Example 6.33 *Write an ALP for 8085 to find the larger of two numbers stored in memory locations 2101 H and 2102 H. Store the result in memory location 2103 H.*

Solution.

Main Program:

```
Label   Mnemonics  Operand     Comments
        LXI  H,     2101 H  ; Get H-L pair with 2101 H.
        MOV  A,     M       ; Move the first number in
                              accumulator.
```

	INX H		; Increment the H-L pair.
	CMP M		; Compare the second number with first number.
	JNC	NXT	; If the accumulator content is larger then jump to NXT.
	MOV A,	M	; Else move $[M_{H-L}]$ to accumulator.
NXT	INX H		; Increment the H-L pair.
	MOV M,	A	; Store the result in the required memory location.
	HLT		; Stop processing.

Note: To get the smaller number from the given two numbers, use the instruction JC in place of JNC.

Example 6.34. *Write an ALP for 8085 to find the largest number among three numbers stored in memory locations staring at 2101 H. Store the result in memory location 2104H.*

Solution.

Main Program:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI H,	2101 H	; Get H-L pair with 2101 H.
	MOV A,	M	; Move the first number in accumulator.
	INX H		; Increment the H-L pair.
	CMP M		; Compare the second number with first number.
	JNC	NXT	; If the accumulator content is larger then jump to NXT.
	MOV A,	M	; Else move $[M_{H-L}]$ to accumulator.
NXT	INX H		; Increment the H-L pair.
	CMP M		; Compare the third number with the larger of first two numbers.
	JNC	NXT1	; If the accumulator content is larger then jump to NXT1.
	MOV A,	M	; Else move $[M_{H-L}]$ to accumulator.
NXT1	INX H		; Increment the H-L pair.
	MOV M,	A	; Store the result in the required memory location.
	HLT		; Stop processing.

Example 6.35. *Write an ALP for 8085 to find the largest number among a series of N numbers stored in memory locations staring at 2101 H. The number N is stored in memory location 2100 H. Store the result in memory location 2201 H.*

Solution.

Main Program:

Label	**Mnemonics**	**Operand**	**Comments**

```
        LXI  H,     2100 H    ; Get H-L pair with 2100 H.
        MOV  C,     M         ; Store the number N in C-register
                              which will be used as the counter.
        INX  H                ; Increment the H-L pair.
        MOV  A,     M         ; Move the first number in
                              accumulator.
        DCR  C                ; Decrement count.
LOOP    INX  H                ; Increment the H-L pair.
        CMP  M                ; Compare the second number with
                              first number.
        JNC         NXT       ; If the accumulator content is larger
                              then jump to NXT.
        MOV  A,     M         ; Else move [M_{H-L}] to accumulator.
NXT     DCR  C                ; Decrement count.
        JNZ         LOOP      ; Jump to LOOP if not zero.
        STA         2201 H    ; Store the result in 2201 H.
        HLT                   ; Stop processing.
```

6.5 PROGAMS TO ARRANGE A GIVEN SERIES IN ASCENDING OR DESCENDING ORDER

Example 6.36. *Write an ALP for 8085 to arrange a series of N-numbers in descending order. The number N is stored in memory location 2100 H. The series is stored in memory location starting at 2101 H. The result is to be stored in memory locations starting at 2201 H.*

Solution.

Main Program:

```
Label   Mnemonics  Operand   Comments
        LXI  SP,    XXXX H    ; Initialize stack pointer.
        LXI  D,     2201 H    ; Get D-E pair with 2201 H.
        LXI  H,     2100 H    ; Get H-L pair with 2100 H.
        MOV  B,     M         ; Store the count N in B-register.
START   LXI  H,     2100 H    ; Get H-L pair with 2100 H.
        MOV  C,     M         ; Store the number N in C-register
                              also.
        INX  H                ; Increment the H-L pair.
        MOV  A,     M         ; Move the first number in
                              accumulator.
        DCR  C                ; Decrement count.
LOOP    INX  H                ; Increment the H-L pair.
        CMP  M                ; Compare the second number with
                              first number.
        JNC         NXT       ; If the accumulator content is larger
                              then jump to NXT.
        MOV  A,     M         ; Else move [M_{H-L}] to accumulator.
NXT     DCR  C                ; Decrement count.
        JNZ         LOOP      ; Jump to LOOP if not zero.
```

```
        STAX  D                 ; Store the largest number in
                                  memory location addressed by D-E
                                  register pair.
        CALL    SUBR            ; Call the subroutine SUBR to put
                                  00 H to the memory location where
                                  the largest number was found.
        INX  D                  ; Increment D-E register pair.
        DCR  B                  ; Decrement B.
        JNZ     START           ; Jump to START if not zero.
        HLT                     ; Stop processing.
```

Subroutine Program:

Label	Mnemonics	Operand	Comments
SUBR	LXI H,	2100 H	; Get H-L pair with 2100 H.
	MOV C,	M	; Store the number N is C-register which will be used as the counter.
AGAIN	INX H		; Increment the H-L pair.
	CMP M		; Compare the number with the largest number obtained in the main program.
	JZ	NXT	; If it is largest jump to NXT.
	DCR C		; Decrement count.
	JNZ	AGAIN	; If counts not complete jump to AGAIN.
	MVI A,	00 H	; Store 00 H to the accumulator.
	MOV M,	A	; Put 00 H to the location at which the largest number was found.
	RET		; Go back to main program.

Example 6.37. *Write an ALP for 8085 to arrange a series of N-numbers in ascending order. The number N is stored in memory location 2100 H. The series is stored in memory location starting at 2101 H. The result is to be stored in memory locations starting at 2201 H.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LXI SP,	XXXX H	; Initialize stack pointer.
	LXI D,	2201 H	; Get D-E pair with 2201 H.
	LXI H,	2100 H	; Get H-L pair with 2100 H.
	MOV B,	M	; Store the count N in B-register.
START	LXI H,	2100 H	; Get H-L pair with 2100 H.
	MOV C,	M	; Store the number N in C-register also.
	INX H		; Increment the H-L pair.
	MOV A,	M	; Move the first number in accumulator.
	DCR C		; Decrement count.
LOOP	INX H		; Increment the H-L pair.

```
        CMP  M              ; Compare the second number with
                              first number.
        JC       NXT        ; If the accumulator content is
                              smaller then jump to NXT.
        MOV  A,  M          ; Else move [M H-L] to accumulator.
NXT     DCR  C              ; Decrement count.
        JNZ      LOOP       ; Jump to LOOP if not zero.
        STAX  D             ; Store the largest number in
                              memory location addressed by D-E
                              register pair.
        CALL     SUBR       ; Call the subroutine SUBR to put
                              FF H to the memory location
                              where the smallest number was
                              found.
        INX  D              ; Increment D-E register pair.
        DCR  B              ; Decrement B.
        JNZ      START      ; Jump to START if not zero.
        HLT                 ; Stop processing.
```

Subroutine Program:

Label	Mnemonics	Operand	Comments
SUBR	LXI H,	2100 H	; Get H-L pair with 2100 H.
	MOV C,	M	; Store the number N is C-register which will be used as the counter.
AGAIN	INX H		; Increment the H-L pair.
	CMP M		; Compare the number with the smalest number obtained in the main program.
	JZ	NXT	; If it is smallest jump to NXT.
	DCR C		; Decrement count.
	JNZ	AGAIN	; If counts not complete jump to AGAIN.
	MVI A,	FF H	; Store FF H to the accumulator.
	MOV M,	A	; Put FF H to the location at which the smallest number was found.
	RET		; Go back to main program.

6.6 PROGRAMS ON MULTIPLICATION

There are two methods for multiplication of two numbers.

1. Repetitive Addition Method
2. Shift and Add Method

Repetitive Addition Method

In this method multiplicand is added by the multiplier times.

For example, we wish to multiply 8 and 4 numbers. So add 08 to 00 for four times. i.e.

```
    0 0
+   08          1 time
-------------------------
    08
+   08          2 times
-------------------------
    16
+   08          3 times
-------------------------
    24
+   08          4 times
-------------------------
    32          Answer
-------------------------
```

Shift and Add Method

For example, we wish to multiply two decimal numbers 32 and 12 as:

```
  3 2     Multiplicand
x 1 2     Multiplier
-----------------------
  6 4
3 2       Shift by one digit
-----------------------
3 8 4     Answer
-----------------------
```

In this example, first of all multiplicand 32 is multiplied by 2 and the result 64 is copied in the temporary register (or written on the scratch pad). Again 32 is multiplied by 1 and result 32 is shifted left by one digit to make it 32o. Now 64 and 320 are added and we get the answer 384.

Similar method used in binary multiplication. In binary multiplication, when a multiplicand is multiplied by 1, we get the product equal to multiplicand. When a multiplicand is multiplied by 0, we the product zero. For example 08 and 04 are multiplied as:

```
      0 0 0 0 1 0 0 0
    X 0 0 0 0 0 1 0 0
-----------------------
      0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0
  0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
-----------------------
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
-----------------------
```

The answer is 0020 H (32_{10}). It may be noted that the answer is not in BCD.
This method may further be illustrated using the following flow chart.

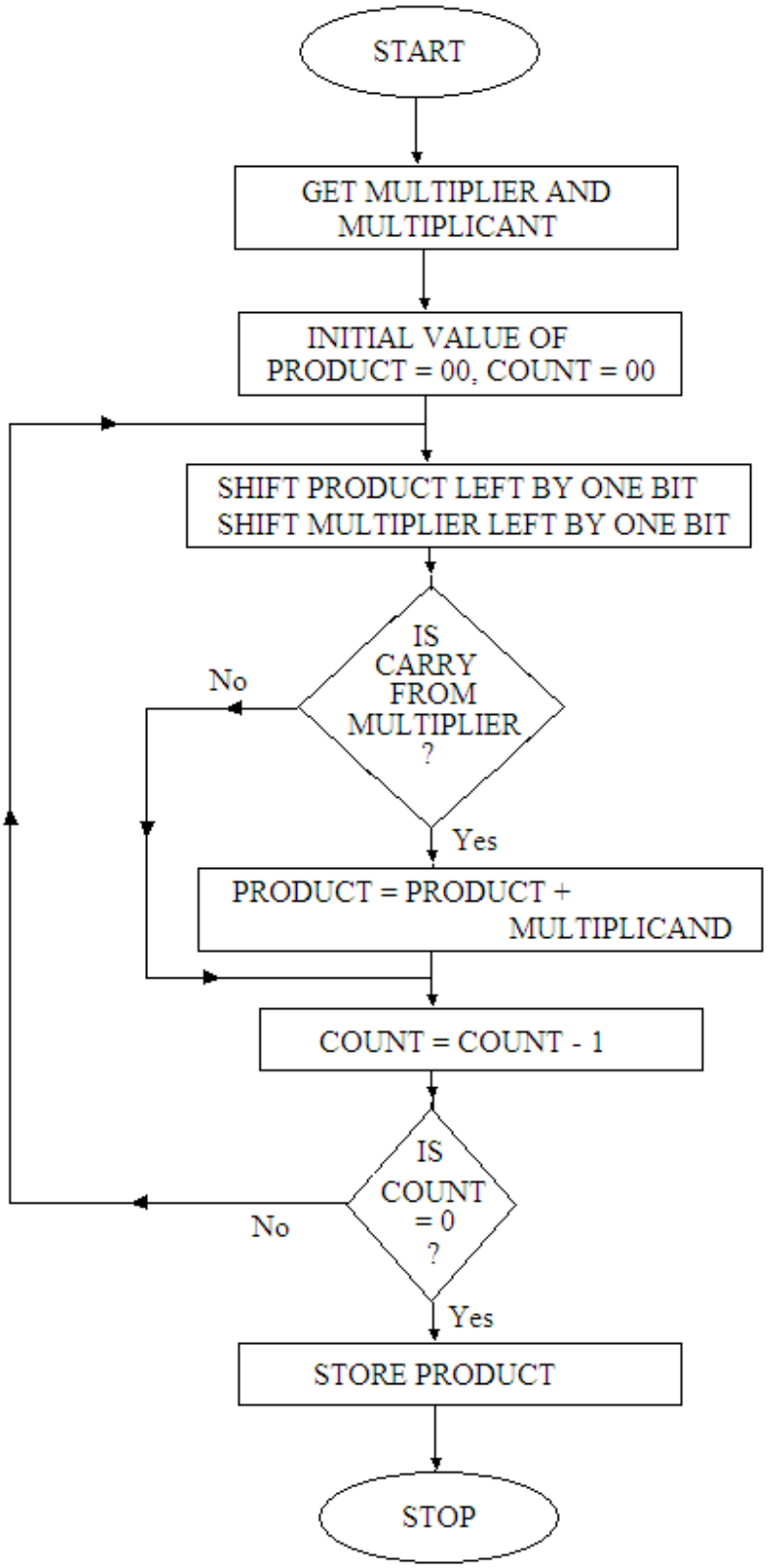

START
GET MULTIPLIER AND MULTIPLICANT
INITIAL VALUE OF PRODUCT = 00, COUNT = 00
SHIFT PRODUCT LEFT BY ONE BIT
SHIFT MULTIPLIER LEFT BY ONE BIT
IS CARRY FROM MULTIPLIER ?
No
Yes
PRODUCT = PRODUCT + MULTIPLICAND
COUNT = COUNT - 1
IS COUNT = 0 ?
No
Yes
STORE PRODUCT
STOP

Fig. 6.1

Example 6.38. *Write an ALP for 8085 to multiply any number (stored in memory location 2101 H) by 2 and store the result in 2102 H memory location.*

Solution.

Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2101 H	; Get the H-L pair with 2101 H.
	MOV A,	M	; Move the number in accumulator.
	STC		; Set the carry flag.
	CMC		; Complement the carry (it clear the carry.
	RAL		; Rotate accumulator with carry i.e. it multiplies the accumulator content by two.
	INX H		; Increment the H-L pair.
	MOV M,	A	; Store the result in 2102 H memory location.
	HLT		; Stop processing.

Example 6.39. *Write an ALP for 8085 to multiply two numbers using repetitive addition method. The two numbers are in memory locations 2101 H and 2102 H. Store the result in 2103 H and 2104 H.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LXI H,	2101 H	; Get H-L pair with 2101 H.
	MOV B,	M	; Get first number in B-register.
	INX H		; Increment H-L pair.
	MOV A,	M	; Get the second number in accumulator.
	CMP B		; Compare the two numbers.
	JNC	NXT	; If A < B then use A as counter.
	MVI D,	00 H	; Move 00 to D register.
	MOV E,	B	; Move first number to E register.
	JMP	NXT1	; Jump to NXT1.
NXT	MVI D,	00 H	; If A>B, so B should be used as counter.
	MOV E,	A	; Store accumulator content in E-register.
	MOV A,	B	; Store B-register content in accumulator.
NXT1	LXI H,	0000 H	; Put 00 to H and L registers.
	ORA A		; ORing of A with A to find if A = 00 H.
	JZ	END	; If zero jump to End.
	DAD D		; Add the contents of H-L pair with D-E pair and the result is in H-L pair.

```
        DCR  A                  ; Decrement count.
        JNZ      LOOP           ; If not zero jump to LOOP.
END     SHLD     2103 H         ; Store the result.
        HLT                     ; Stop processing.
```

Example 6.40. *Write an ALP for 8085 to multiply two numbers using shift and add method. The two numbers are in memory locations 2101 H and 2102 H. Store the result in 2103 H and 2104 H.*

Solution.

Main Program:

```
Label   Mnemonics   Operand     Comments
        LXI  H,     2101 H      ; Get H-L pair with 2101 H.
        MOV  E,     M           ; Get first number in E-register.
        MVI  D,     00 H        ; Extend to 16 bits.
        INX  H                  ; Increment H-L pair.
        MOV  A,     M           ; Get the multiplier in accumulator.
        LXI  H,     0000 H      ; Put 00 to H and L registers.
        MVI  B,     08 H        ; Get count = 08.
MULT    DAD  H                  ; Multiplicand = 2 x Multiplicand..
        RAL                     ; Rotate accumulator left to find if
                                  most significant bit is 1.
        JNC         NXT         ; If no carry jump to NXT.
        DAD  D                  ; Get Product = product +
                                  multiplicand.
NXT     DCR  B                  ; Decrement B.
        JNZ         MULT        ; If not zero jump to MULT.
        SHLD        2103 H      ; Store the result in 2103 H and
                                  2104 H .
        HLT                     ; Stop processing.
```

6.7 PROGRAM ON 8-BIT DIVISION

To divide 16 bit dividend with 8 bit divisor, the divisor is subtracted from the 8 most significant bits of the dividend. If there is no borrow, the bit of the quotient is set to 1, otherwise 0. The dividend is then shifted left by one bit before each trial of subtraction. The dividend and quotient share a register pair. Due to shift of dividend one bit of the register falls vacant in each subtraction. The quotient is stored in vacant bit positions. It is illustrated by the flow chart shown in figure 6.2.

If the dividend of the division is an 8-bit number, then it is extended to a 16-bit number by placing zeros in the MSBs positions.

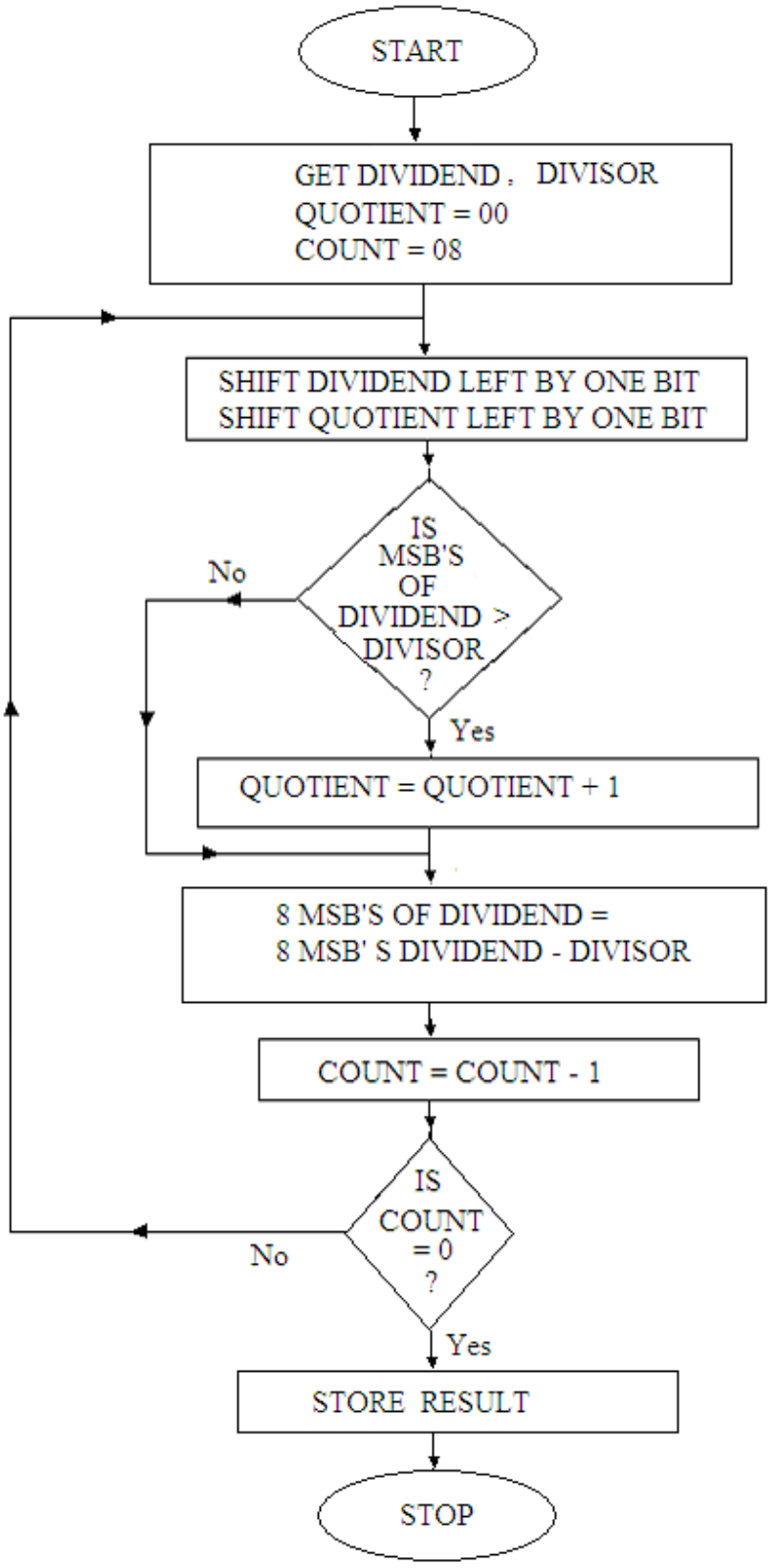

START
GET DIVIDEND , DIVISOR
QUOTIENT = 00
COUNT = 08
SHIFT DIVIDEND LEFT BY ONE BIT
SHIFT QUOTIENT LEFT BY ONE BIT
IS MSB'S OF DIVIDEND > DIVISOR ?
No
Yes
QUOTIENT = QUOTIENT + 1
8 MSB'S OF DIVIDEND =
8 MSB' S DIVIDEND - DIVISOR
COUNT = COUNT - 1
IS COUNT = 0 ?
No
Yes
STORE RESULT
STOP

Fig. 6.2

Example 6.41. *Write an ALP for 8085 to divide a 16-bit number by an 8-bit number. The dividend bytes are stored in memory locations 2101 H and 2102 H. (LS byte in 2101 H and MS byte in 2102 H). The divisor is stored in memory location 2103 H. The quotient is to be stored in the memory location 2104 H and the remainder is to be stored in the memory location 2105 H.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LHLD	2101 H	; Get 16-bit dividend in H-L pair.
	LDA	2103 H	; Get divisor in accumulator.
	MVI C,	08 H	; Get count 08 in C-register.
	MVI D,	00 H	; Quotient = 00.
	MOV B,	A	; Keep the divisor in B-register also.
LOOP	DAD H		; Shift dividend left by one bit.
	MOV A,	D	; Keep the quotient in accumulator.
	ADD A		; Shift quotient left by one bit.
	MOV D,	A	; Store the quotient in D-register.
	SUB B		; Perform trial subtraction.
	JC	NXT	; If carry then dividend = previous dividend.
	MOV H,	A	; Put most significant bits of dividend in register H.
	INR D		; Increment quotient by 1.
NXT	DCR C		; Decrement C.
	JNZ	LOOP	; If not zero jump to LOOP.
	MOV L.	D	; Store the quotient in L-register.
	SHLD	2104 H	; Store the result in 2104 H and 2105 H .
	HLT		; Stop processing.

6.8 MISCELLANEOUS PROGRAMS

Example 6.42. *Write an ALP for 8085 to find the square of a given number stored in memory location 2101 H and the result is to be stored in memory location 2102 H. Use Look-up table starting at 2200 H.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LXI D,	2200 H	; Get D-E pair with 2200 H (Starting address of the look-up table).
	LDA	2101 H	; Get the number in accumulator.
	MOV L,	A	: Load the number to L-register.
	MVI H,	00 H	; Get 00 H to H-register.
	DAD D		; Get effective address in H-L register pair.
	MOV A,	M	; Get the result in accumulator.

	STA	2102 H	; Store the result in the required memory location.
	HLT		; Stop processing.

Look-up table:

Location	Data
2200 H	00 H
2201 H	01 H
2202 H	04 H
2203 H	09 H
2204 H	10 H
2205 H	19 H
2206 H	24 H
2207 H	31 H
2208 H	40 H
2209 H	51 H
220A H	64 H
220B H	79 H
220C H	90 H
220D H	A9 H
220E H	C4 H
220F H	E1 H

Example 6.43. *Write an ALP for 8085 to find the square of a given number stored in memory location 2101 H and the result is to be stored in memory location 2102 H. Use the addition of successive odd integers.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LDA	2101 H	; Get the number in accumulator.
	MVI L,	00H	: Load 00 H to L-register.
	ORA L		; Find if number is zero.
	JZ	END	; If number is zero then no need to find the square and jump to END.
	MOV C,	A	; Move the number to C-register.
	MOV A,	L	; Move the content of L-reg to accumulator so that square is zero.
	MVI B,	01 H	; Move 01 to B-register (first odd number).
LOOP	ADD B		; Add next odd number.
	INR B		; Find next odd number to add
	INR B		; 02.
	DCR C		; Decrement C-register contents.
	JNZ	LOOP	; Continue if not zero.
END	STA	2102 H	; Store the result in the memory location 2102 H.

```
        HLT                         ; Stop processing.
```

Example 6.44. *Write an ALP for 8085 to find the square root of a given number (perfect positive square) stored in memory location 2101 H and the result is to be stored in memory location 2102 H. Use the subtraction of successive odd integers.*

Solution.

Main Program:

Label	Mnemonics	Operand	Comments
	LDA	2101 H	; Get the number in accumulator.
	ORA A		; Find if number is zero.
	JZ	END	; If number is zero then no need to find the square root and jump to END.
	MVI B,	01 H	; Move 01 to B-register (first odd number).
	MVI C,	01 H	; Initial value of square root as one.
LOOP	SUB B		; Subtract next odd number.
	JZ	END	; If number is zero then jump to END.
	INR B		; Find next odd number to add
	INR B		; 02.
	INR C		; Decrement square root value by 1.
	JNC	LOOP	; Continue if no carry.
END	MOV A,	C	; Store the result in accumulator.
	STA	2102 H	; Store the result in the memory location 2102 H.
	HLT		; Stop processing.

Example 6.45. *Write an ALP for 8085 to find the Fibonacci series. Sixteen terms of this series are to be store in the memory locations starting at 2101 H. Let 00 H and 01 H data are stored in memory locations 2101 H and 2102 H respectively before the execution of the program.*

Solution. The terms of Fibonacci series are given as:

1, 1, 2, 3, 5, 8, 13, 21,. . . . .

This sequence is defined by assuming first two terms have the same value 1, and each term afterwards is the sum of the previous two terms, that is,

1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5, 3 + 5 = 8, and so on.

The required terms of the

Main Program:

Label	Mnemonics	Operand	Comments
	MVI C,	10 H	; Decimal number 16 (10 H) is stored in C-register which will be used as counter.
	LXI H,	2100 H	; Get H-L pair with 2100 H.
	MOV A,	M	; Move first data to accumulator.
	INX H		; Point to next data.
LOOP	ADD M		; Get next term of the series.

```
INX  H                  ; Point to the next memory location
                          for storing the next term.
MOV  M,     A           ; Store the next term.
DCX  H                  ; Get previous term of the Fibonacci
                          series.
MOV  A,     M           ; Get this term in accumulator.
INX  H                  ; Point to the next memory location.
DCR  C                  ; Decrement count.
JNZ         LOOP        ; If not zero jump to LOOP.
HLT                     ; Stop processing.
```

Data stored before the execution of the program.

2100 H	00 H
2101 H	01 H

PROBLEMS

1. Write an assembly language program of 8085 to fill the RAM area from 2500 H to 25FF H with a byte 33 H.
2. Sixteen bytes of data are stored in memory locations 2001 H to 2010 H. Write an assembly language program of 8085 to transfer this block of data to 2006 H to 2015 H.
   (Hint: In this case data will be transferred in the reverse order otherwise some of the data will coincide.)
3. Write a program (WAP) in assembly language of 8085 to compare two hexadecimal numbers, stored in 2001 H and 2002 H memory locations, for equality. If the numbers are equal, the number itself will be stored in 2003 H memory location, else 00 H will be stored in the memory location 2003 H.
4. WAP in assembly language of 8085 to test 3rd bit (D_3 bit) of a byte stored at 2000 H memory location. If the bit D3 is zero store 00 at 2101 H else store the same number at 2101 H.
5. Write an assembly language program (ALP) of 8085 to find the logical AND and Logical OR of 26 H and 39 H. Store the result in 2500 H and 2501 H.
6. Write an ALP of 8085 to load EE H to the memory locations starting at 2501 H. The length of data bytes is given in 2500 H memory location.
7. Write an ALP for 8085 to add two 8-bit numbers stored in memory locations 2101 H and 2102 H. The result should be stored in 2103 H and the carry if any should be stored in 2104 H.
8. Write an assembly language program for 8085 to add two 8 bit numbers stored in memory locations 2101 H and 2102 H. The answer is required in decimal form and it should be stored in 2103 H and the carry if any should be stored in 2104 H.
9. Write an ALP for 8085 to add 833 and 776 decimal numbers. The result should be stored in 2101 H to 2103 H memory locations. The memory location 2103 H should store the carry bit if any.
   (Hint: First convert the decimal numbers 833 and 776 to hexadecimal numbers.)
10. Write an ALP for 8085 to subtract an 8 bit number (stored in 2101 H memory location) from another 8 bit number (stored in 2102 H memory location). The

answer should be stored in 2103 H and the borrow bit if any should be stored in 2104 H.

11. Repeat the problem 10 to get the result in decimal form.
12. Write an ALP for 8085 to find smaller of two numbers stored in 2301 H and 2302 H. Store the smaller number in memory location 2303 H.
13. Write an ALP for 8085 to find the smallest of the three numbers stored in memory locations starting at 2301 H. Store he smallest number in 2304 H.
14. Write an ALP for 8085 to find the smallest number among a series of 16 numbers stored in the memory locations starting at 2301 H. The number 16 (10 H) is stored in 2300 H. The smallest number should be should be stored in 2401 H.
15. Write an ALP to find 13 terms of Fibonacci series. The terms of the series are to be stored in the memory locations starting at 2501 H.
16. Write an ALP for 8085 to shift an 8-bit number left by two bits. The number is stored in memory location 2501 H and the result is to be stored in 2502 H memory location.
    (Hint: Keep the number in accumulator and use ADD A instruction twice.)
17. Write an ALP for 8085 to shift a 16-bit number left by two bits. The number is stored in memory locations 2501 H and 2502 H. The result is to be stored in 2503 H and 2504 H memory locations.

7

Interrupt Instructions of 8085

In the preceding chapters the architecture, instruction set and programming details of 8085 microprocessor were discussed. The microprocessor forms the parts of microcomputers. Many complicated and sophisticated operations can be performed by using the microprocessor based systems. These systems have the facilities to transfer the data from the microprocessor to the outside world or vice-versa. The transfer of data takes place through the data bus with the help of address and control buses. Thus special circuitry is needed to convert data from a wide range of input devices into suitable form for microprocessor buses. The circuitries used between the microprocessor and input/output devices are known as interfaces. This chapter deals how the external devices (peripherals) get connected to the microprocessor. In addition, different ways of data transfer from the microprocessor to the peripheral devices and vice-versa, and the restart instructions will also be discussed.

7.1 METHODS OF I/O OPERATIONS

There are two ways for the transfer of data from microprocessor to I/O devices and vice-versa.

1. Memory Mapped I/O
2. I/O Mapped I/O or Isolated I/O

7.1.1 Memory Mapped I/O

Memory Mapped I/O system is used in smaller system. In this system there is only one address space, some addresses are assigned to memories and some addresses to I/O devices i.e. the addresses to I/O devices are different from the addresses which have been assigned to memories. In other words the I/O devices and memory share the memory map, with some address reserved for the I/O devices and rest for the memory. This scheme is shown in figure 7.1. The microprocessor use R/W signals to determine the direction of data flow. The main advantage of this scheme is that it does not require additional decoding circuitry and the set of instructions may be used to fetch data either from the I/O devices or from the memory locations.

Following instructions may be used for the data transfer in this scheme:

MOV M, A — It moves the contents of accumulator to M_{H-L}. If H-L pair contains the address of memory location, then the data will be transferred to memory location; if on the other hand H-L pair contains the address of the I/O devices, then the accumulator data will be transferred to I/O devices.

MOV A, M — It moves the contents of M_{H-L} to accumulator. If H-L pair contains the address of memory location, then the data will be transferred from memory location; similarly if H-L pair contains the address of the I/O devices, then the data from I/O devices will be transferred to accumulator.

STA address — It stores the contents of accumulator to addressed location. If the address represents the address of memory location then the accumulator contents will be stored to memory location; if on the other hand address represents the address of the I/O devices, then the accumulator data will be transferred to I/O devices.

LDA address — It stores the addressed contents to the accumulator. If the address represents the address of memory location then the contents stored in memory location will be transferred to accumulator; if on the other hand address represents the address of the I/O devices, then the data from the I/O devices will be transferred to the accumulator.

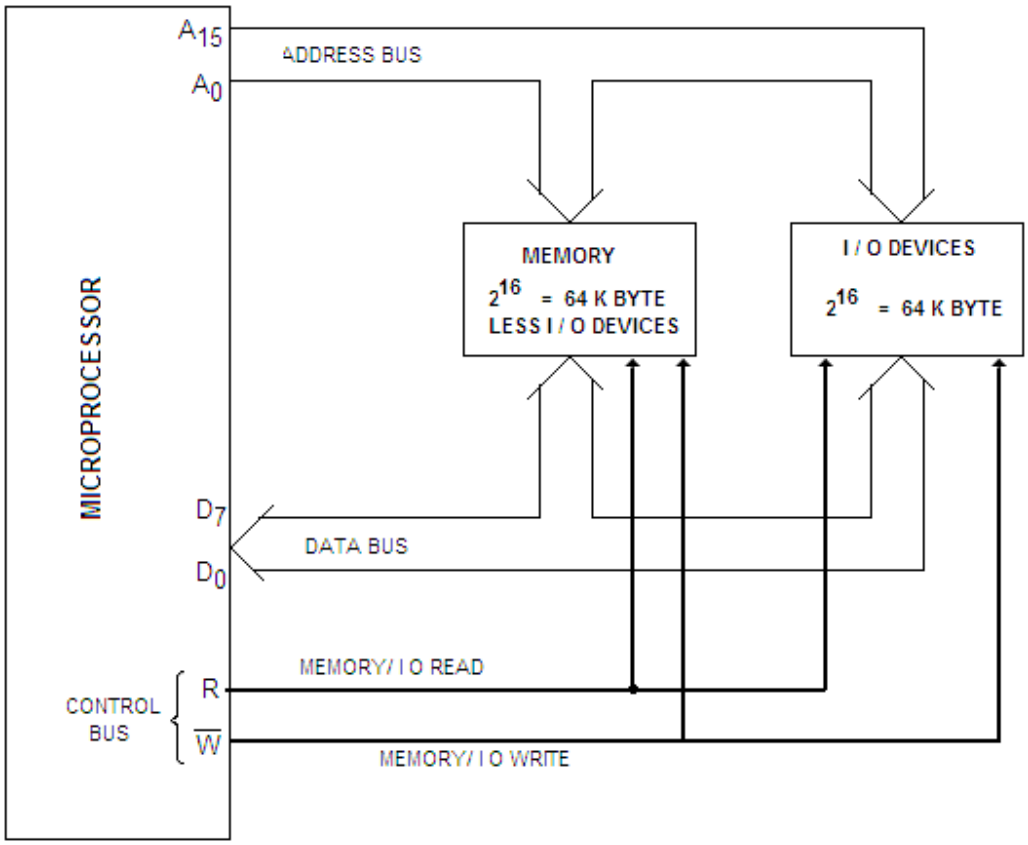


Fig. 7.1

Many other instructions like LDAX, STAX and other arithmetic and logic instructions etc may also be used in this memory mapped I/O systems.

7.1.2 I/O Mapped I/O or Isolated I/O

Larger systems use this technique for the data communication to I/O devices. Figure 7.2 shows this system used in 8085 microprocessor. In this technique the

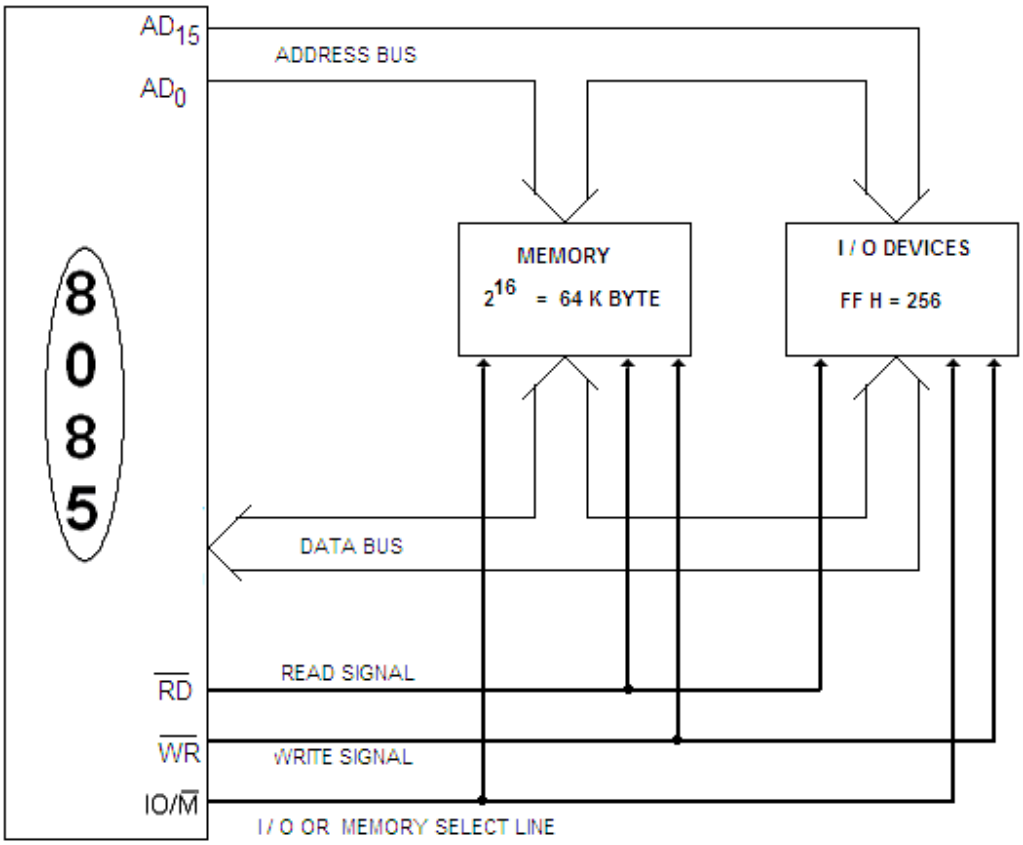


Fig 7.2

microprocessor treats the memory and I/O devices separately, using different control lines for each of them. In 8085A the $IO/\overline{M}$ signal is used for this purpose. If this signal is low (0), it represents the memory. However, if this signal is high (1), it represents the I/O operations. $\overline{RD}$ and $\overline{WR}$ signals in association with $IO/\overline{M}$ signal help in performing I/O read/write operations or memory read/write operations. Table 7.1 shows the operations of these signals.

Table 7.1

$\overline{RD}$	$\overline{WR}$	$IO/\overline{M}$	Operations	Address
0	1	0	$\overline{MEMR}$ Memory Read	$\mathbf{A_{15}}$-$\mathbf{A_0}$
1	0	0	$\overline{MEMW}$ Memory Write	$\mathbf{A_{15}}$-$\mathbf{A_0}$
0	1	1	$\overline{IOR}$ I/O Read	$\mathbf{A_7}$-$\mathbf{A_0}$
1	0	1	$\overline{IOW}$ I/O Write	$\mathbf{A_7}$-$\mathbf{A_0}$

Figure 7.3 shows the generation of $\overline{MEMR}$ (Memory Read), $\overline{MEMW}$ (Memory Write), $\overline{IOR}$ (I/O Read) and $\overline{IOW}$ (I/O Write) signals.

In this scheme, special instructions are used to perform the I/O operations. The special instructions are:

IN XX H (XX H is the Port address of 1 byte , which is in between 00 H to FF H)

OUT XX H (XX H is the Port address of 1 byte , which is in between 00 H to FF H)

The address XX H of the I/O port is duplicated on both the high and low order address bus and the signal $IO/\overline{M}$ ensures that the memory location corresponding to that address (XXXX H) does not respond.

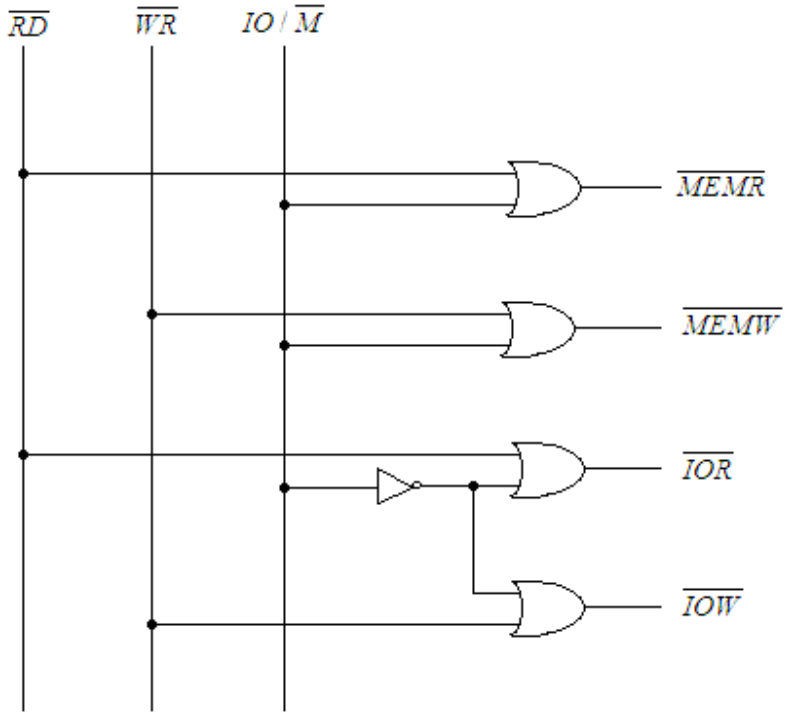


Fig. 7.3

Comparison of Memory Mapped I/O and I/O Mapped I/O is given in table 7.2.

Memory Mapped I/O	I/O Mapped I/O
1. Large Number of instructions like MOV A, M, MOV M, A, LDA, STA etc are used for the data transfer.	Only two instructions IN Port, OUT Port are used for the data transfer from microprocessor to I/O devices and vice versa.
2. Data transfer is possible between any register of C.P.U. and I/O devices.	Data transfer is possible only between Accumulator and I/O devices.
3. Arithmetic and logic operations can be performed with the data of I/O Ports using with the instructions like ORA M, XRA M, ANA M, ADD M, SUB M etc.	Arithmetic and logic operations can not be performed in this scheme.

4. Memory mapping of 64 K byte is shared between system memory and I/O ports.	256 (FF H) I/O devices may be interfaced.
5. 16-bit address lines are needed.	8-bit address lines are needed.
6. It takes different T-states depending on the instructions.	It takes 10 T-states for its execution.

7.2 DATA TRANSFER SCHEMES

In microprocessor based systems several input / output devices are connected and data transfer may take place between microprocessor and memory, microprocessor and I/O devices and memory & I/O devices. Not much of the problems arise for the data communication between microprocessor and memory as same technology is used in the manufacturing of memory and microprocessor. The speed of the memory is almost compatible with the speed of microprocessor. But for the data transfer between the microprocessor and I/O devices, the problems arise due to mismatch of the speed of the I/O devices and the speed of microprocessor or memory. To overcome the problem of speed mismatch between the microprocessor and I/O devices following data transfer schemes (fig. 7.4) may be considered. The data transfer schemes were categorized depending upon the capabilities of I/O devices for accepting serial or parallel data.

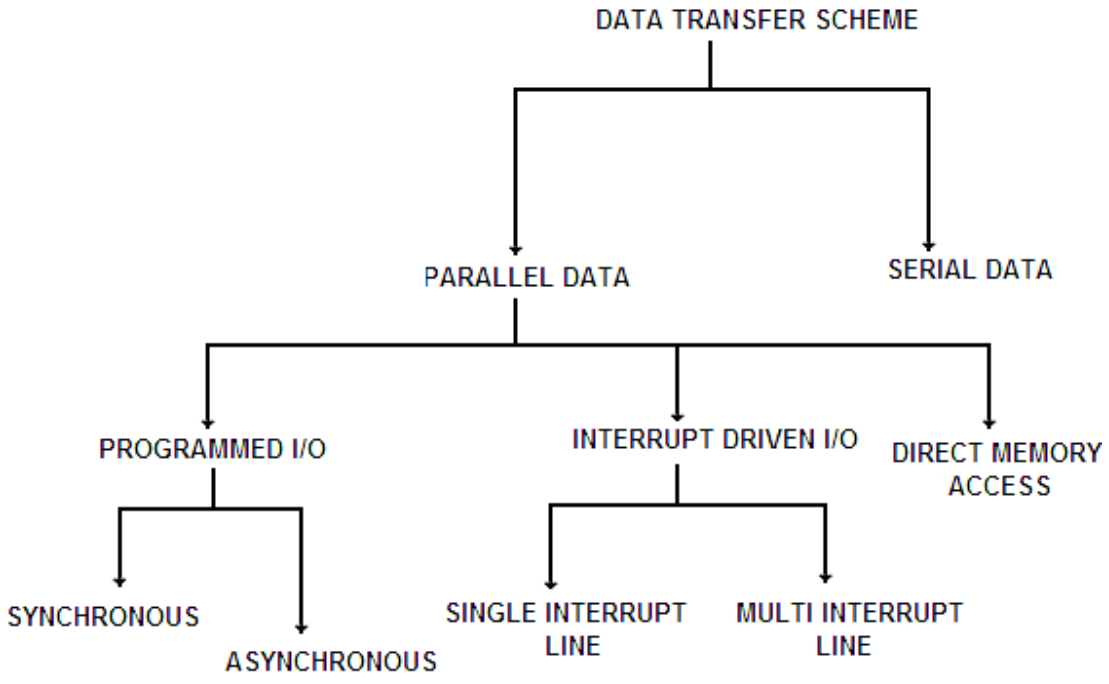


Fig. 7.4

The 8085 microprocessor is a parallel device i.e. it transfers eight bits of data simultaneously over eight data lines (parallel I/O mode). However in many situations, the parallel I/O mode is either impractical or impossible. For example, parallel data communication over a long distance becomes very expensive. Similarly, parallel data communication is not possible with devices such as CRT terminal or Cassette tape etc. For these devices, therefore, serial I/O mode is used which transfer a single bit on a single line at a tine. For serial data transmission, 8-bit parallel word is converted to a stream of eight serial bit using parallel-to-serial conversion. Similarly, in serial reception of data, the microprocessor receives a stream of 8-bit one by one which are then converted to 8-bit parallel word using serial-to-parallel conversion. The parallel data transfer will now be discussed in the following sub-sections.

7.2.1 Programmed I/O Data Transfer

This method of data transfer is generally used in the simple microprocessor systems where speed is unimportant. This method uses instructions to get the data into or out of the microprocessor. The data transfer can be synchronous or asynchronous depending upon the type and the speed of the I/O devices.

Synchronous type of data transfer can be used when the speed of the I/O devices matches with the speed of the microprocessor. The common clock pulse synchronizes the microprocessor and the I/O devices. In such data transfer scheme because of the matching of the speed, the microprocessor does not have to wait for the availability of the data; the microprocessor immediately sends data for the transfer as soon as the microprocessor issues a signal.

The asynchronous data transfer method is used when the speed of the I/O devices is slower than the speed of the microprocessor. Because of the mismatch of the speed, the

internal timing of the I/O device is independent from the microprocessor and thus two

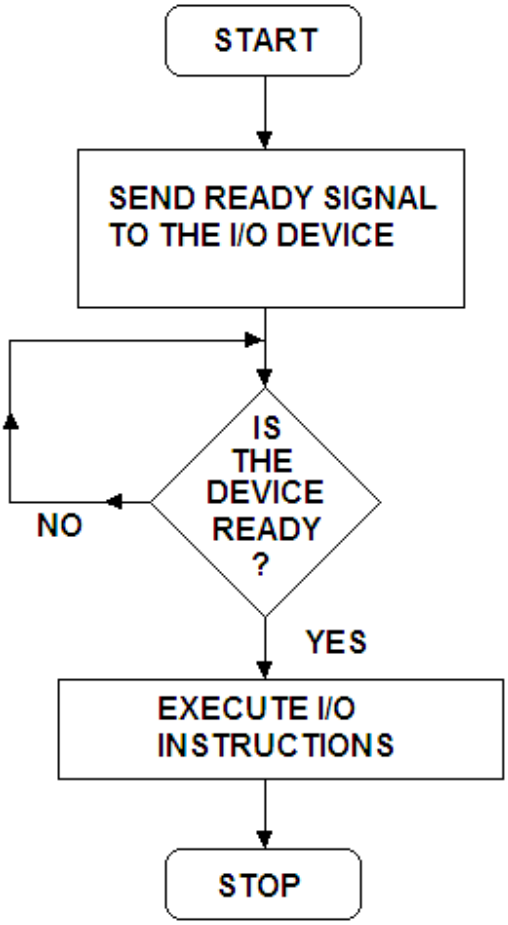


Fig. 7.5

units are said to be asynchronous to each other. The asynchronous data transfer is normally implemented using 'handshaking' mode. In the handshaking mode some signals are exchanged between the I/O device and microprocessor before the data transfer takes place. The microprocessor has to check the status to the input/output device, if the device is ready for the data transfer. The microprocessor initiates the I/O device to get ready; the status of the I/O device is continuously checked by the microprocessor till the I/O device becomes ready, the microprocessor sends instructions to transfer the data. Flow chart for this mode of data transfer is shown in figure 7.5.

Fig. 7.6 illustrates the asynchronous handshaking process to transfer the data from the microprocessor to I/O device. In this figure, the microprocessor sends a ready signal to I/O device. When the device is ready to accept the data, the I/O device sends an 'ACK' (Acknowledge) signal to microprocessor indicating that the I/O device has acknowledged the 'Ready' signal and is ready for the transfer of data.

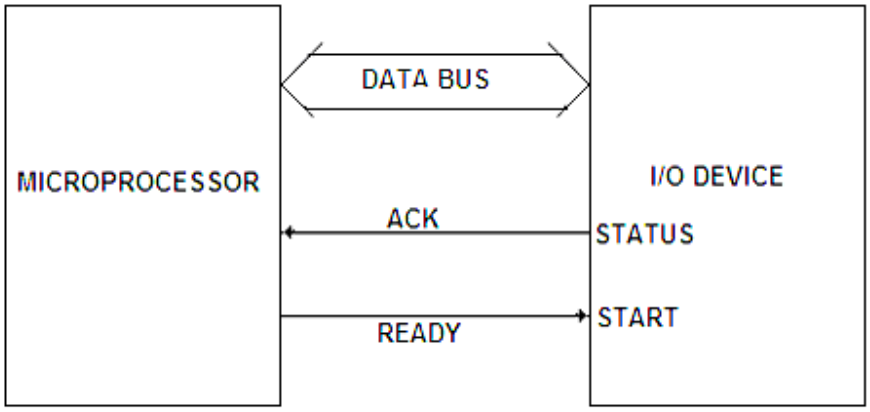


Fig. 7.6

Figure 7.7 shows the asynchronous handshaking process to transfer the data from the I/O device to microprocessor. In this case I/O device issues the ready signal to microprocessor indicating that I/O device is ready to send the data to microprocessor. In response to this signal, valid data signal is sent by the microprocessor to I/O device and then the valid data is put on the data bus for the transfer.

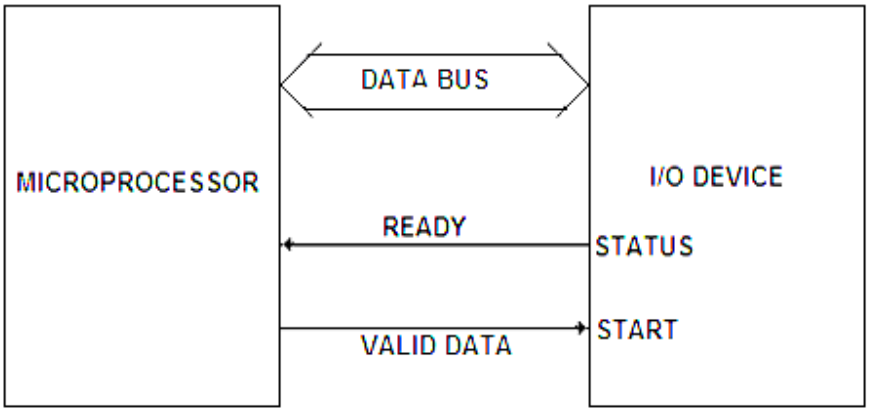


Fig. 7.7

7.2.2 Interrupt Driven I/O Data Transfer

In the programmed I/O data transfer method discussed above, the microprocessor is busy all the time in checking for the availability of data from the slower I/O devices and also in checking if I/O device is ready for the data transfer. In other words in this data transfer scheme, some of the microprocessor time is wasted in waiting while an I/O device is getting ready. The interrupt driven I/O data transfer method is efficient as no microprocessor time is wasted in waiting for an I/O device to be ready. The I/O device informs the microprocessor for the data transfer whenever the I/O device is ready. This is achieved by interrupting the microprocessor. The interrupt is hardware facilities provided on the microprocessor. In the beginning the microprocessor initiates data transfer by requesting the I/O device 'to get ready' and then continue executing its original program rather wasting its time by checking the status of I/O device. Whenever the device is ready to accept or supply data, it informs the processor through a control signal known as

interrupt signal. In response to this interrupt signal, the microprocessor sends back an interrupt acknowledge signal to the I/O device indicating that it received the request (Fig. 7.8). It then suspends its job after executing the current instruction. It saves the contents

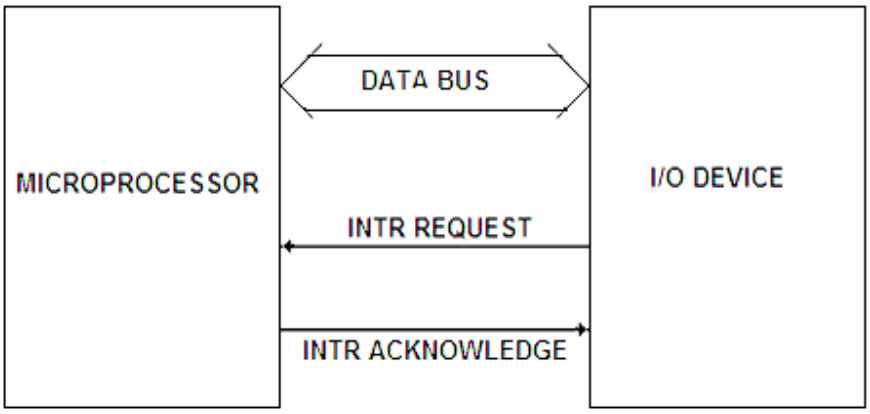


Fig. 7.8

of program counter and status to stack and jumps to the subroutine program. This subroutine program is called Interrupt Service Subroutine (ISS) program. The ISS saves the processor status into stack; and after executing the instruction for the data transfer, it restores the processor status and then returns to main program. The flow chart for this method of data transfer is shown in figure 7.9.

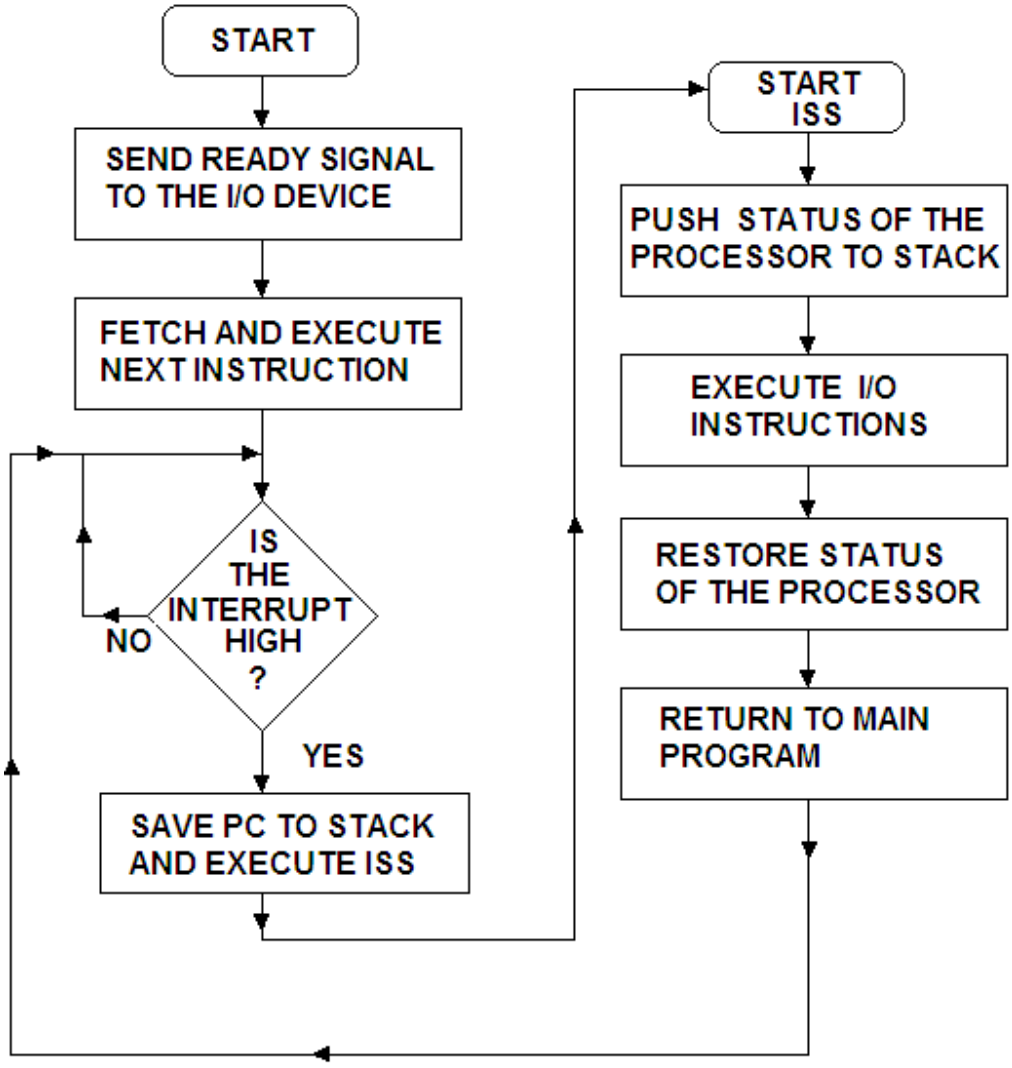


Fig. 7.9

As already discussed, several input/output devices may be connected to microprocessor using Interrupt Driven Data Transfer Scheme. Following interrupt request configuration may arise while interfacing the I/O devices to microprocessor.

1. Single Interrupt system
2. Multi Interrupt System

1. Single Interrupt System

When only one interrupt line is available with the microprocessor and several I/O devices are to be connected, then the method is known as Single Interrupt System. Figure 7.10(a) shows the way to connect several devices to one active low input interrupt terminal ($\overline{INTR}$) of the microprocessor. However, to connect the several I/O devices to active high interrupt terminal (*INTR*) is shown in figure 7.10(b). In the active low interrupt line of the microprocessor the devices are connected to $\overline{INTR}$ terminal through different open collector NOT gates; when any of the devices is active it provides a low

signal to $\overline{INTR}$ enabling the interrupt line. Similarly, in the active high interrupt line the I/O devices are connected through an OR gate. When any of the device is high the output of OR gate sends a high signal to interrupt line (INTR).

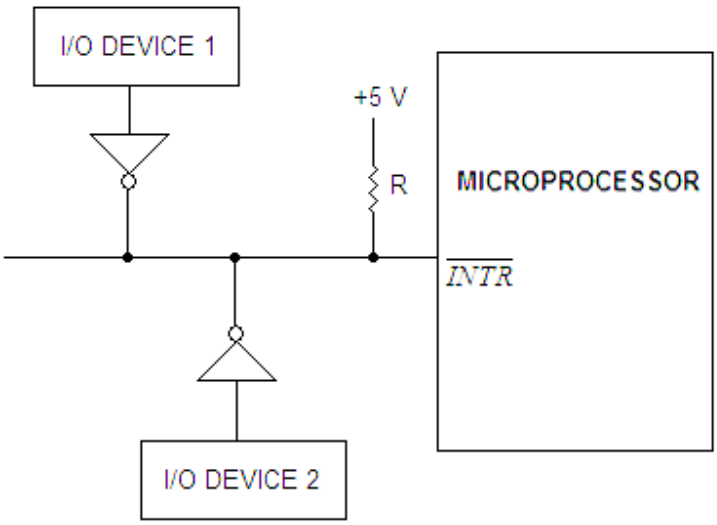


Fig. 7.10(a)

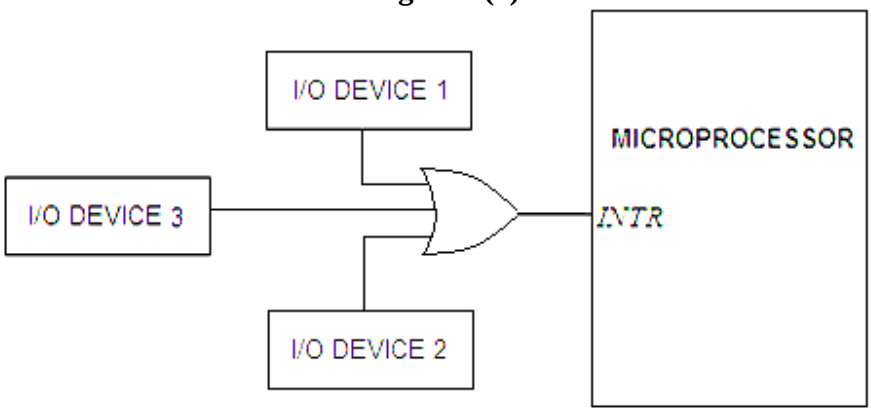


Fig. 7.10(b)

When the interrupt line is active in either of the two methods discussed above, then the microprocessor will not know which device has sent the interrupt signal. Three techniques are commonly used to solve the problem for the microprocessor that is requesting the interrupt and resolving any simultaneous requests by two or more devices. These are:

1. **Polling**: The interrupt signal from each device can be used to set one bit of a register wired as an input port. When an interrupt occurs, the ISS polls this port to see who requested service. A priority is automatically established by the order of polling. This technique is very simple but has the negative of degrading response time.
2. **Daisy Chain**: This technique has been shown in figure 7.11. In this method each I/O device has an Interrupt Enable Input (IEI) and an Interrupt Enable Output (IEO). An interrupt request can be made only if IEI is high. A serial

connection like a chain of all the I/O devices is made. The highest priority I/O device is placed at the first position followed by the lower priority devices in sequence. If any device sends an interrupt signal i.e. low to the interrupt line, then the INTR line of the processor is enabled. The interrupt acknowledge signal (INTA) is enabled in response to low INTR line. In figure 7.11, device 2 is requesting an interrupt causing its IEO to be low. This in turn disables devices 3 and 4. It may be noted that device 1 is still able to request an interrupt because it has a higher priority. The interrupt acknowledge signal can be used to reset IEO of the interrupting device.

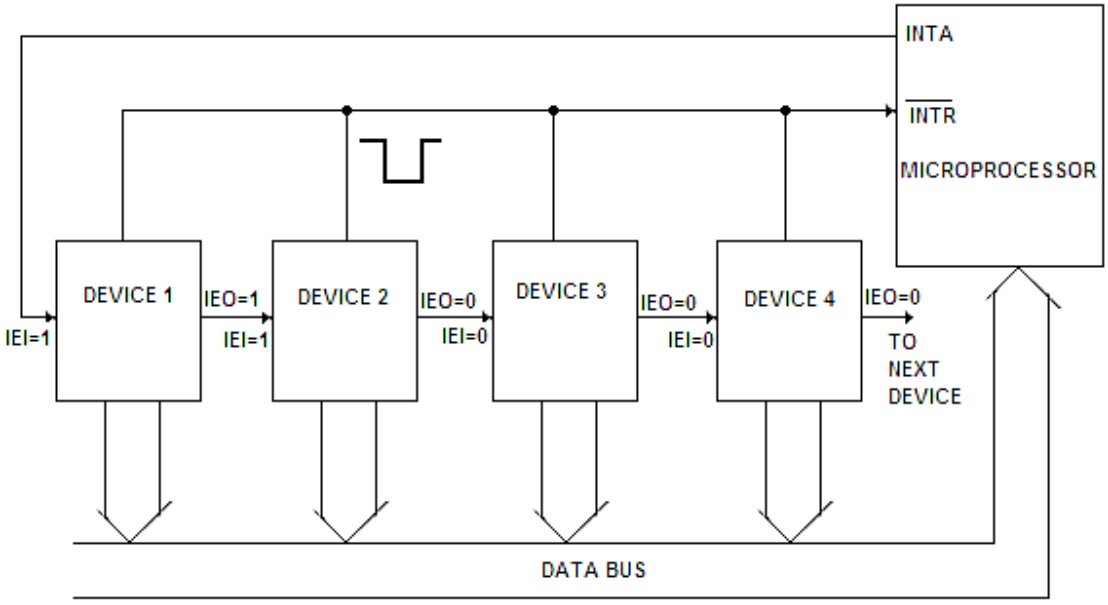


Fig. 7.11

3. **Priority Interrupt Controller (PIC)**: In this method several I/O devices may be connected to a single interrupt line through programmable interrupt controller (IC 8259). Up to 8 input/output devices may be connected to the microprocessor. If more than 8 I/O devices to be connected, more PICs (programmable interrupt controllers) are used in cascade. The details of PIC will be discussed in a later chapter.

2. Multi Interrupt System

When the microprocessor has several interrupt terminals and one I/O device is to be connected to each interrupt terminal, then it is known as multi interrupt system. In this scheme, the number of I/O devices to be connected to the interrupt lines should be equal to or less than the number of interrupt terminals. In this way one device is connected to each level of interrupt. So when a device interrupts the microprocessor, it immediately knows which device has interrupted. Such an interrupt scheme is known as vectored interrupt.

7.2.3 Direct Memory Access (DMA) Data Transfer

In programmed I/O or interrupt driven I/O methods of data transfer between the I/O devices and external memory is via the accumulator. For bulk data transfer from I/O

devices to memory or vice-versa, these two methods discussed above are time consuming and quite uneconomical even though the speed of I/O devices matches with the speed of microprocessor; since the data is first transferred to accumulator and then to concerned device. The Direct Memory Access (DMA) data transfer method is used for bulk data transfer from I/O devices to microprocessor or vice-versa. In this method I/O devices are

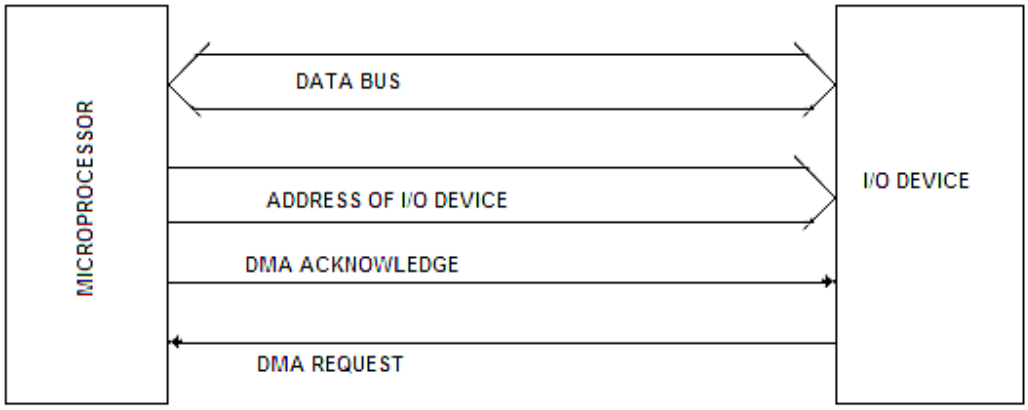


Fig. 7.12

allowed to transfer the data directly to the external memory without being routed through accumulator. For this the microprocessor relinquishes the control over the data bus and address bus, so that these can be used for transfer of data between the devices. For the data transfer using DMA process, a request to the microprocessor by the I/O device is sent and on receipt of such request, the microprocessor relinquishes the address and data buses and informs the I/O devices of the situation by sending Acknowledge signal as shown in figures 7.12 and 7.13. The I/O device withdraws the request when the data transfer between the I/O device and external memory is complete.

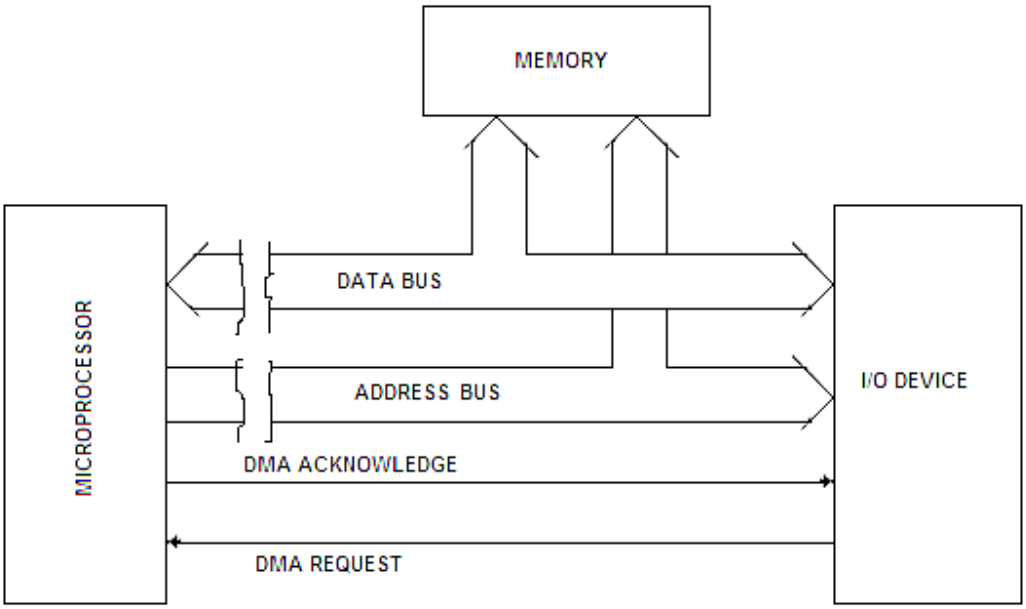
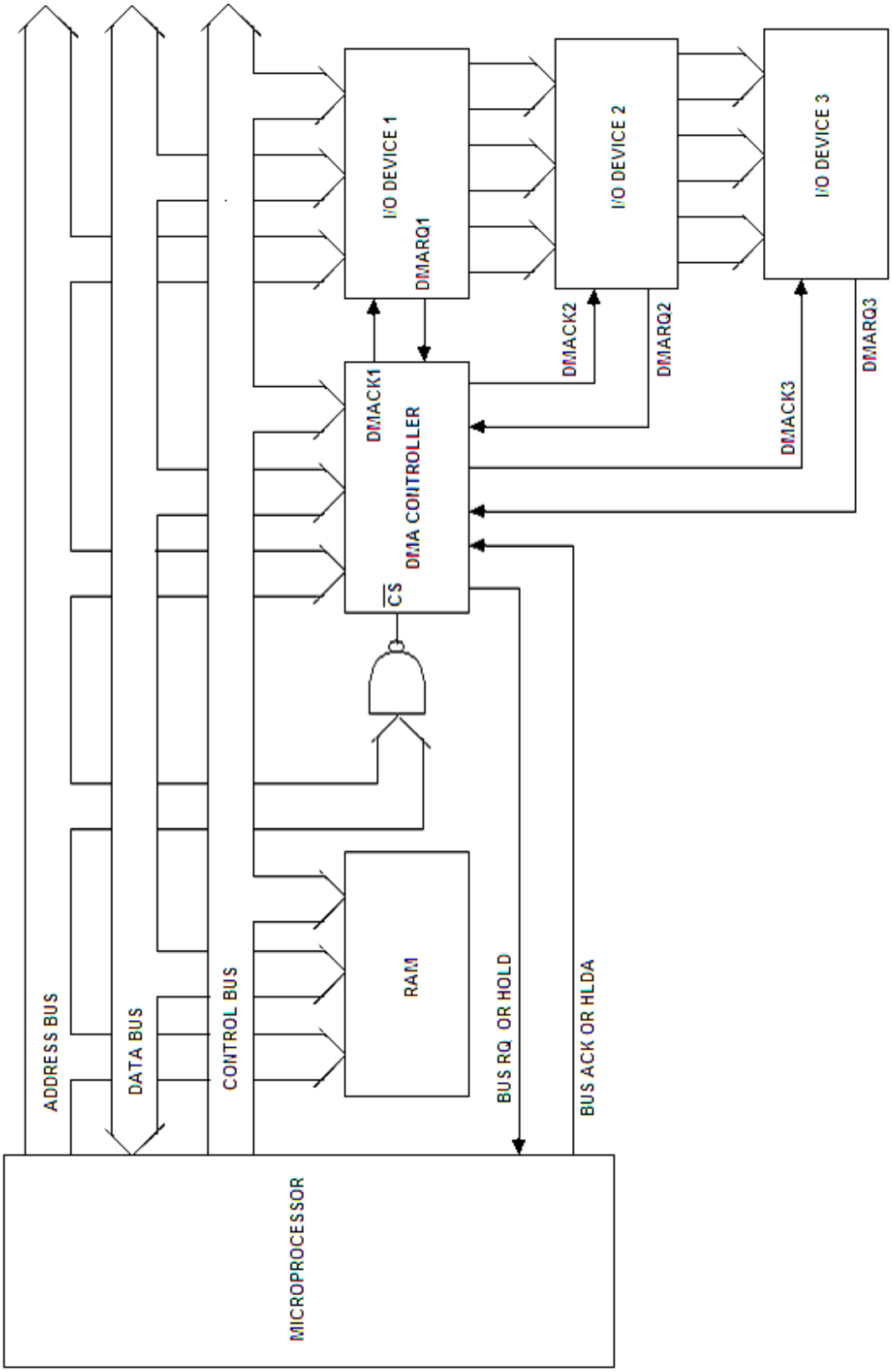


Fig. 7.13

It may be mentioned here that DMA transfer the data of the following types:

- Memory to I/O device
- I/O device to memory
- Memory to memory
- I/O device to I/O device

For transferring the data through DMA, an interfacing chip known as DMA Controller is used with the microprocessor that helps to generate the addresses for the data to be transferred from the I/O devices (Fig. 7.14). The peripheral device sends the request signal (DMARQ) to the DMA controller and the DMA controller in turn passes it to the microprocessor (HOLD signal). On receipt of the DMA request the microprocessor sends an acknowledge signal (HLDA) to the DMA controller. On receipt of this signal (HLDA) the DMA controller sends a DMA acknowledge signal (DMACK) to the I/O device. The DMA controller then takes over the control of the buses of microprocessor and controls the data transfer between RAM and I/O device. When the data transfer is complete, DMA controller returns the control over the buses to the microprocessor by disabling the HOLD and DMACK signals.


I/O DEVICE 1
DMARQ1
I/O DEVICE 2
I/O DEVICE 3
DMACK2
DMARQ2
DMACK3
DMARQ3
DMACK1
DMA CONTROLLER
CS
ADDRESS BUS
DATA BUS
CONTROL BUS
RAM
BUS RQ OR HOLD
BUS ACK OR HLDA
MICROPROCESSOR

7.3 THE 8085 INTERRUPTS

As already discussed, in Interrupt Driven I/O data transfer methods the microprocessor gets interrupted by the I/O device when it is ready to transfer the data. The microprocessor suspends its job after executing the current instruction. It saves the contents of program counter to stack and jumps to the subroutine program. This subroutine program is called Interrupt Service Subroutine (ISS) program. The ISS saves the processor status into stack; and after executing the instruction for the data transfer, it restores the processor status and then returns to main program. The ISS executes the relevant set of instructions stored at a predetermined memory block. The Intel 8085 microprocessor has five hardware interrupt inputs on its chip. Besides these, the microprocessor has eight interrupt instruction in the instruction set. The microprocessor responds to both the hardware and software interrupts. In the following sections the details of both software and hardware interrupts will be discussed.

7.4 SOFTWARE INTERRUPTS

The 8085 microprocessor has eight software interrupts namely, RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6 and RST 7.

The syntax for these interrupt instructions is given by:

RST n

where *n* varies from 0 to 7.

These instructions are single byte instructions. When an interrupt occurs, a CALL instruction to a predetermined location of the memory is executed. The effect of each restart instruction is shown in table 7. 4.

Table 7.4

Instruction	Effect	Op Code	Binary equivalent	Vector Location
RST 0	CALL 0000 H	C7	1100 0111	0000 H
RST 1	CALL 0008 H	CF	1100 1111	0008 H
RST 2	CALL 0010 H	D7	1101 0111	0010 H
RST 3	CALL 0018 H	DF	1101 1111	0018 H
RST 4	CALL 0020 H	E7	1110 0111	0020 H
RST 5	CALL 0028 H	EF	1110 1111	0028 H
RST 6	CALL 0030 H	F7	1111 0111	0030 H
RST 7	CALL 0038 H	FF	1111 1111	0038 H

These RST instructions are like vectors because they point to specific locations in the memory. The starting address of each instruction is called a **Vector Location**. The vector location for RST 0 is 0000 H and for RST 1 is 0008 H and so on. The vector locations of each instruction are 8 bytes apart. Therefore 8 bytes of instructions can be stored beginning at any vector location.

Figure 7.15 shows the hardware for the implementation of the software interrupt instruction. This circuit is for the implementation of an instruction RST0. In response to the interrupt request signal, the 8085 microprocessor sends the $\overline{INTA}$ (interrupt acknowledge) signal, which is used to enable the buffer. The hex code C7 H of RST 0 will be placed on the data bus.

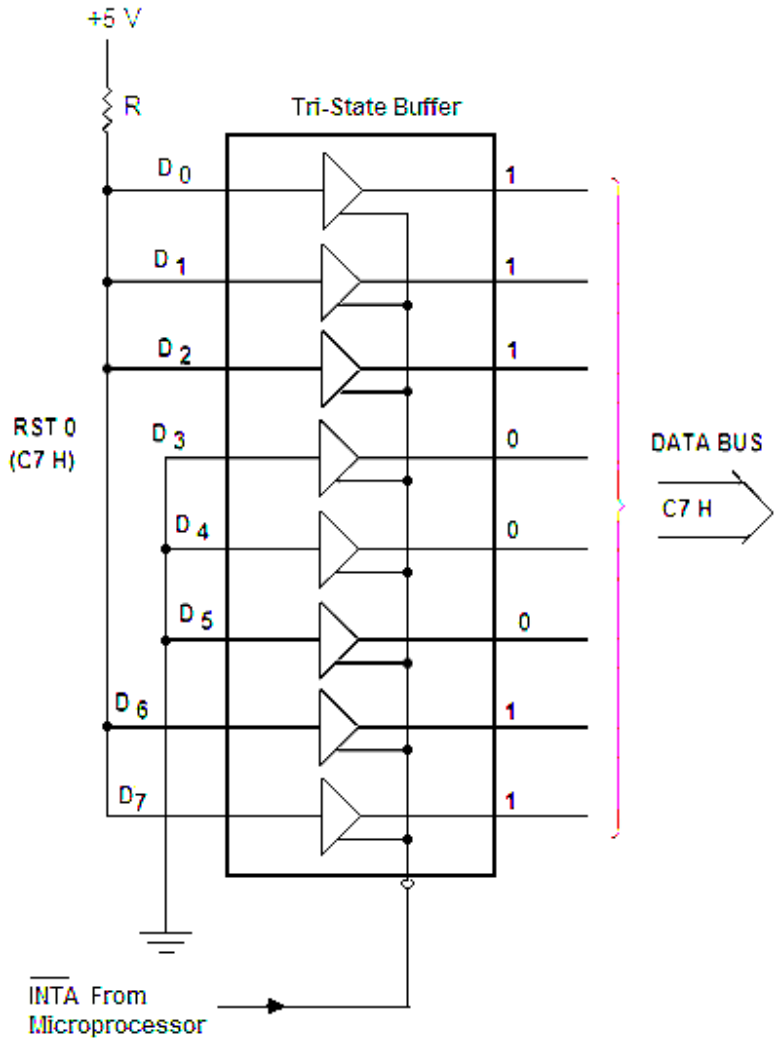


Fig. 7.15

Figure 7.16 illustrates how software interrupt instructions are executed. Suppose in the main program RST2 instruction is given, and when this instruction is executed the contents of the program counter is pushed onto the stack. Then the program counter jumps to the address 0010 H. The subroutine located from 0010 H to 0017 H is executed, with the RET instruction as the last instruction of the subroutine program. So as the RET instruction is executed it returns to the main program.

Similarly, if another restart instruction RST4 is encountered in the main program, then the contents of the program counter are again pushed onto the stack. The program jumps to the subroutine program whose starting address is given by 0020 H. After the execution of this subroutine program it returns to the main program as shown in figure 7.16.

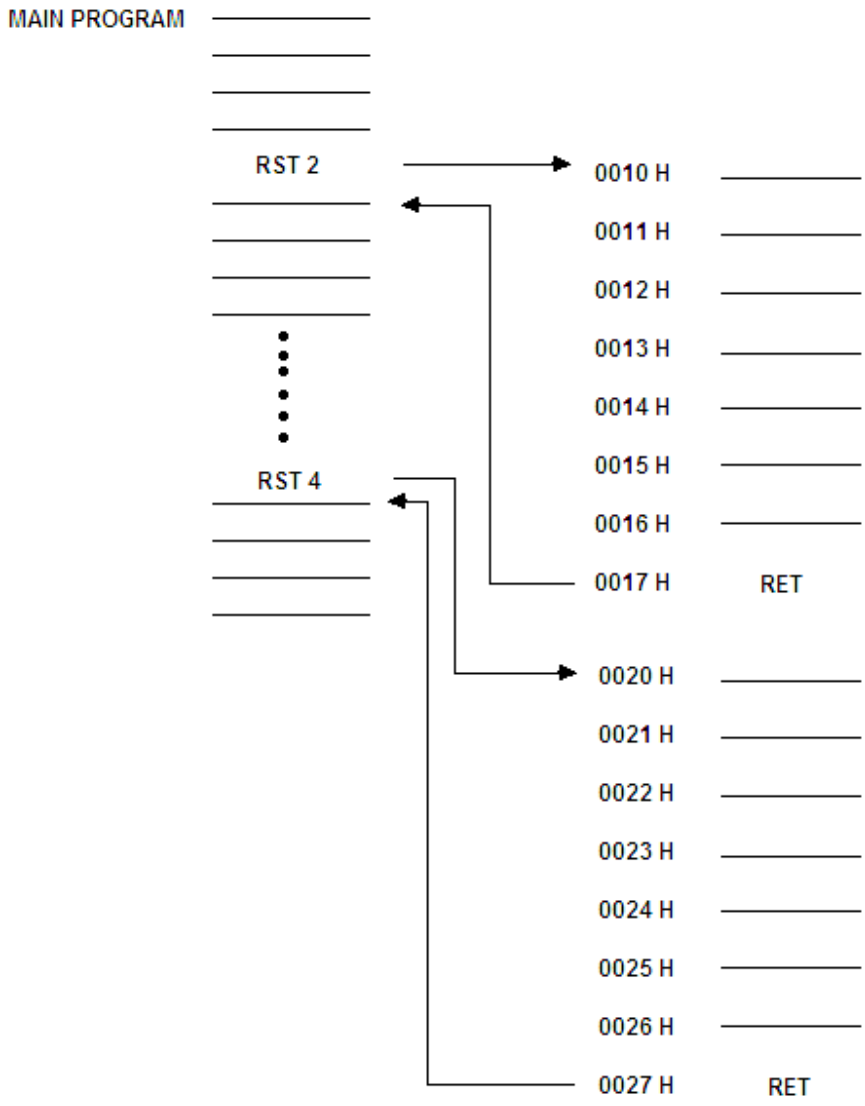


Fig. 7.16

From the above discussion it is clear that the software interrupt instructions are basically special kind of CALL instructions; when any of these instructions is executed it branches to the predetermined address. Notice that all these instructions are of one byte (ref. table 7.4) while the standard CALL instruction is of three bytes. The vector addresses of these 8 software interrupts are 8 bytes apart. So 8 bytes of instructions can be stored in the subroutine program associated with each of these instructions. But generally, the subroutine programs may require more than 8 bytes. For this reason the complete subroutine program are usually not stored in the vector locations, rather the vector locations are used to specify the starting address of the longer subroutine program i.e. in the vector location of these instructions JMP XXXX H may be stored. The XXXX H represents the starting address of the longer subroutine program. The last instruction of the program will be RET, so that after completing the subroutine program it jumps to the main program as shown in figure 7.17.

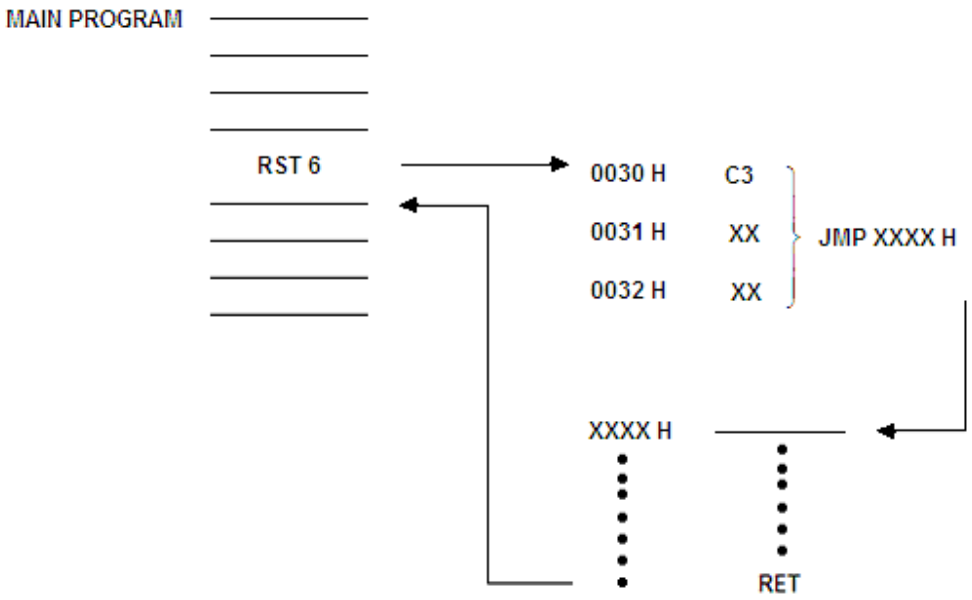


Fig. 7.17

The hardware solution to multi-interrupt problem is carried out by the circuit shown in figure 7.18. The multi-interrupt problem means the possibility of two requests arriving simultaneously. This circuit will accept eight interrupt request and work as per their priority and separate RST instruction for each request is generated. For the generation of all eight RST instructions, a 3-to-8 priority encoder (IC 74LS148) along with a tri-state octal buffer (IC 74LS244) is used as shown in figure 7.18.

The 3-to-8 priority encoder generates a 3 bit binary code corresponding to active input. If two or more inputs are active simultaneously, the highest numbered input will be encoded. The three bits are inverted by the inverted buffer and then applied to an octal buffer (IC 74LS244). The op code corresponding to the input will be generated which is latched on to the data bus if $\overline{INTA}$ signal is active.

The op codes for each RST instruction given in table 7.4 are reproduced below:

Instruction	Op Code	Binary equivalent
RST 0	C7	1100 0111
RST 1	CF	1100 1111
RST 2	D7	1101 0111
RST 3	DF	1101 1111
RST 4	E7	1110 0111
RST 5	EF	1110 1111
RST 6	F7	1111 0111
RST 7	FF	1111 1111

It may be noted from this op code table that D_0 to D_2 and D_6 to D_7 bits of the op codes are same for all the RST instructions (RST 0 to RST 7). The bits D_3 to D_5 are different and are in the sequence of 3 bit binary numbers for the RST 0 to RST 7. In general the code for RST instructions may be written as:

11CCC111

where CCC is

000 for RST 0
001 for RST 1
010 for RST 2
011 for RST 3
100 for RST 4
101 for RST 5
110 for RST 6
111 for RST 7

This sequence permits to use the 3-to-8 priority encoder. It can be understood that a signal say $\overline{INT3}$ is low then A_0 A_1 A_2 will be100 which will be inverted by the inverter buffer as 011 (I_3 I_4 I_5). Since the other lines I_0, I_1, I_2, I_6, I_7 are all high, the octal buffer will provide the op code DF H (op code for RST 3) on the 8085 data bus as soon as the interrupt acknowledge signal ($\overline{INTA}$) is made low in response to the interrupt request signal. Similarly by activating other inputs of the priority encoder, the vector CALL instructions can be generated on the 8085 data bus.

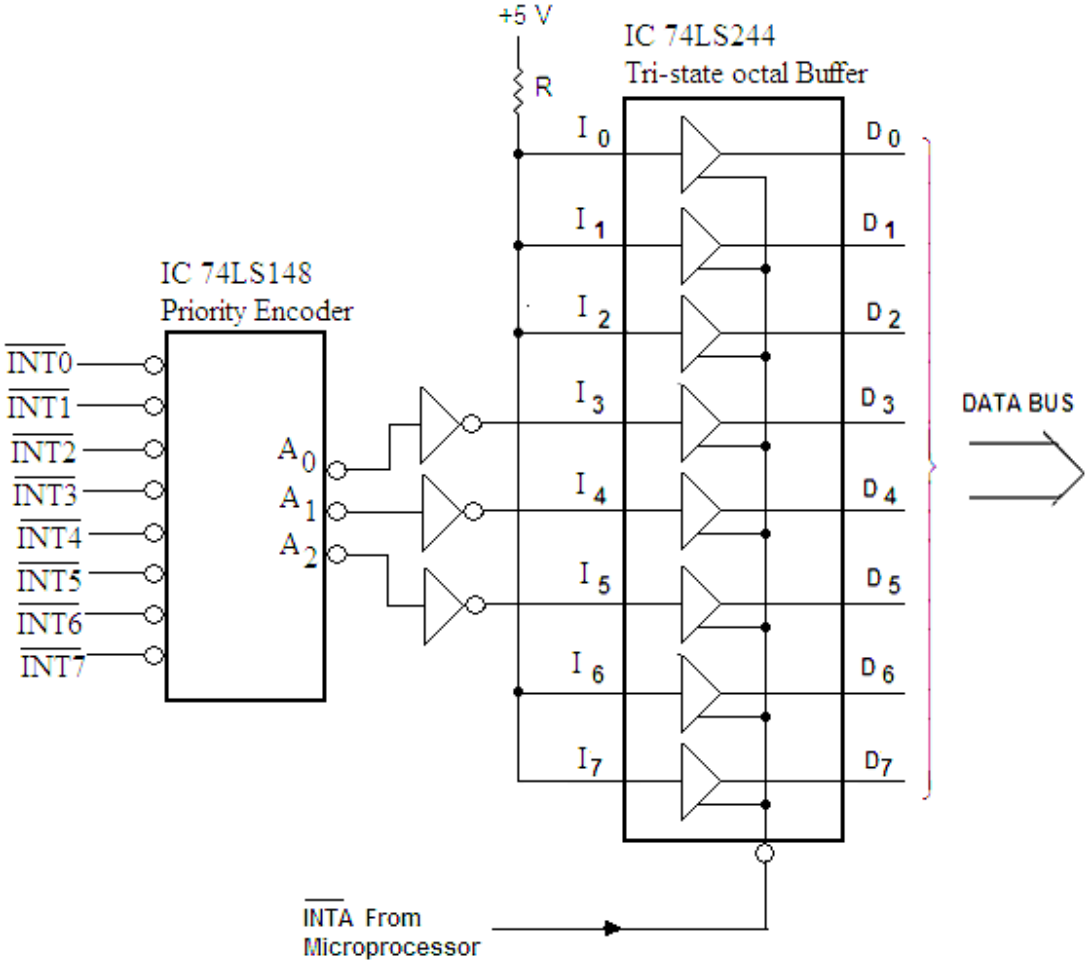


Fig. 7.18

7.5 HARDWARE INTERRUPTS

In addition of software interrupts discussed above the 8085 has five hardware interrupts also. For this, five hardware input pins are provided in the microprocessor. The hardware interrupts are initiated by an external device, by placing an appropriate signal at the interrupt pin of the microprocessor. The five interrupts RST 5.5, RST 6.5, RST 7.5, TRAP and INTR are shown in table 7.5. Out of these interrupts RST 5.5, RST 6.5, RST 7.5 and TRAP are vector interrupts and INTR is the non-vectored interrupt. In vector interrupts the microprocessor automatically sends the specific address to program counter in response to an interrupt request signal. However, in non-vectored interrupt, the interrupt device has to give the address of the interrupt service subroutine.

Table 7.5

Interrupt	Priority	Vector Location
TRAP	1	0024 H
RST 7.5	2	003C H
RST 6.5	3	0034 H
RST 5.5	4	002C H
INTR	5	-----

It may be noted from the table that TRAP has the highest priority, RST 7.5 next highest and so on. If two or more hardware interrupts are served at the same time then the microprocessor executes them in order of their priority level; i.e. TRAP is served first, then RST 7.5 and so on. When highest priority interrupt is executed, the lower priority interrupts remain pending rather they are known as pending interrupts.

The RST 7.5, RST 6.5 and RST 5.5 are the maskable interrupts and TRAP is non-maskable interrupt. The maskable means the prevention of any interrupt. Some times it may be required to prevent one or more interrupts when a certain task is being carried out by the microprocessor; for this the masking of these interrupts is done. The masking of any interrupt is carried out by an instruction known as SIM (Set Interrupt Mask). The SIM is one byte instruction. To find the status of the interrupts i.e. to know which interrupt is masked or which interrupt is pending, another instruction known as RIM (Read Interrupt Mask) is used. These two instructions RIM and SIM will be discussed in section 7.7.

7.6 INTERRUPT CONTROL CIRCUIT

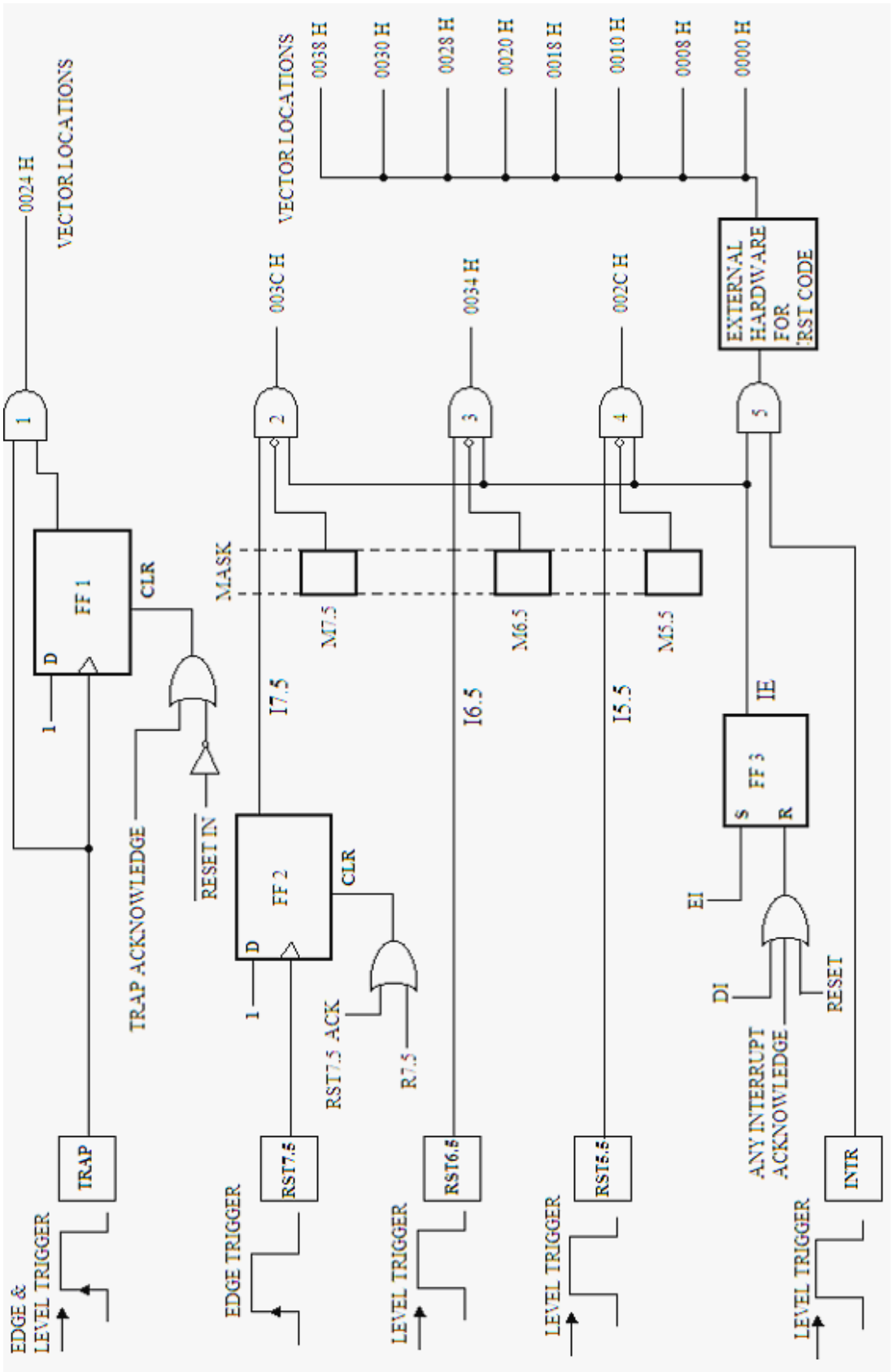
Figure 7.19 shows the interrupt control circuit with 8085 microprocessor. The TRAP is non-maskable interrupt i.e. it is neither be affected by any flag nor can be masked. It is used to handle very important functions. Since it is a highest priority interrupt, so it is used to take care of parity errors, power failure and other events that needs immediate attention. In the case of power failure, it may execute a routine to transfer the contents of the main memory to the back up memory (if any); and also for parity errors, the data may be corrected before carrying on. It is edge and level trigger which means the input has to go high and stays high. The rising edge and level triggered TRAP signal triggers the D flip-flop. The logic '1' at the output of D flip-flop and logic '1' of the TRAP input enable the AND gate to trigger the TRAP interrupt i.e. it calls the vector location 0024 H as shown in figure 7.19. Once the TRAP is recognized, further inputs at the TRAP will not be considered unless the interrupt is reset. The TRAP interrupt will be reset if the $\overline{RESET}$ signal is active (low) or internal TRAP

Acknowledge signal is high. After the 8085 microprocessor recognizes a TRAP interrupt, it will send a high TRAP acknowledge signal which will reset the D flip-flop.

RST 7.5 is a maskable interrupt which can be enabled or disabled by using SIM instruction. This is an edge triggered interrupt. When a leading edge signal appears at the RST 7.5 pin of 8085 microprocessor, D flip-flop 2 will be set (ref. figure 7.19). The output of this flip-flop is labeled as I 7.5 which is known as pending interrupt. This is one input of AND gate 2. The AND gate 2 will not be enabled until other two inputs of the gates are high. The RST flip-flop will be reset either by having a high R 7.5 bit or by having a high RST Acknowledge signal. The interrupt Acknowledge signal resets it for future RST 7.5 interrupt.

RST 6.5 and RST 5.5 are also maskable interrupts which may be enabled or disabled using SIM instruction. Both are level triggered interrupts. When a high signal (constant voltage of +5 V) appears at RST 6.5 pin of the microprocessor, it will enable one pin of AND gate 3. Till RST 6.5 interrupt pin is high and other two pins of AND gate 3 are low, this interrupt is known as pending interrupt (I 6.5). Similarly, when a high signal appears on RST 5.5 pin of the microprocessor, it will enable one terminal of AND gate 4. This interrupt will be pending interrupt (I 5.5) till RST 5.5 pin is high and other two inputs of AND gate 4 are low.

It may be noted from the figure 7.19 that one input each of the AND gates 2, 3 and 4 is connected to IE signal (called interrupt enable flag). The second pin of these gates will be enabled if IE signal is high. The IE signal may be made to high by enabling


VECTOR LOCATIONS
0024 H
VECTOR LOCATIONS
0038 H
0030 H
0028 H
0020 H
0018 H
0010 H
0008 H
0000 H
003C H
0034 H
002C H
EXTERNAL HARDWARE FOR RST CODE
1
2
3
4
5
FF 1
D
CLR
MASK
M7.5
M6.5
M5.5
I7.5
I6.5
I5.5
IE
FF 3
S
R
EI
DI
RESET
TRAP ACKNOWLEDGE
RESET IN
FF 2
RST7.5 ACK
R7.5
ANY INTERRUPT ACKNOWLEDGE
TRAP
RST7.5
RST6.5
RST5.5
INTR
EDGE & LEVEL TRIGGER
EDGE TRIGGER
LEVEL TRIGGER
LEVEL TRIGGER
LEVEL TRIGGER

the EI (Enable Interrupt) signal, which may be enabled by a software instruction **EI**. This instruction will be discussed in the succeeding section.

The mask bits M 7.5, M 6.5 and M 5.5 for the interrupts may be set using the SIM instruction. To mask an interrupt, the corresponding mask bit has to be set. So to enable a pending interrupt, the corresponding mask bit should be made low (by SIM instruction) and interrupt enable flag should made high (by EI instruction).

All the maskable interrupts may be disabled by either sending a low signal to $\overline{RESETIN}$ terminal or a high signal to 'Any Interrupt Acknowledge' terminal. The DI signal may also disable all the maskable interrupts. The DI (Disable Interrupts) is a software instruction.

All these hardware interrupts discussed above, once enabled will execute the corresponding hardware CALL instruction specified by their vector locations, as per their priority levels.

INTR is a lowest priority, maskable and level triggered interrupt. It uses handshaking. A high signal to INTR pin of the microprocessor will cause the current instruction to complete and will put the contents of the program counter into stack. The microprocessor will then generate a low $\overline{INTA}$ (interrupt acknowledge) signal. This signal is then used to enable a tristate buffer for the execution of hardware CALL instruction. It will execute the service subroutine program corresponding to any of the 8 software interrupts (RST 0 through RST 7).

7.7 INTERRUPT INSTRUCTIONS

Once the microprocessor recognizes any of the interrupts, it immediately disables all the interrupts except TRAP. This is done just to ensure that no further interrupts are recognized while the interrupt service subroutine (ISS) is being executed. Once the ISS is complete the program is required to enable the interrupts again. For this one byte instruction EI is introduced just before the RET instruction of ISS.

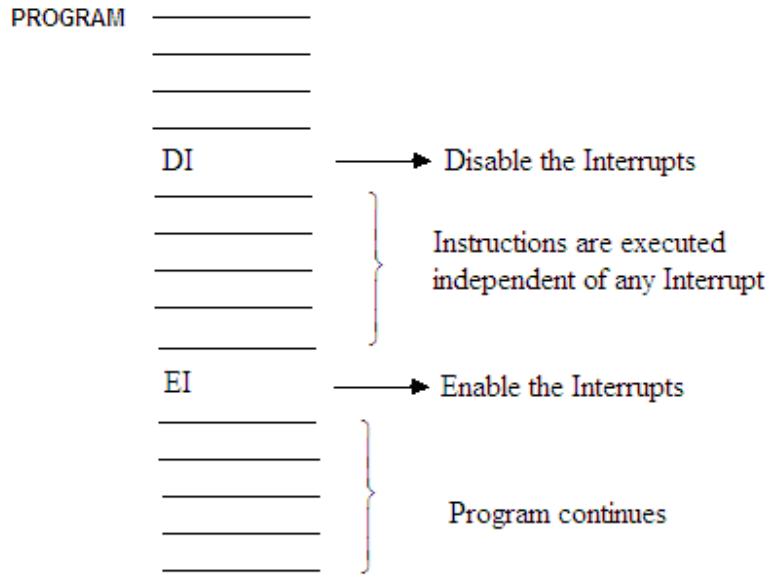


Fig. 7.20

Some portion of the main program may, sometimes be executed without being enabled the interrupt signals. For this the interrupts are to be disabled. The interrupts may be disabled by one byte instruction DI (Disable Interrupts). Later the interrupts may be enabled as shown in figure 7.20.

EI and DI Instructions

The instruction

EI

stands for Enable Interrupt. When this instruction is executed, it produces a high EI signal (fig. 7.19), which sets the flip-flop 3 and produces a high signal to interrupt enable flag (IE). This way EI instruction enables all the interrupts except TRAP.

The instruction

DI

stands for Disable Interrupts. When this instruction is executed, it produces a high DI bit of flip-flop 3 (fig. 7.19). This resets the flip-flop and results a low IE (interrupt enable flag) signal. The low IE signal disables all the interrupts except TRAP.

SIM and RIM Instructions

It is often required to selectively enable a few interrupts and disable others. The selectively enabling or disabling the interrupts can be done by an instruction

SIM

It stands for set interrupt masks. This instruction is used by loading the accumulator as shown in figure 7.21. The accumulator is loaded by *MVI A, data* (data bits are as per the requirements) instruction. The SIM instruction is then executed.

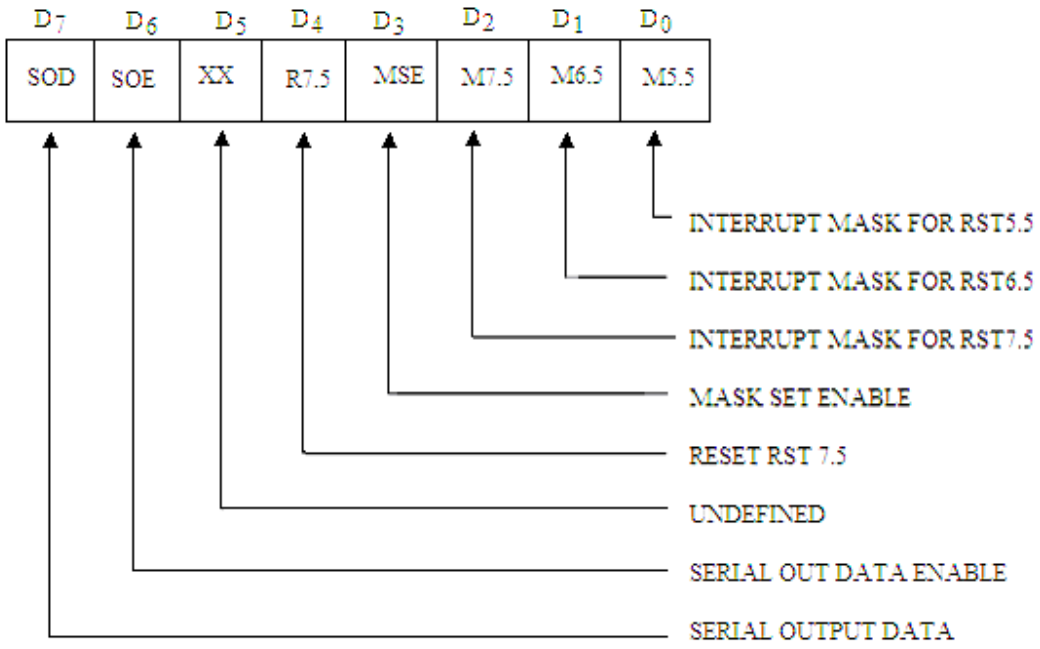


Fig. 7.21

The meaning of the different mask bits are given below:

The bits D_0 to D_2 are the mask bits (M 5.5 to M 7.5). A high to either of these bits represents that the particular interrupt is masked and a low, however, to either of these bits represents the enabling of that particular interrupt.

The bit D_3 is known as MSE (Mask Set Enable). When this bit is low, the mask bits D_0 to D_2 are ignored. A high to this bit indicates that the bits D_0 to D_2 are valid as described above.

The bit D_4 when high resets the flip-flop 2 (figure 7.19), in order to override RST 7.5 without servicing it.

D_5 is undefined bit.

The bits D_6 to D_7 are used for the serial transfer of data through the SOD line. The working of these pins will be described later.

Let us take an example to illustrate the function of SIM instruction. For example we have

```
MVI A, 0C H
SIM
```

instructions in an assembly language program for 8085 microprocessor. When first instruction MVI A, 0C H is executed it will have the data (as shown in figure 7.22) for

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	1	1	0	0

Fig.7.22

the execution of SIM. The SIM instruction after its execution will enable MSE signal (as D_3 bit is high). RST 7.5 interrupt is masked (bit D_2 is 1) and RST 5.5 and RST 6.5 interrupts are unmasked (enabled) as bits D_0 and D_1 are both 0. It will prevent the RST 7.5 interrupt from arriving at the final output.

The another interrupt instruction is

RIM

It stands for Read Interrupt Mask. This instruction will give the present status of the interrupt. This instruction loads the accumulator with 8 bit data whose details are given in figure 7.23.

The meanings of the different bits for RIM are given below:

The bits D_0 to D_2 represent the masking of RST 5.5, RST 6.5 and RST 7.5 interrupts. A high to either of these bits represents that the particular interrupt is enabled; and a low to either of these bits represents that the particular interrupt is disabled.

The bit D_3 is known as interrupt enable flag (IE). When this bit is low, all the interrupts except TRAP are disabled and a high to this bit mask the bits D_0 to D_2 are ignored. A high to this bit indicates that the bits D_0 to D_2 are valid as described above.

The bits D_4 to D_6 represent the pending interrupts. A high to either of these bits represents that particular interrupt is pending; and a low to either of these bits represents that particular interrupt is not pending.

The bit D_7 is a serial input data and used for the serial input data through SID line. The working of this bit will be discussed later.

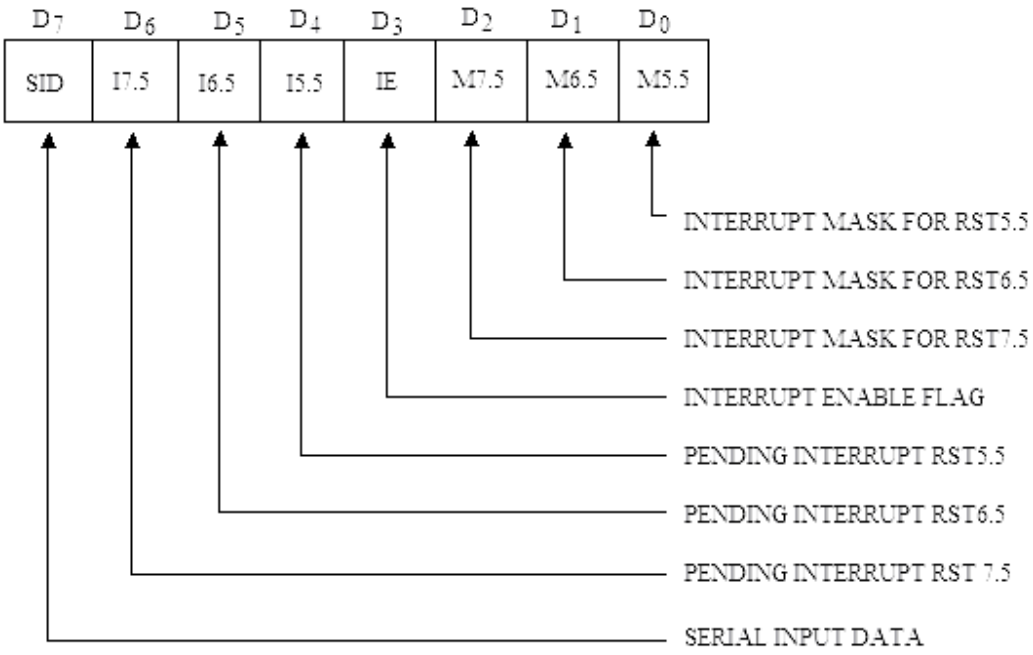


Fig. 7.23

Let us take an example to illustrate the function of RIM instruction. Suppose after the execution of RIM instruction, the accumulator contains 2A H as the data as shown in figure 7.24. The high to I 6.5, IE and M 6.5 bits indicate that RST 6.5 is a pending interrupt, the interrupt system is enabled and the RST 6.5 is currently masked.

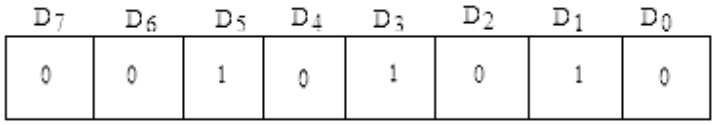


Fig. 7.24

Example 7.1. *Write an assembly language program for 8085 microprocessor to enable RST 5.5 interrupt and disable RST 6.5 and RST 7.5 interrupts.*

Solution. To enable RST 5.5, the bit D_0 (M 5.5) for the accumulator should be 0 (unmask); and for disabling RST 6.5 and RST 7.5, the bits D_1 and D_2 (M 6.5 and M 7.5) should be masked (1). Also MSE should be 1 (enable). The program will, therefore, be as given below:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SOD	SOE	XX	R 7.5	MSE	M 7.5	M 6.5	M 5.5	
0	0	0	0	1	1	1	0	
								=0E H

```
EI
MVI A, 0E H
SIM
```

Example 7.2. *Write an assembly language program for 8085 microprocessor to enable all the interrupts.*

Solution. The program for this case will be as given below. The bits D_0 to D_2 should be unmask and MSE should be enabled.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SOD	SOE	XX	R 7.5	MSE	M 7.5	M 6.5	M 5.5	
0	0	0	0	1	0	0	0	
								=08 H

```
EI
MVI A, 08 H
SIM
```

Example 7.3. *Write an assembly language program for 8085 microprocessor to enable RST 5.5 and RST 6.5 interrupt and reset 7.5 interrupt.*

Solution. The program for this case will be as given below. The bits D_0 and D_1should be unmasked and MSE should be enabled. For resetting of RST 7.5, the bit D_4 (R 7.5) should also be 1.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SOD	SOE	XX	R 7.5	MSE	M 7.5	M 6.5	M 5.5	
0	0	0	1	1	1	0	0	
								=1C H

```
EI
MVI A, 1C H
SIM
```

Example 7.4. *Write an assembly language program for 8085 microprocessor to check if RST 5.5 is pending. If it is pending, enable it without affecting any other interrupt else return to main program..*

Solution. The program for this problem is given below. The RIM instruction will check if RST 5.5 is a pending interrupt. For this bit pattern of the accumulator will be checked. If bit D_4 is 1, the RST 5.5 is pending interrupt otherwise not.

For RIM

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SID	I 7.5	I 6.5	I 5.5	IE	M 7.5	M 6.5	M 5.5

For SIM

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SOD	SOE	XX	R 7.5	MSE	M 7.5	M 6.5	M 5.5

```
      Label   Mnemonics   Operand     Comments
              RIM                     ; Read Interrupt mask
              MOV  B,     A           ; Mask information is moved to B-
                                        reg.
              ANI         10 H        ; Check if RST 5.5 is pending.
              JNZ         NXT         ; Jump to NXT if it is pending
                                        interrupt.
              EI                      ; Enable interrupts.
              RET                     ; RST 5.5 is not pending return to
                                        main program.
NXT           MOV  A,     B           ; Get bit pattern (RST 5.5 is
                                        pending).
              ANI         0E H        ; Enable RST 5.5 by not masking D0
                                        bit (For SIM).
              ORI         08 H        ; Enable MSE for SIM.
```

```
SIM
JMP        ISS        ; Jump to service subroutine for RS
                        T5.5.
```

7.8 SERIAL INPUT AND OUTPUT DATA TRANSFER

As already discussed in the architecture of 8085, two pins (Pin Nos. 4 and 5) are provided for SOD (Serial Out Data) and SID (Serial In Data) lines. These lines are used for serial data transfer. The data transfer to or from the SID or SOD lines is possible using the Instructions RIM (Read Interrupt Mask) and SIM (Set Interrupt Mask).

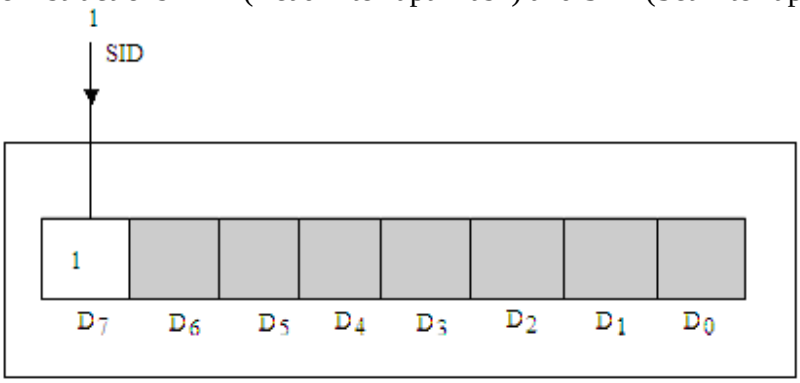


Fig. 7.25

The data on the SID line (Pin 5 of 8085) is loaded into accumulator at bit D_7 whenever a RIM instruction is executed. In other words a RIM instruction may be executed each time a new bit arrives at the SID input. For example, let a bit '1' arrives at the SID input. RIM instruction is now executed. After the execution of RIM instruction D_7 bit of the accumulator will be 1 as shown in figure 7.25.

Further to input 8 bit data serially through SID line, RIM instruction is executed 8 times and each time D_7 bit may be isolated and saved for the conversion of serial data into parallel data.

The SIM instruction sends the D_7 bit of the accumulator to the SOD line of 8085. For this transfer, D_6 bit (SOE) of the accumulator must be high as shown in figure 7.26.

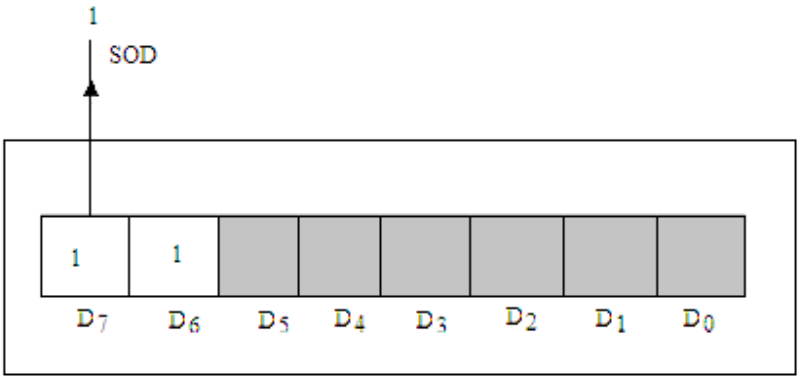


Fig. 7.26

Suppose we wish to send a '0' bit to the SOD line, this can be done as:

```
MVI A, 40 H
SIM
```

Similarly, to send a '1' bit to the SOD line, we use

```
        MVI A, C0 H
        SIM
```

The rotate or other instructions may used to convert 8 bit parallel data to serial data stream at the SOD output.

It may be noted from the above discussion that bit D_7 is used for SID line and this bit has nothing to do with the interrupt system. Similarly, bits D_6 and D_7 are used for SOD line and these bits have nothing to do with the interrupt system. So no new instructions are to be used for the serial transfer of data, i.e. interrupt instructions RIM and SIM may be used for this purpose also.

Example 7.5. *Consider a switch is connected to SID line and an LED to SOD line of 8085. It is required to input the SID line via switch and output the switch status to the LED. In other words when the switch is ON or OFF the LED should glow or not glow. Write an assembly language program to accomplish this.*

Solution. Program to accomplish the required work of the problem is given as:

```
Label   Mnemonics   Operand    Comments
START   RIM                    ; Read Interrupt mask, Bit D7 of A is SID.
        ORI         40 H       ; Enable SOE for SIM.
        SIM                    ; Output to LED.
        JMP         START      ; Jump to START.
```

Example 7.6. *Write an assembly language program to generate a symmetrical square wave of known frequency at the SOD line of 8085 microprocessor.*

Solution. To generate a square wave of certain frequency, SOD line should remain high for certain time and then low for the same amount of time. This is done by using the program given below. The time for the delay should be as per the frequency of the wave to be generated.

MAIN PROGRAM:

```
Label   Mnemonics   Operand    Comments
START   MVI  A,      C0 H      ; Enable SOE and SOD (D6 and D7) and disables all interrupts for SIM.
        SIM                    ; Sends high signal to SOD line.
        CALL        DELAY      ; Delay is introduced for SOD to remains high for certain time.
        MVI  A,     40 H       ; SOD is made low; and SOE and all interrupts are disabled for SIM.
        SIM                    ; Sends low signal to SOD line.
        CALL        DELAY      ; Again delay is introduced for SOD to remain low for certain time.
        JUMP        START      ; Repeats the process.
```

SUBROUTINE PROGRAM:

```
Label   Mnemonics   Operand    Comments
DELAY   LXI  D,     XXX H      ; Loads DE register pair with a 16-bit number.
LOOP    DCX  D                 ; Decrements DE register pair by 1.
```

```
        MOV  A,     E       ; Moves the contents of E register to
                            accumulator.
        ORA  D              ; ORing of the contents of D and E
                            registers are performed to set the
                            zero flag.
        JNZ  LOOP           ; If result is not zero than jump to
                            loop.
        RET                 ; Go back to main program.
```

The 16-bit number XXX H loaded to DE register pair is as per the delay introduced in the program (discussed in chapter 4).

Example 7.7. *Write an assembly language program to input an 8 bit data serially through SID line of 8085 microprocessor. Convert it to 8 bit parallel data and store this data to 2500 H memory location.*

Solution. The program is given as:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI B,	00 H	; Clears Register B.
	MVI C,	08 H	; Preset counter to 8.
LOOP	RIM		; Get the bit through SID line.
	ANI	80 H	; Isolate the bit received through SID line.
	ORA B		; Convert to parallel word.
	RRC		; Rotate right.
	MOV B,	A	; Save the accumulator contents to B register.
	DCR C		; Decrement the contents of C register.
	JNZ	LOOP	; Go to LOOP if not zero.
	RLC		; Rotate left.
	STA	2500 H	; Store the parallel data to 2500 H memory location.
	HLT		

Example 7.8. *An 8 bit data (say 32 H) is to be outputted serially through SOD line of 8085. An LED connected to SOD line, should glow or not glow each time a bit (1 or 0) is outputted through SOD line. A delay of 1 sec is to be introduced between each transfer of a bit. Write an assembly language program to implement this.*

Solution. The program to output 8 bit data serially through SOD line is given below, which is self explanatory.

MAIN PROGRAM

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	00 H	; Loads the data 32 H to accumulator.
	MVI C,	08 H	; Preset counter to 8.
LOOP	RRC		; Rotate LSB to MSB.

```
        MOV  B,     A       ; Save the accumulator contents to B
                            register.
        ANI         80 H    ; Isolate SOD bit.
        ORI         40 H    ; Enable SOE bit for SIM.
        SIM                 ; Sends the bit through SOD line.
        MOV  A,     B       ; Save the present contents to
                            accumulator.
        CALL        DELAY   ; Call the delay for 1 sec.
        DCR  C              ; Decrement the contents of C
                            register.
        JNZ         LOOP    ; Go to LOOP if not zero.
        HLT
```

The delay subroutine program for 1 sec delay may be written as discussed in chapter 4.

SUBROUTINE PROGRAM:

```
Label   Mnemonics   Operand     Comments
DELAY   LXI  D,     F424 H  ; Loads DE register pair with a 16-
                            bit number F424 H.
LOOP    DCX  D              ; Decrements DE register pair by 1.
        MOV  A,     E       ; Moves the contents of E register to
                            accumulator.
        ORA  D              ; ORing of the contents of D and E
                            registers are performed to set the
                            zero flag.
        JNZ         LOOP    ; If result is not zero than jump to
                            loop.
        RET                 ; Go back to main program.
```

Example 7.9. *Write an assembly language program that is interrupted by applying a rising pulse at RST7.5 terminal manually. The program copies 256 bytes of data stored at memory locations starting at 3000 H to memory locations starting at 4000 H. The interrupt signal should, however, be able to introduce a delay of 1 sec. After the delay it should then move to main program. The main program for transferring the data is in a loop that repeats the same task again and again.*

Solution. Here is the program that implements the given task.

MAIN PROGRAM

```
Label   Mnemonics   Operand     Comments
        EI                  ; Enable interrupts
        MVI  A,     08 H    ; Enable RST7.5, RST6.5 and
                            RST5.5.
        SIM
LOOP    LXI  H,     3000 H  ; Load the H-L pair with the starting
                            source address.
        LXI  D,     4000 H  ; Load the D-E pair with the starting
                            address of destination address.
```

```
        MVI  B,     FF H      ; load the B-reg. with the bytes of
                                  data.
NXT     MOV  A,     M         ; Load the accumulator with the
                                  contents stored in memory location
                                  addressed by H-L register pair.
        STAX D                ; Store the accumulator contents to
                                  the destination address given by D-
                                  E register pair.
        INX  H                ; Increments the H-L pair for next
                                  number.
        INX  D                ; Increments the D-E pair
        DCR  B                ; Decrement B.
        JNZ         NXT       ; If not zero jump to NXT.
        JMP         LOOP      ; Jump to start.
        HLT
```

When the interrupt signal is enabled, it calls its vector location 003C H.

At the vector location say it is stored that JMP FFBD H i.e. monitor transfers the program to the memory location FFBD H. Now the user has to transfer from FFBD H to a memory location from where service subroutine for RST7.5 is written. The user transfers the program from FFBD H to 2000 H with the instructions JMP 2000 H. The location 2000 H indicates the starting address of interrupt service subroutine. The interrupt service subroutine is given as:

Interrupt Service Subroutine (at the starting address 2000 H)

```
Label   Mnemonics  Operand     Comments
        PUSH  PSW             : Save accumulator and flag contents
                                  in stack.
        PUSH  B               ; Save the contents of B-C register
                                  pair in stack.
        PUSH  D               ; Save the contents of D-E register
                                  pair in stack.
DELAY   LXI  D,     F424 H    ; Loads DE register pair with a 16-
                                  bit number F424 H.
LOOP    DCX  D                ; Decrements DE register pair by 1.
        MOV  A,     E         ; Moves the contents of E register to
                                  accumulator.
        ORA  D                ; ORing of the contents of D and E
                                  registers are performed to set the
                                  zero flag.
        JNZ         LOOP      ; If result is not zero than jump to
                                  loop.
        POP  D                ; Restore the contents of D-E
                                  register pair from the stack.
        POP  B                ; Restore the contents of B-C register
                                  pair from the stack.
```

```
POP  PSW          ; Restore the contents of
                    Accumulator and flag contents
                    form the stack.
EI                ; Enable interrupts
RET               ; Go back to main program.
```

Program written in ISS is the program for delay of one second as already discussed.

PROBLEMS

1. Discuss the memory Mapped I/O operation for the transfer of data from microprocessor to I/O devices and vice-versa.
2. Discuss the I/O Mapped I/O operation for the data transfer from microprocessor to I/O devices and vice-versa.
3. Give the comparison of memory mapped I/O and isolated I/O scheme fro the data transfer from microprocessor to I/O devices and vice-versa.
4. Why the problems arise when the data is transferred from the microprocessor to I/O devices and vice-versa? Mention various data transfer scheme.
5. Discuss programmed I/O data transfer scheme.
6. Describe Interrupt driven I/O data transfer scheme.
7. What do you mean by handshaking? How is it used in asynchronous data transfer between microprocessor and I/O devices?
8. What is DMA? Using block diagram explain how the data is transferred by a DMA controller.
9. Discuss DMA for 8085.
10. What is an interrupt? How data is transferred between the microprocessor and I/O devices.
11. What is the difference between software and hardware interrupts? Discuss software interrupts of 8085.
12. How is the INTR interrupt used in 8085?
13. Differentiate between
    (i) Memory mapped I/O and I/O mapped I/O data transfer schemes.
    (ii) Maskable and Non-maskable Interrupts
    (iii) Hardware and Software interrupts
14. Explain briefly the following:
    (i) DMA data transfer scheme
    (ii) Interrupt driven data transfer scheme
15. What are interrupt terminals available in 8085 microprocessor? How SIM and RIM instructions are used to set and read the interrupts.
16. What are hardware interrupts? What is meant by Vectored interrupts?
17. What are the functions of SID and SOD pins of 8085? How RIM and SIM instructions are used to input a bit through SID line and output a bit through SOD line.
18. Explain RIM and SIM instructions.
19. Discuss EI and DI instructions.
20. Write down the procedure to mask the RST6.5 interrupt.
21. Explain RST *n* interrupt circuit of 8085.
22. Discuss the bit pattern for RIM instruction.

23. Discuss the bit pattern for SIM instruction.
24. Draw and explain the interrupt control circuit for 8085 microprocessor.
25. Mention interrupt instructions of 8085. Discuss any two of them.
26. How will you enable all the interrupts of 805?
27. How will you enable RST5.5 and disable RST6.5 and RST7.5?
28. Write an assembly program of 8085 to generate asymmetrical square wave at the SOD line of 8085.
29. Write an assembly language program to input 256 bytes of data serially through SID line of 8085 microprocessor. Convert the 256 bytes of data received serially to parallel data and store the data to memory locations starting at 2500 H.
30. Here are some instructions

    MVI A, 1D H
    SIM

    After SIM instruction is executed, which interrupts are masked.

    (Ans.: RST 5.5 and RST 7.5 are masked.)
31. Here are some instructions

    MVI A, XX H
    SIM

    What should be the value of XX H in MVI instruction so that RST 5.5 and RST 6.5 are masked and all other don't care bits should be set to zero.

    (Ans.: XX H = 0B H)

8

Programmable Peripheral Interface (PPI) 8255A

The various methods of data transfer from the microprocessor to output devices or vice-versa has already been discussed in the preceding chapter. Special interface circuits, known as peripheral interface circuits are to be used for this purpose. The interfacing devices may be classified into two categories namely general purpose peripherals and special purpose peripherals. Basically the I/O devices to be connected to microprocessors are known as peripherals, these are printers, floppy drives, CRT and Cassette recorder etc. The general purpose peripherals are:

- Programmable Peripheral Interface (PPI)
- Programmable Interval Timer
- Programmable Interrupt Controller
- Programmable DMA Controller
- Programmable Communications Interface.

The special purpose peripherals used for interfacing a microprocessor to a specific type of I/O device are:

- Programmable Keyboard and Display Interface
- Programmable Hard Disk Controller
- Programmable Floppy Disk Controller

The present chapter will confine to the discussion of Programmable Peripheral Interface (PPI) IC 8255 A and its application.

8.1 DETAILS OF PPI IC 8255A

The input/output devices are generally interfaced to the microprocessor through the input/output port as shown in figure 8.1. The input/output port is either non-programmable or programmable. A non-programmable port can either be connected in input mode or output mode, i.e. if both input and output devices are to be connected to the microprocessor two separate non-programmable ports are to be used, one for input device and other for the output device. INTEL 8212 is an 8-bit non-programmable I/O port. Figures 8.2 (a) and (b) show the interfacing of 8212 in input and output mode respectively. However, a programmable I/O port can be programmed to act either as an input port or an output port. The INTEL 8255A is a programmable port device. It is most

versatile Programmable Peripheral Interface which may be connected to almost any

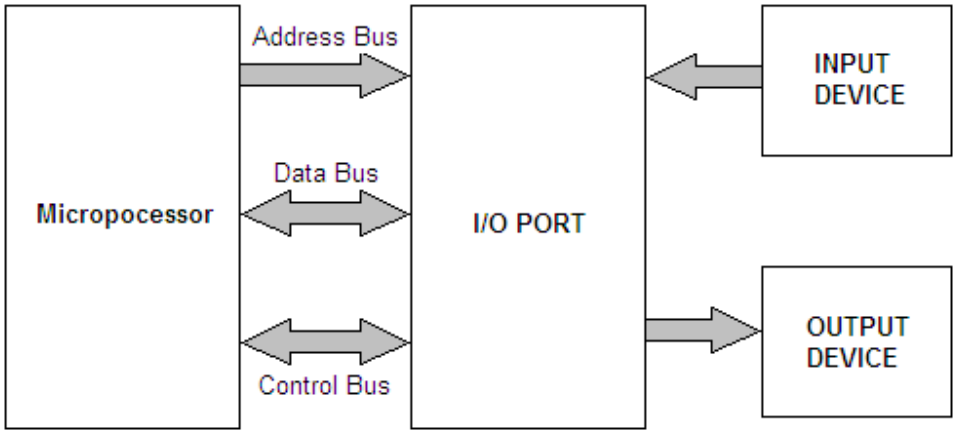


Fig. 8.1

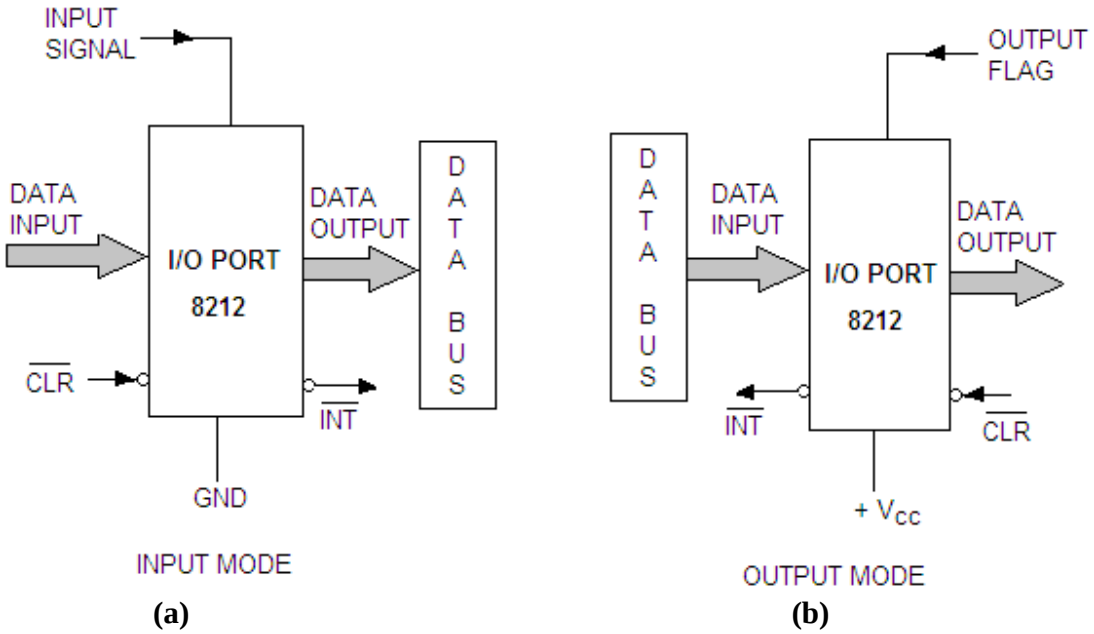


Fig. 8.2

microprocessor. This IC is widely used and can be programmed to transfer the data to the input/output devices. It is a 40 pin dual in line IC package, whose pin configuration and block diagram are shown in figures 8.3 and 8.4 respectively.

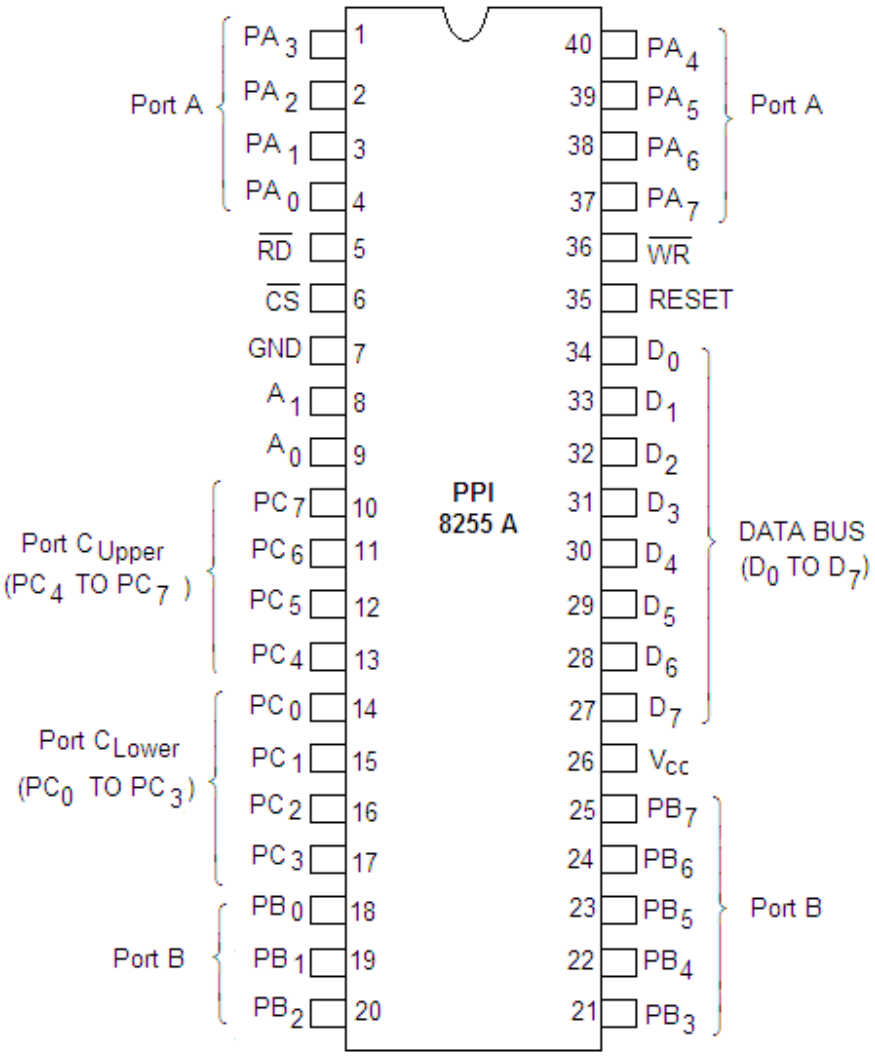


Fig. 8.3

The IC 8255 A has three 8-bit ports:

Port-A
Port-B
and Port-C

The port-C can be used into two 4-bit ports represented as Port C_{Upper} and Port C_{Lower}.

Port-A (PA_7-PA_0) and Port-C_{Upper} (PC_7-PC_4) together form Group A, whereas Port-B (PB_7-PB_0) and Port-C_{Lower} (PC_3-PC_0) form Group B as shown in figure 8.5.

After deciding the configuration of 3 ports (which port is to be used as input port and which port is to be used as output port), a control word of the command has to be given to the microprocessor. An 8-bit control word is formed for this purpose which may be transferred to the control word register of 8255 through the accumulator.

The control register has 6 control lines $\overline{RD}$, $\overline{WR}$, *RESET*, A_1, A_0 and $\overline{CS}$ whose functions are as given below:

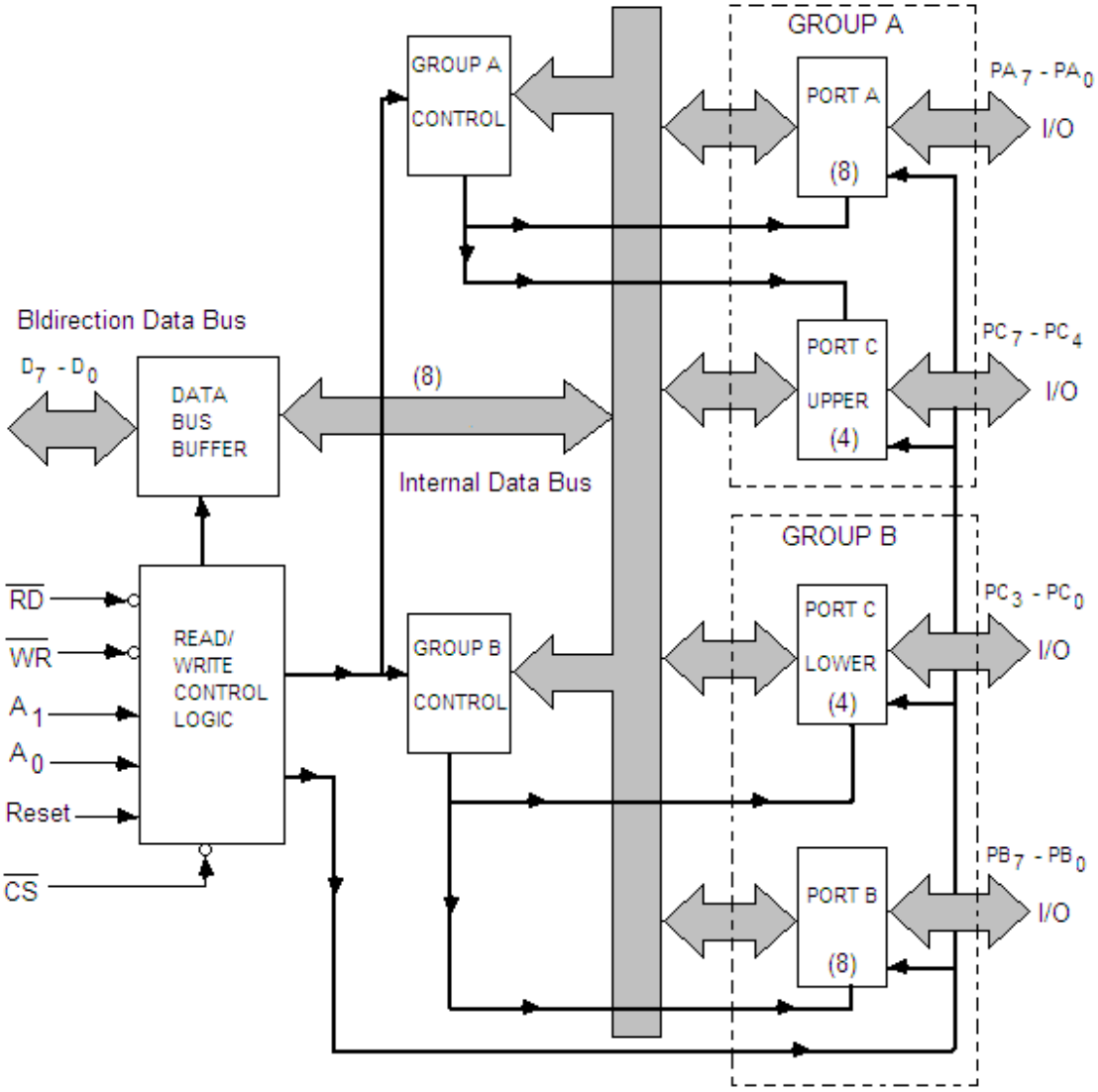


Fig. 8.4

(1) $\overline{\text{RD}}$ (Read): It is a read signal which is active low. When this signal goes low, it allows the microprocessor to read the data from the selected I/O ports of the 8255 PPI.

(2) $\overline{\text{WR}}$ (Write): It is the write signal which is also active low. When this signal goes low, the microprocessor writes (loads) the data into the selected I/O ports or the control register of 8255.

(3) RESET (Reset): This is a reset line which is active high. When a high signal appears on this line, it clears the control register and sets all the ports in the input mode.

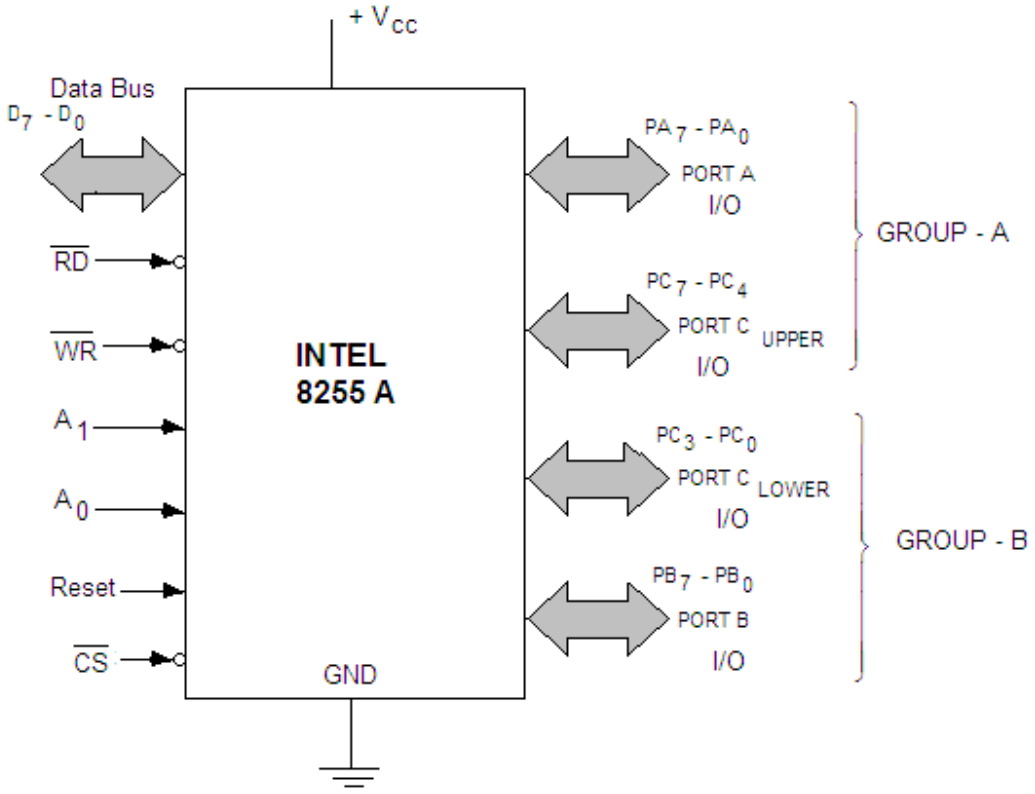


Fig. 8.5

(4) A_0 and A_1: These lines A_0 and A_1 are used to address the three ports and the control word as shown in table 8.1. If the $\overline{CS}$ (chip select) terminal of 8255 is low, the lines A_0 and A_1 decide the ports. The CPU can read or write into these registers by using $\overline{RD}$ and $\overline{WR}$ signals.

Table 8.1

A_1	A_0	
0	**0**	**Port-A**
0	**1**	**Port-B**
1	**0**	**Port-C**
1	**1**	**Control word register**

(5) $\overline{CS}$ (Chip Select): It is an active low chip select terminal. It is used to select 8255 by applying low signal to $\overline{CS}$ terminal. When a high signal appears on this line, the 8255 will not be selected, the data bus buffer that connects 8255 to the system data bus remains floated.

The 6 bits AD_2-AD_7 of address data bus (low order address bus) of the microprocessor are decoded to provide $\overline{CS}$; the remaining two bits AD_0-AD_1 may be used for the selection of control register or any of the three ports as shown in figure 8.6. Since the six bits of address data bus are decoded for $\overline{CS}$, so as many as $2^6 (= 64)$ PPI (8255) can be connected to any system. For the selection of any port of 8255 AD_1 and AD_0 bits of the address data bus are connected to A_1 and A_0 terminals of 8255.

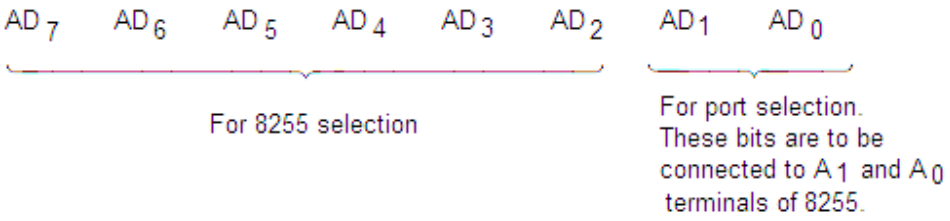


Fig. 8.6

Figure 8.7 shows the chip-select logic for 8255. From this logic diagram it is clear that if AD_7-AD_2 are all 0s then it enables $\overline{CS}$ to select this chip. The port selection will depend on the bits AD_1-AD_0 as given in table 8.2.

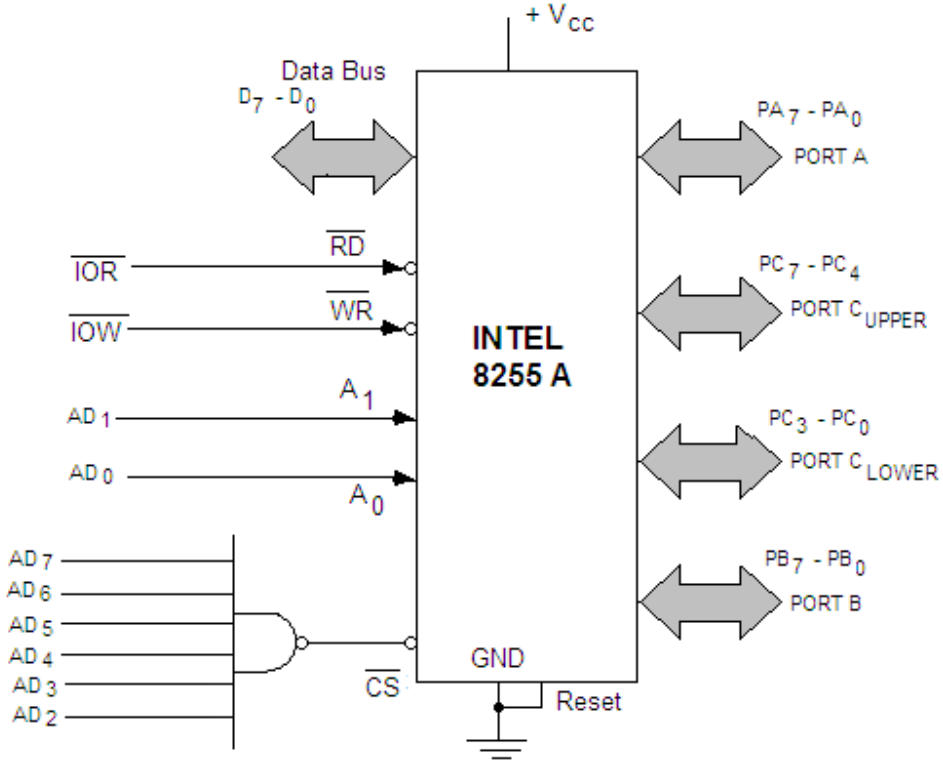


Fig. 8.7

Table 8.2

$\overline{CS}$									
AD_7	**AD_6**	**AD_5**	**AD_4**	**AD_3**	**AD_2**	**AD_1**	**AD_0**	**HEX ADDRESS**	**PORT NAME**
0	0	0	0	0	0	0	0	00 H	Port-A
0	0	0	0	0	0	0	1	01 H	Port-B
0	0	0	0	0	0	1	0	02 H	Port-C
0	0	0	0	0	0	1	1	03 H	Control word Register

Generally the microprocessor kits available with the laboratories have two 8255 connected with the system; 8255-I and 8255-II. Figure 8.8 shows the chip select logic for

the second 8255. If AD_7-AD_2 are chosen as per this diagram then the second 8255 (8255-II) will be selected. The port selection will depend on the bits AD_1-AD_0 as given in table 8.3. The M/S Vinytics, New Delhi has adopted this logic in their microprocessor kits.

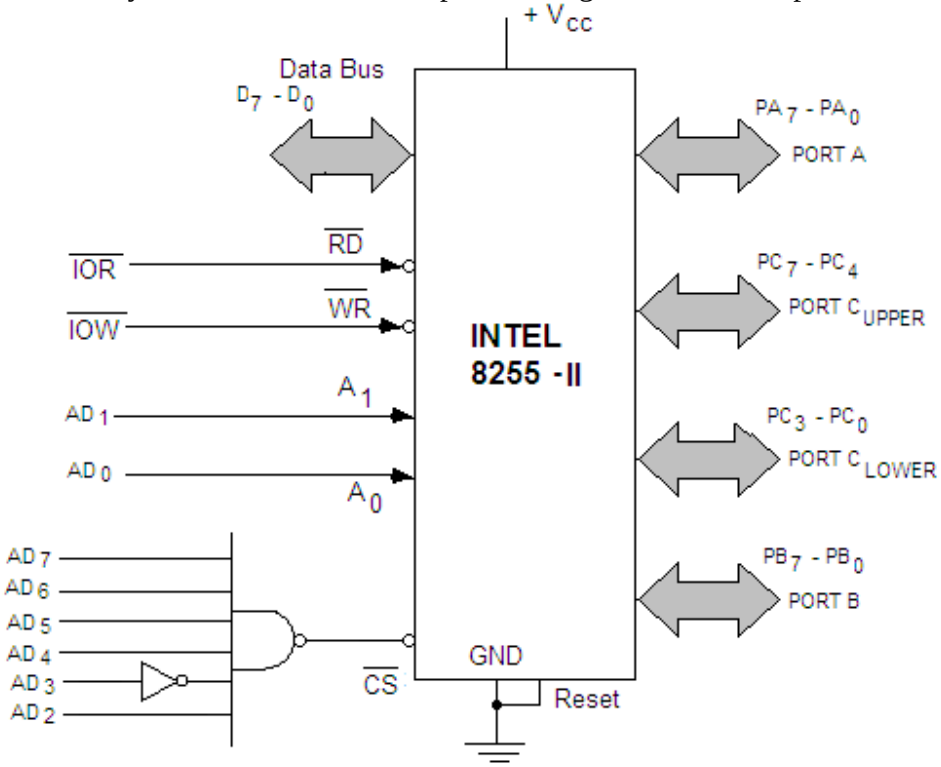


Fig. 8.8

Table 8.3

$\overline{CS}$									
AD_7	**AD_6**	**AD_5**	**AD_4**	**AD_3**	**AD_2**	**AD_1**	**AD_0**	**HEX ADDRESS**	**PORT NAME of 8255-II**
0	0	0	0	1	0	0	0	08 H	Port-A
0	0	0	0	1	0	0	1	09H	Port-B
0	0	0	0	1	0	1	0	0A H	Port-C
0	0	0	0	1	0	1	1	0B H	Control word Register

It may be mentioned here that if the instruction OUT 03 H is executed then it transfers the contents of accumulator to the control word register of 8255-I. Similarly, if the instruction OUT OB H is executed then the accumulator contents will be transferred to control word register of 8255-II. For the control word, accumulator is loaded with the contents that are necessary to use the ports of 8255 as input port or output port. The procedure for the generation of control word will be discussed in a later section of this chapter. Now if 8255-I is initialized to use Port A as input port and ports B and C as output ports, then the execution of IN 00 H instruction will transfer the data from port A

of 8255-I to the accumulator. The execution of OUT 01 H instruction will transfer the accumulator contents to the port B of 8255-I. Similarly, 8255-II may be initialized by giving the proper control word to the control word register by using the instruction OUT 0B H. The IN or OUT instruction having the port address as 08, 09 or 0A may be used for the data transfer through 8255-II.

8.2 OPERATIONAL MODES OF 8255A

The operational modes of 8255 A PPI can be classified into two broad groups (fig. 8.9).

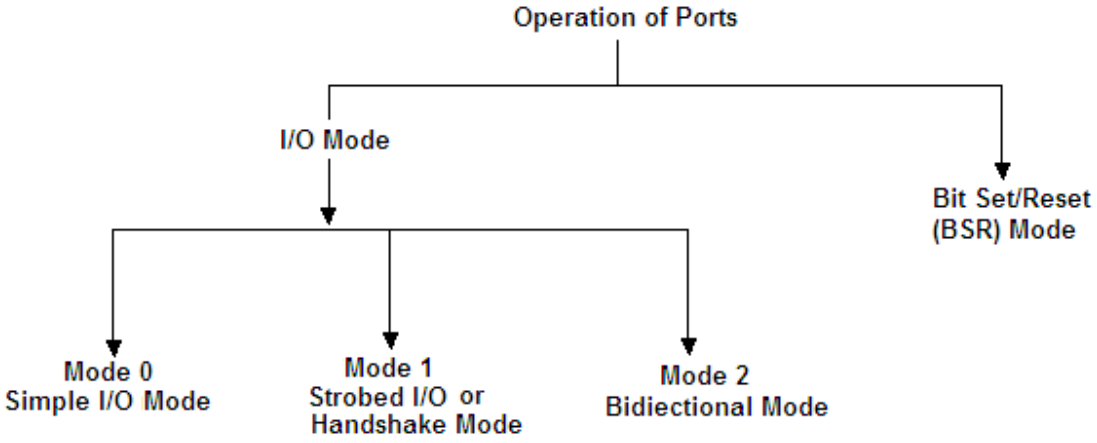


Fig. 8.9

1. I/O Mode

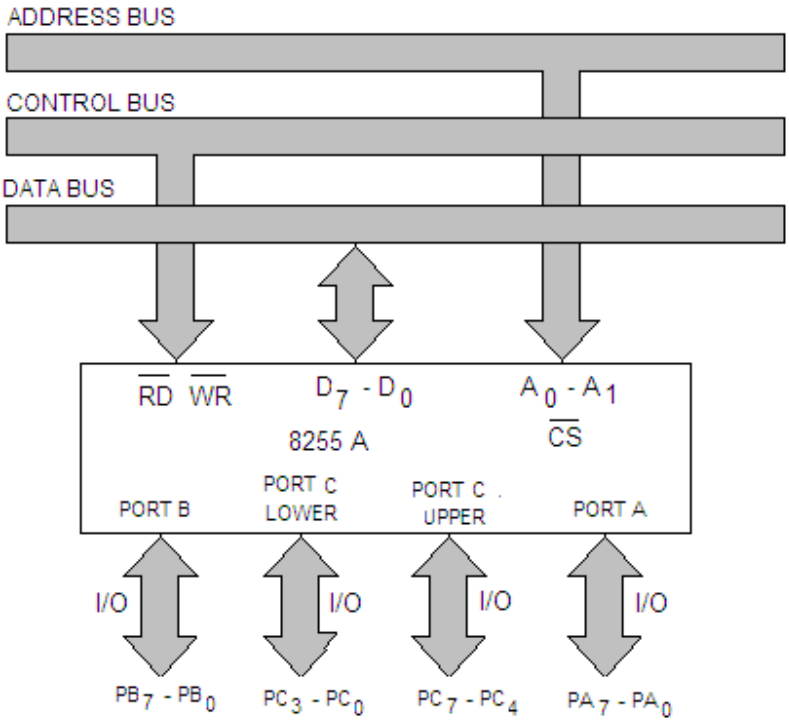
The input/output mode can further be subdivided into three groups:

- **Mode 0 – Simple Input/output mode**

  In this mode the 8 bit port A (PA_0-PA_7) can be configured as input or output port. The port B (PB_0-PB_7) can also be configured, in the similar fashion, as input or output operation. However there is flexibility for the port C. It can be divided into two 4 bit ports, the port C_{Lower} (PC_0-PC_3) and port C_{Upper} (PC_4-PC_7). Each of them can be set independently for input or output operation. In this way there are four ports (port-A, port-B, C_{Lower} and port C_{Upper}) and each of them can be set either as an input port or an output port. Here these ports are simple input or output ports i.e. these ports can work without handshaking. In this mode the outputs are latched whereas the inputs are not latched. The basic definition of this mode is shown in figure 8.10.

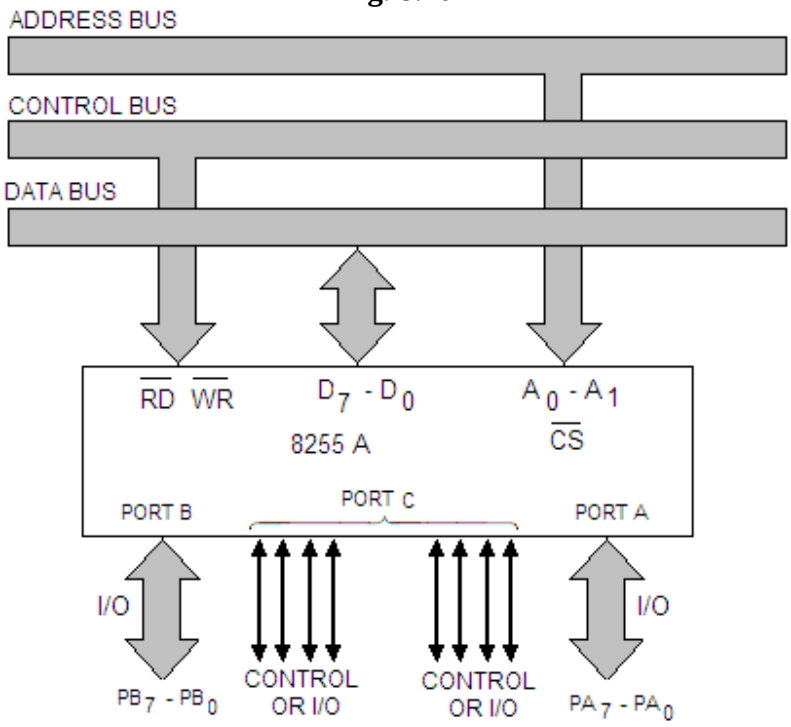
- **Mode 1 – Strobed Input/output or Handshake mode**

  In this mode of operation handshaking is used for the input or output data transfer. As discussed earlier, there are two groups in 8255 PPI: Group A and Group B. Both these groups have one 8-bit port and one 4-bit port. Port-A and Port C_{Upper} form group A whereas Port-B and Port C_{Lower} form group B. The 8-bit port of each group can be programmed for input or output operation with latched input and latched output facilities. The bits of Port C are used for handshaking. Figure 8.11 shows the basic definition of this mode.



Mode-0 Operation

Fig. 8.10



Mode-1 Operation

Fig. 8.11

- **Mode 2 – Bidirectional Mode**

  In this mode Port A can be programmed to operate as a bidirectional port. When Port A is programmed in this mode of operation, Port B can be used either in Mode 0 or Mode 1. For mode 2 operation PC_3 to PC_7 bits are used for handshaking. In this mode too both inputs and outputs are latched. The basic definition of this mode is shown in figure 8.12.

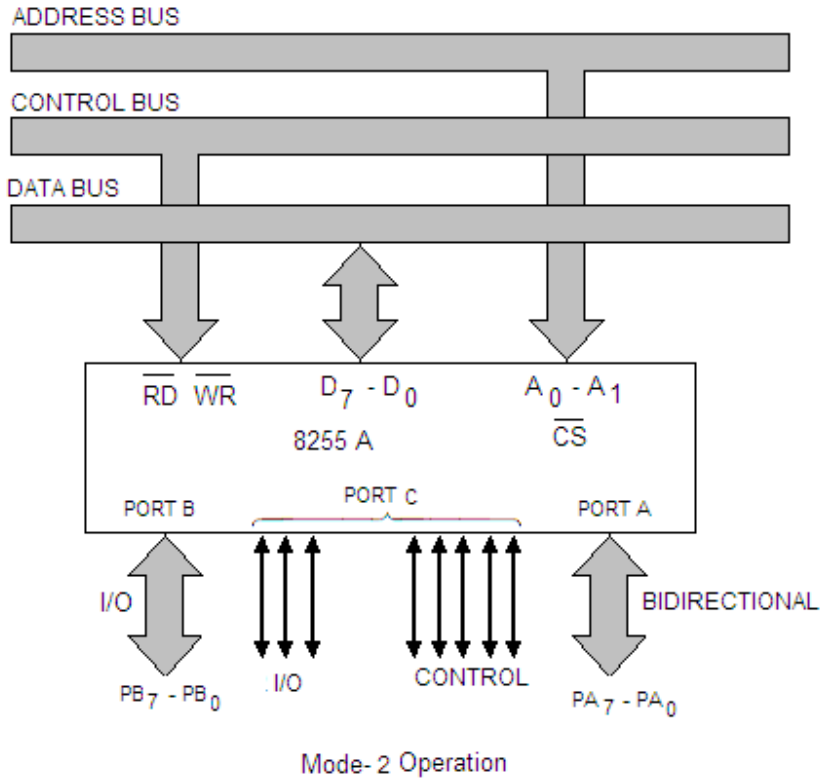


Fig. 8.12

2. Bit Set/Reset Mode

The modes of operation of 8255 PPI may be selected by the software. The details of these modes will be discussed in the following sections.

8.3 CONTROL WORD FORMAT FOR 8255A

As discussed above, a port can be programmed to act as an input port or an output port. A control word is, therefore, to be formed for programming the ports of 8255A. The format for the control word is shown in figure 8.13. The control word, as per the requirement of the programmer, is written into the control word register of 8255A. No read operation of the control word is allowed. The description of the control bits is given below:

Bit D_0 Sets Port C_{Lower} as input or output port.
To make Port C_{Lower} as input port this bit is set to 1.
To make Port C_{Lower} as output port this bit is set to 0.

Bit D_1 Sets Port B as input or output port.
To make Port B as input port this bit is set to 1.
To make Port B as output port this bit is set to 0.

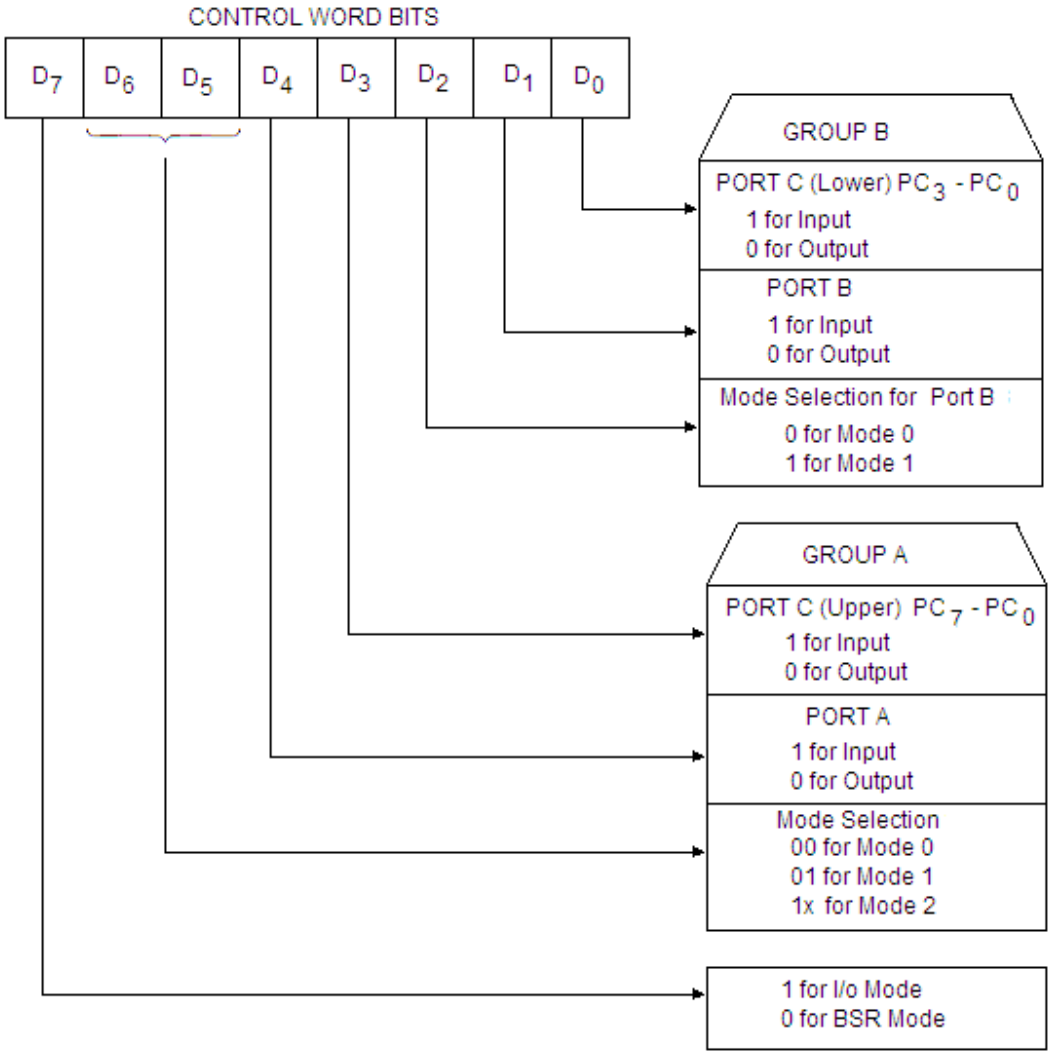


Fig. 8.13

Bit D_2 This bit is for mode selection for the port B.
If this bit is set to 0, the port B will operate in mode 0.
If this bit is set to 1, the port B will operate in mode 1.

Bit D_3 Sets Port C_{Upper} as input or output port.
To make Port C_{Upper} as input port this bit is set to 1.
To make Port C_{Upper} as output port this bit is set to 0.

Bit D_4 Sets Port A as input or output port.
To make Port A as input port this bit is set to 1.
To make Port A as output port this bit is set to 0.

Bits D_5 and D_6 These two bits are used to determine the I/O mode of port A.
These bits are defined for the various modes of port A as follows:

D_6	D_5	Mode of port A
0	0	Mode 0
0	1	Mode 1
1	0 or 1	Mode 2

Bit D_7 This bit specifies either I/O function or bit set/reset function (BSR mode).

If this bit is set to 1 then the 8255 will work in I/O mode.
If this bit is set to 0 then the 8255 will work in BSR mode.

There are 16 combinations of control words for various configurations of the ports of 8255 for Mode 0 operations. These control words are shown in table 8.4.

Table 8.4

Control Word Bits								Control Word	PORT A	PORT C_{Upper}	PORT B	PORT C_{Lower}
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0					
1	0	0	0	0	0	0	0	80 H	Output	Output	Output	Output
1	0	0	0	0	0	0	1	81 H	Output	Output	Output	Input
1	0	0	0	0	0	1	0	82 H	Output	Output	Input	Output
1	0	0	0	0	0	1	1	83 H	Output	Output	Input	Input
1	0	0	0	1	0	0	0	88 H	Output	Input	Output	Output
1	0	0	0	1	0	0	1	89 H	Output	Input	Output	Input
1	0	0	0	1	0	1	0	8A H	Output	Input	Input	Output
1	0	0	0	1	0	1	1	8B H	Output	Input	Input	Input
1	0	0	1	0	0	0	0	90 H	Input	Output	Output	Output
1	0	0	1	0	0	0	1	91 H	Input	Output	Output	Input
1	0	0	1	0	0	1	0	92 H	Input	Output	Input	Output
1	0	0	1	0	0	1	1	93 H	Input	Output	Input	Input
1	0	0	1	1	0	0	0	98 H	Input	Input	Output	Output
1	0	0	1	1	0	0	1	99 H	Input	Input	Output	Input
1	0	0	1	1	0	1	0	9A H	Input	Input	Input	Output
1	0	0	1	1	0	1	1	9B H	Input	Input	Input	Input

8.4 PROGRAMMING IN MODE 0

As discussed earlier, the Ports A, B, and C can be configured as simple input or output ports by writing the appropriate control word in the control word register. In the control word D_7 is set to 1 as 8255A is to be used in simple I/O mode, D_6, D_5 and D_2 are

all set to 0 as all the ports are to be used in mode 0. The remaining bits D_4, D_3, D_1 and D_0 determine if the corresponding ports are to be used as input port or output port (1 for input port and 0 for output port). Since these are four bits to decide which ports are to be used as input ports and which ports are to be used as output ports, so for this purpose, there will be 16 possible combinations of control words listed in table 8.4.

For example, the control word when the ports of 8255A PPI are to be used in mode 0 with Port A as input port, port B as output port, Port C_{Upper} as output port and Port C_{Lower} as input port is given as:

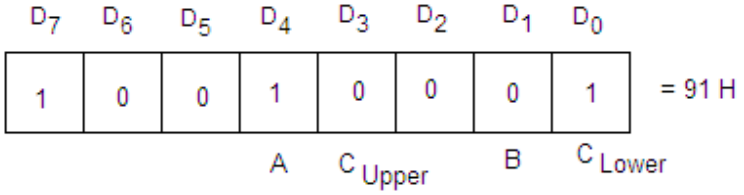


Fig. 8.14

The control word 91 H is to be loaded to the control word register (CWR) of 8255A through the OUT instruction. Let the control word 91 H is to be loaded to control word register of 8255-I connected to the system (port address is 03 H discussed earlier). It may be done with the following instructions:

MVI A, 91 H
OUT 03 H

Now with OUT or IN instructions, the data may be transferred to or from the output devices. The proper port address is to be given with these instructions.

Example 8.1. *Obtain the control word when the ports of 8255A are to be used in mode 0 with port-A as input port and port B and port C as output port.*

Solution. The control word for this case is given as:

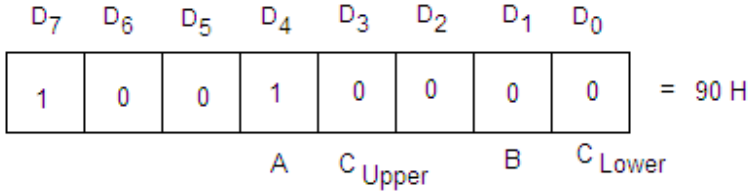


Fig. 8.15

Example 8.2. *Obtain the control word when the ports of 8255A are to be used in mode 0 with port-A as output port and port B as input port and port C as output port.*

Solution. The control word for this case is given as:

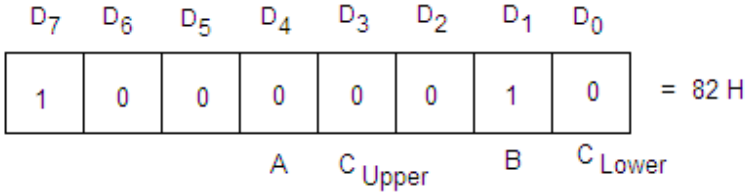


Fig. 8.16

Example 8.3. *Make control word when the ports of 8255A are to be used in mode 0 with all the ports as output ports.*

Solution. The control word for this case is given as:

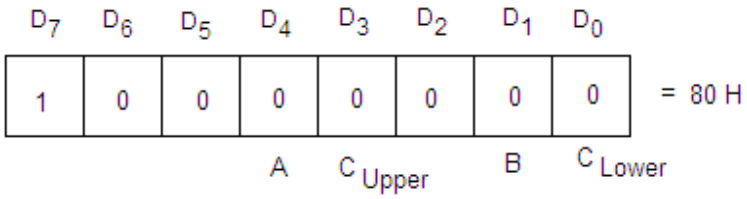


Fig. 8.17

Example 8.4. *Write an assembly language program to generate a square ware of 1 KHz frequency using 8255A. The wave should be available at PA_0 terminal of Port-A.*

Solution. The Program to perform the task given in the problem is shown below:

MAIN PROGRAM:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	80 H	; Initialize 8255-I to work all the ports as output ports.
	OUT	03 H	; Write the control word (80 H) in the control word register of 8255-I.
LOOP	MVI A,	00 H	; Load 00 H to accumulator.
	OUT	00 H	; Send 00H (0V) to PA_0.
	CALL	DELAY	; Jump to dealy subroutine to introduce a delay of 0.5 msec.
	MVI A,	01 H	; Load 01H to accumulator.
	OUT	00 H	; Send 01 H (5V) to PA_0.
	CALL	DELAY	; Jump to dealy subroutine to introduce a delay of 0.5 msec.
	JMP	LOOP	; Jump to loop to repeat the process.

SUBROUTINE PROGRAM:

Label	**Mnemonics**	**Operand**	**Comments**
DELAY	LXI D,	003FH	; Loads DE register pair with a 16-bit number (63_{10}).
LOOP1	DCX D		; Decrements DE register pair by 1.
	MOV A,	E	; Moves the contents of E register to accumulator.
	ORA D		; ORing of the contents of D and E registers are performed to set the zero flag.
	JNZ	LOOP1	; If result is not zero than jump to LOOP1.
	RET		; Go back to main program.

The subroutine program introduces a delay of approximately 0.5 msec (discussed in section 4.3). The PA_0 terminal will be low for 0.5 msec and high for 0.5 msec. The total time will be 1 msec and thus a signal of 1 KHz will be generated at PA0 of port-A. A CRO may be connected to this terminal to observe the wave.

Example 8.5. *Write an assembly language program to glow 8-LEDs sequentially with a delay of one second. The LEDs are connected to 8 bits of port-B of 8255-II attached with the system.*

Solution. The program to perform the task of the problem is given below:

MAIN PROGRAM:

Label	Mnemonics	Operand	Comments
	MVI A,	80 H	; Initialize 8255-II to work all the ports as output ports.
	OUT	0B H	; Write the control word in the control word register of 8255-II.
	MVI A,	01 H	; Load 01 H to accumulator.
REPEAT	OUT	09 H	; Send 01H (5V) to PB_0 for the glow of LED
	CALL	DELAY	; Jump to delay subroutine - to introduce a delay of 1 sec.
	RLC		; Rotate the contents of accumulator left for sequential glow of LEDs.
	JMP	REPEAT	; Jump to Repeat for the next LED to glow.

SUBROUTINE PROGRAM:

Label	Mnemonics	Operand	Comments
DELAY	PUSH PSW		; Send the contents of Acc and flag register to stack.
	MVI C,	02 H	; Load C register with 02 H data.
LOOP	LXI D,	F424 H	; Loads DE register pair with a 16-bit number (63_{10}).
LOOP1	DCX D		; Decrements DE register pair by 1.
	MOV A,	E	; Moves the contents of E register to accumulator.
	ORA D		; ORing of the contents of D and E registers are performed to set the zero flag.
	JNZ	LOOP1	; If result is not zero than jump to LOOP1.
	DCR C		; Decrement the contents of C register.
	JNZ	LOOP	; If the result is not zero Jump to LOOP.
	POP PSW		; Acc and flag contents are popped from the stack.
	RET		; Go back to main program.

The subroutine program given here is already discussed in section 4.3.

In place of RLC in the main program RRC may be written, which will allow the LEDs to glow in the opposite sequence.

Example 8.6. *Write a program in assembly language of 8085, to switch on an a.c. bulb after a delay of 1 Min. A relay circuit may be connected to PA_0 of 8255-I to switch on the bulb.*

Solution. The relay circuit may be connected to PA_0 of 8255A as shown in figure 8.18. When PA0 is made high (through software), the transistor T_1 goes into saturation and thus energizes the relay. The normally open (N/O) terminals of the relay get connected

together and the bulb may be switched on. To provide high (logic 1) to PA0, the program in assembly language of 8085 is written as:

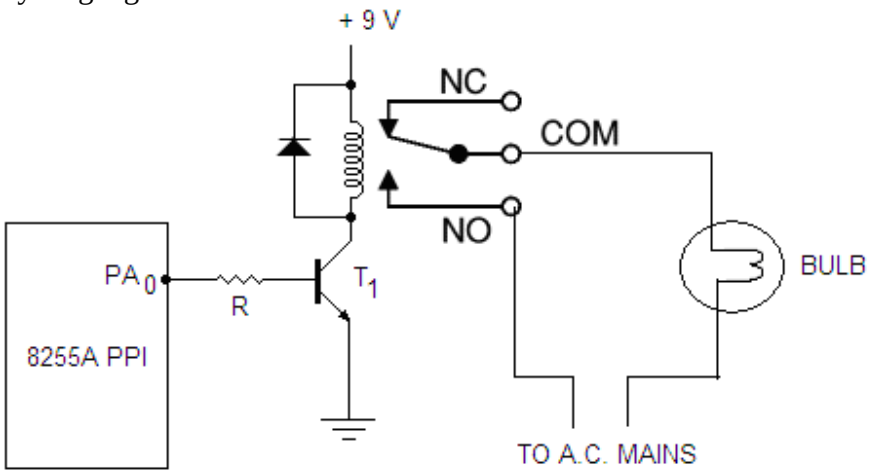


Fig. 8.18

MAIN PROGRAM:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	80 H	; Initialize 8255-I to work all the ports as output ports.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	MVI A,	00 H	; Load 00 H to accumulator.
	OUT	00 H	; Send 00H (0V) to PA_0 for the bulb to remain OFF.
	CALL	DELAY	; Jump to delay subroutine - to introduce a delay of 1 Min.
	MVI A,	01 H	; Load 01 to accumulator.
	OUT	00 H	; Send 01 (5V) to PA0 for the bulb to ON.
	HLT		

SUBROUTINE PROGRAM:

Label	**Mnemonics**	**operand**	**Comments**
DELAY	MVI C,	78 H	; Load C register with 78 H (120_{10})data so that the delay of 0.5sec is run 120 times (60sec).
LOOP	LXI D,	F424 H	; Loads DE register pair with a 16-bit number (63_{10}).
LOOP1	DCX D		; Decrements DE register pair by 1.
	MOV A,	E	; Moves the contents of E register to accumulator.
	ORA D		; ORing of the contents of D and E registers are performed to set the zero flag.
	JNZ	LOOP1	; If result is not zero than jump to LOOP1.

```
        DCR  C                  ; Decrement the contents of C
                                  register.
        JNZ      LOOP           ; If the result is not zero Jump to
                                  LOOP.
        RET                     ; Go back to main program.
```

Example 8.7. *Eight switches are connected to 8 bits of 8255 A (assume that the switches are debouncer switches) and 8 LEDs are connected to the 8 bits of Port B of 8255 A. Write an assembly language program that whenever any of the switch is depressed the corresponding LED glows. For example, if the switch number 3 is switched on, the LED connected to bit 3 should glow; similarly for the other.*

Solution. The program to perform the task of the problem is given below:

MAIN PROGRAM:

Label	Mnemonics	Operand	Comments
	MVI A,	99 H	; Initialize 8255-I to make port A and port C as input ports and port B as output port.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
REPEAT	IN	00 H	; Send 01H (5V) to PB_0 for the glow of LED
	OUT	01 H	; Output the input data to glow the LED corresponding to depressed switch.
	JMP	REPEAT	; Jump to Repeat.

8.5 PROGRAMMING IN MODE 1 (STROBED INPUT/OUTPUT)

As already discussed, in Mode 1 handshake (Strobes) signals are used for the data transfer between the microprocessor and I/O devices. The handshake signals are exchanged between the microprocessor and peripherals. Both group A and group B of 8255 can be made to operate in Mode 1. The two ports (Port A and Port B) can be configured either as input port or output port. Each port (Port A and Port B) uses three lines of port C as handshake signals (i.e. total 6 lines of port C are used for handshake signals). The remaining two lines of port C can be used for simple I/O operation.

Bits PC_0, PC_1 and PC_2 of port C are used as handshake signals for port B in input and output modes. Bits PC_3, PC_4 and PC_5 of port C are used as handshake signals for port-A in input mode only. The remaining bits of port C (PC_5 and PC_6) are used as simple I/O operation as programmed by bit D_3 of the control word.

Bits PC_3, PC_6 and PC_7 of port C are used as control (handshake) signals for port A in output mode. In this case bits PC_4 and PC_5 of port C are used as simple I/O operation as programmed by bit D_3 of the control word. The remaining bits PC_0, PC_1 and PC_2 of port C are used for handshake signals for port B.

The combination of Mode 1 and Mode 0 operation is also possible. For example, when port A is programmed to operate in mode 1, the port B can be operated in mode 0.

8.5.1 Input Control Signals in Mode 1

Figure 8.19(a) shows the input configuration of both Port A and Port B to be used in mode 1. The control signals used for handshaking are also shown in this figure. Bits

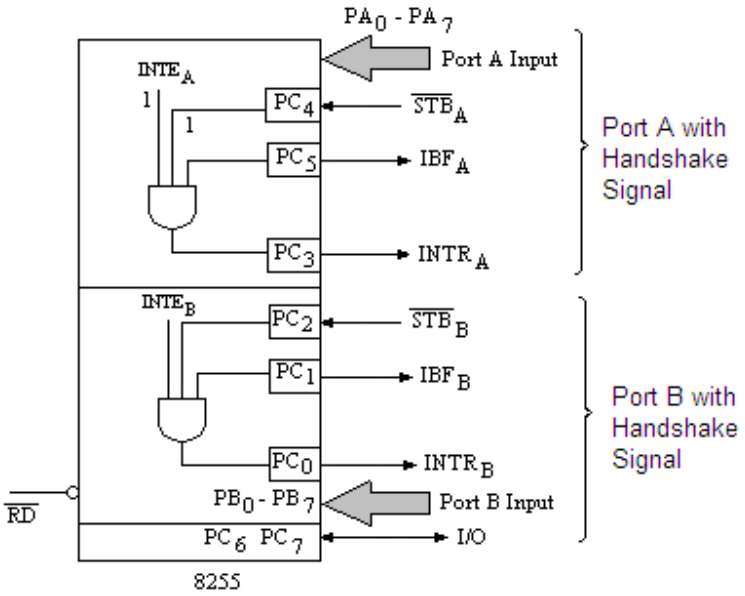
PC_3, PC_4 and PC_5 of port C are used for handshaking when port A is used as input port; and bits PC_0, PC_1 and PC_2 are used for handshaking when port B is also used for input port. Bits PC_6 and PC_7 are used for simple input/output operation. The control word for using Port A and Port B as input ports in mode 1 is shown in figure 8.19(b).

The functions of three control signals used for handshaking are given as:

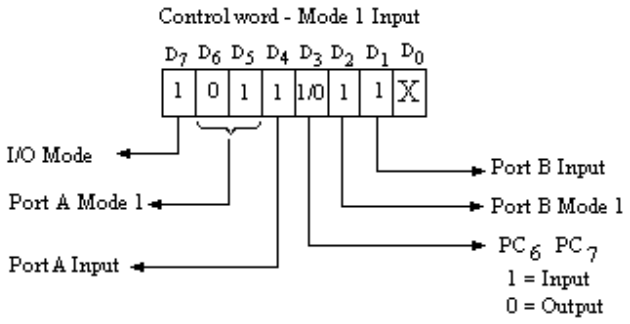
$\overline{\text{STB}}$ (Strobe Input) This is an active low signal generated by the peripheral device to indicate that it has transmitted a byte of data. In response to this signal, 8255 generates IBF (Input Buffer Full) and INTR (Interrupt Request) signals.

IBF (Input Buffer Full) This is an acknowledge signal sent by 8255. A high to this signal indicates that the input latch has received the data byte.

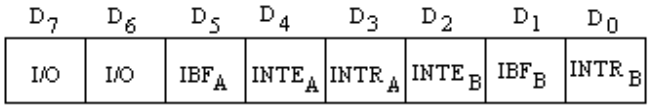
INTR (Interrupt Request) This is an output signal sent by 8255 which may be used to interrupt the microprocessor. This signal is generated when $\overline{\text{STB}}$, IBF and INTE (Interrupt Enable) are all low (logic 0). At the falling edge of the $\overline{\text{RD}}$ signal, this is reset.



(a)

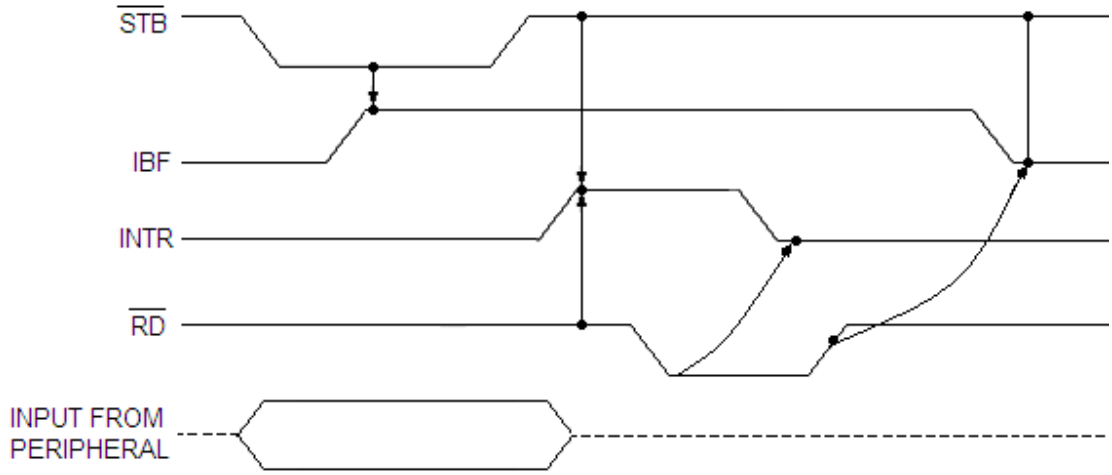


(b)



Status Word - Mode 1 Input

(c)



(d) **Fig. 8.19**

INTE (Interrupt Enable) This is an internal flip-flop which is used to enable or disable the generation of INTR signal. The two flip-flops $INTE_A$ and $INTE_B$ are set/reset using the BSR mode. The bit PC_4 enables or disables $INTE_A$; and PC_2 enables or disables $INTE_B$.

Figure 8.19(c) shows the status word (i.e. the status of the signals INTR, INTE and IBF signals etc.) which will be placed in the accumulator if the port C is read.

As shown in figure 8.19(d), the peripheral first loads data into the input port A or B by making $\overline{STB}$ input low. This low $\overline{STB}$ signal is generated by the peripheral device to indicate that it has transmitted a byte of data. The 8255A generates IBF and INTR in response to the active low $\overline{STB}$ signal.

8.5.2 Output Control Signals in Mode 1

As already discussed Bits PC_3, PC_6 and PC_7 of port C are used as control (handshake) signals for port A in output mode. In this case bits PC_4 and PC_5 of port C are used as simple I/O operation as programmed by bit D_3 of the control word. The remaining bits PC_0, PC_1 and PC_2 of port C are used for handshake signals for port B.

Figure 8.20(a) shows the input configuration of both Port A and Port B to be used in mode 1. The control signals used for handshaking are also shown in this figure. Bits PC_3, PC_6 and PC_7 of port C are used for handshaking when port A is used as output port; and bits PC_0, PC_1 and PC_2 are used for handshaking when port B is also used for output port. Bits PC_4 and PC_5 are used for simple input/output operation. The control word for using Port A and Port B as input ports in mode 1 is shown in figure 8.20(b).

The functions of three control signals used for handshaking are given as:

$\overline{OBF}$ (Output Buffer Full) This is an active low signal; it activates when the microprocessor writes data into output latch of 8255. This

indicates to an output peripheral that a new data is ready to be read. It goes high again after the 8255 receives a $\overline{ACK}$ signal from peripheral.

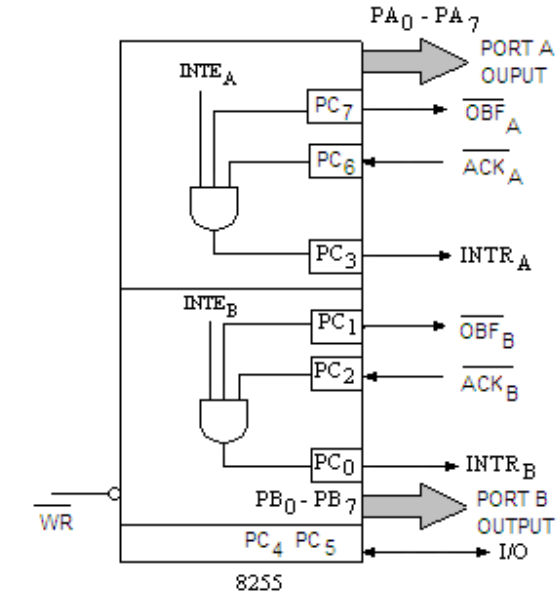
$\overline{ACK}$ (Acknowledge) This is an input signal from a peripheral. It becomes active (low) when peripheral receives the data from 8255 ports.

INTR (Interrupt Request) This is an output signal which is set at the rising edge of the $\overline{ACK}$ signal. This signal can be used to interrupt the microprocessor. It requests for the next data byte for output. This signal is generated when $\overline{OBF}$, $\overline{ACK}$ and INTE (Interrupt Enable) are all high (logic 1). At the falling edge of the $\overline{WR}$ signal this is reset.

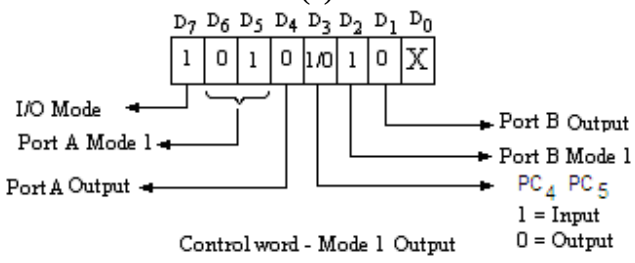
INTE (Interrupt Enable) This is an internal flip-flop to a port and required to be set to generate the INTR signal. The two flip-flops $INTE_A$ and $INTE_B$ are set/reset using the BSR mode.

Figure 8.20(c) shows the status word (i.e. the status of the signals INTR, INTE and OBF signals etc.) which will be placed in the accumulator if the port C is read.

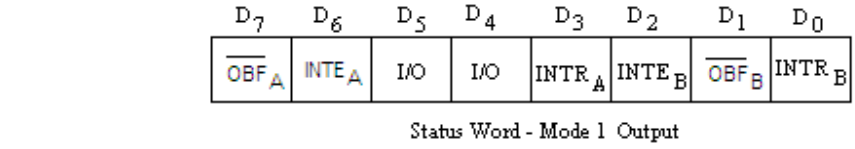
Figure 8.20(d) shows the timing diagram for strobed Output.



(a)

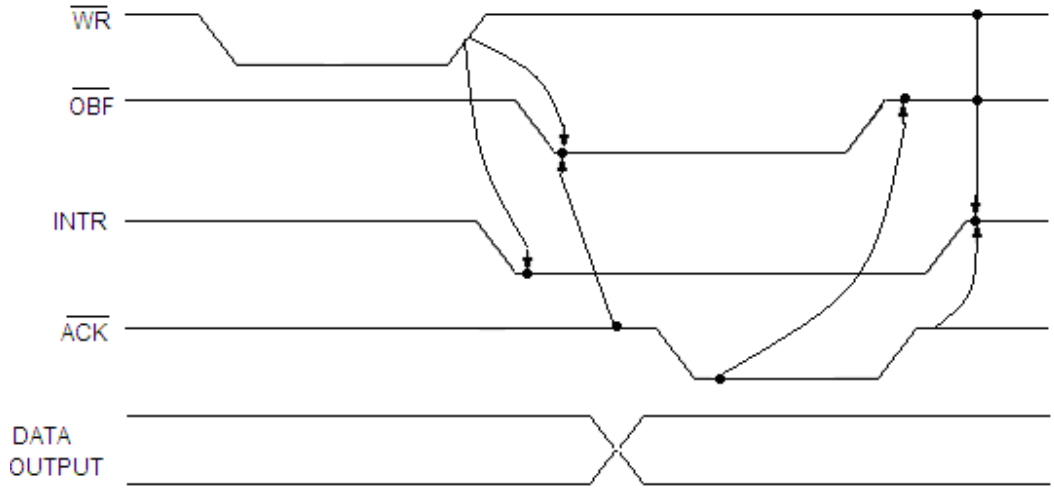


(b)



Status Word - Mode 1 Output

(c)

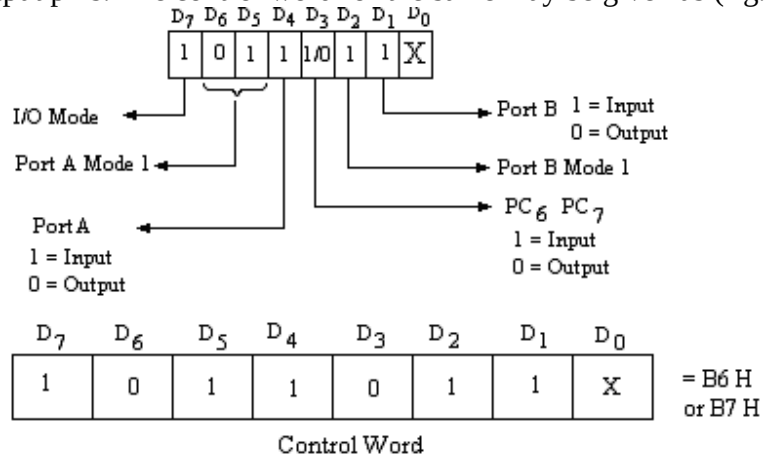


(d) **Fig. 8.20**

Example 8.8. *Obtain the control word for the following configuration of the ports of 8255 A for mode 1 operation:*

Port A – as input port,
Mode for Port A – mode 1
Port B – input port
Mode for Port B – mode 1,
The remaining pins of port C_{Upper} *are to be used output pins.*

Solution. The 8255 A is to be used in mode 1, and port A as input port so PC_3, PC_4 and PC_5 pins of port C will be used as the handshake signals. For port B as input port the PC_0, PC_1 and PC_2 pins of port C will be used as the handshake signals. The remaining pins of port C_{Upper} will be used as simple input output mode. In the present example PC_6 and PC_7 are to be used as output pins. The control word for the same may be given as (fig. 8.21):



Control Word

Fig. 8.21

Example 8.9. *Obtain the control word for the following configuration of the ports of 8255 A for mode 1 operation:*

Port A – as output port,
Mode for Port A – mode 1
Port B – output port,
Mode for Port B – mode 1,
The remaining pins PC_4 and PC_5 of port C are to be used input pins.

Solution. The 8255 A is to used in mode 1, and port A as output port so PC_3, PC_6 and PC_7 pins of port C will be used as the handshake signals. For port B as output port the PC_0, PC_1 and PC_2 pins of port C will be used as the handshake signals. The remaining pins of port C will be used as simple input output mode. In the present example PC_4 and PC_5 are to be used as input pins. The control word for the same may be given as (fig. 8.22):

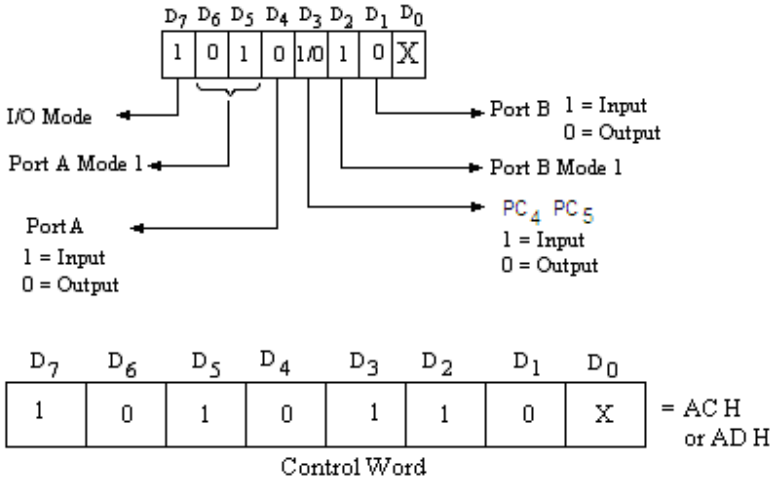


Fig. 8.22

Example 8.10. *Obtain the control word for the following configuration of the ports of 8255 A:*

Port A – as input port,
Mode for Port A – mode 1
Port B – input port,
Mode for Port B – mode 0,
Port C_{Lower} – output port
The remaining pins PC_6 and PC_7 of port C are to be used input pins.

Solution. The 8255 A is to used in mode 1 and mode 0, and port A as input port in mode 1 so PC_3, PC_6 and PC_7 pins of port C will be used as the handshake signals. The port B is to be used as input port in mode 0. The Port C_{Lower} is used as output port and remaining bits of port C (PC_6, PC_1) as input pins. The control word for the same may be given as (fig. 8.23):

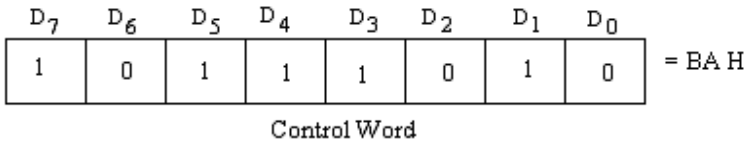


Fig. 8.23

8.6 PROGRAMMING IN MODE 2 (STROBED BIDIRECTIONAL BUS I/O)

In mode 2 operation of 8255A, port A can be used as bidirectional 8 bit I/O bus. The port B can operate in mode 0 or mode 1.

When port A is programmed to operate in mode 2 (bidirectional I/O bus), the five bits of port C (PC_3-PC_7) are used as control (handshake) signals for port A. This is illustrated in figure 8.24. In this case port B can be programmed in mode 0 or mode 1. If port B is used in mode 0, the remaining bits of port C (PC_0-PC_2) are used as input or output. However, if port B is used in mode 1, PC_0-PC_2 bits are used as handshake signals for port B.

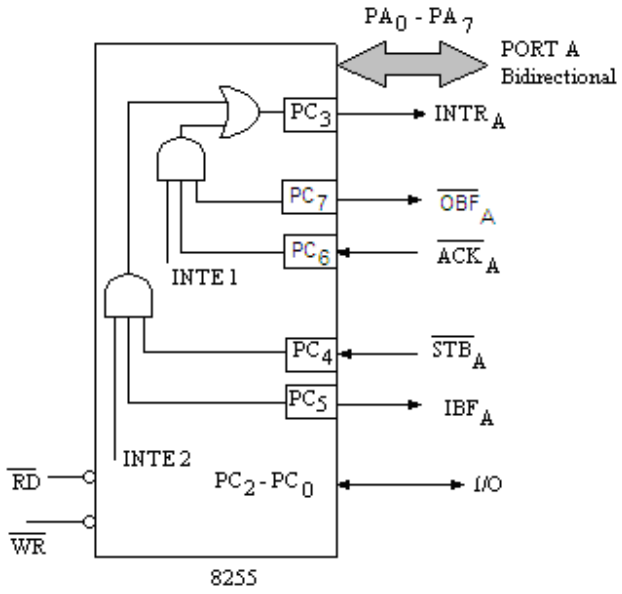


Fig. 8.24

The functions of five control signals used for handshaking of port A are given as:

$\overline{\text{STB}}$ (Strobe Input) This is an active low signal generated by the peripheral device to indicate that it has transmitted a byte of data. In response to this signal, 8255 generates IBF (Input Buffer Full) and INTR (Interrupt Request) signals.

IBF (Input Buffer Full) This is an acknowledge signal sent by 8255. A high to this signal indicates that the input latch has received the data byte.

INTR (Interrupt Request) This is an output signal which is set at the rising edge of the $\overline{\text{ACK}}$ signal. This signal can be used to interrupt the microprocessor. It requests for the next data byte for output. This signal is generated when $\overline{\text{OBF}}$, $\overline{\text{ACK}}$ and INTE (Interrupt Enable) are all high (logic 1). At the falling edge of the $\overline{WR}$ signal this is reset.

INTE (Interrupt Enable) This is an internal flip-flop to a flip-flop to a port and required to be set to generate the INTR signal. The two flip-flops $INTE_1$ and $INTE_2$ are set/reset using the BSR mode.

$\overline{\text{OBF}}$ (Output Buffer Full) This is an active low signal; it activates when the microprocessor writes data into output latch of 8255. This indicates to an output peripheral that a new data is ready to be read. It goes high again after the 8255 receives a $\overline{\text{ACK}}$ signal from peripheral.

If all the possible combinations are considered, then there will be four configurations of the 8255A in mode 2. These combinations with their corresponding control words are shown in figures 8.25 to 8.28. The status word in mode 2 is shown in figure 8.29, which will be placed in the accumulator if the port C is read.

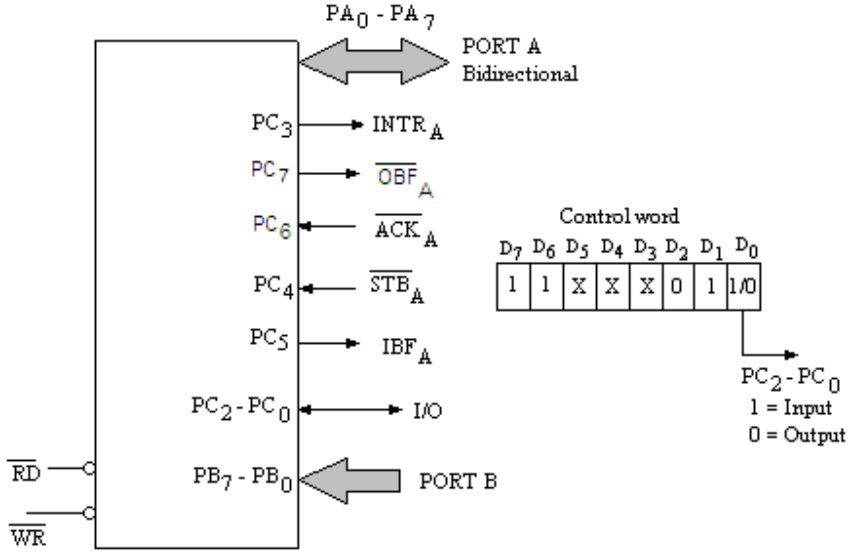


Fig. 8.25

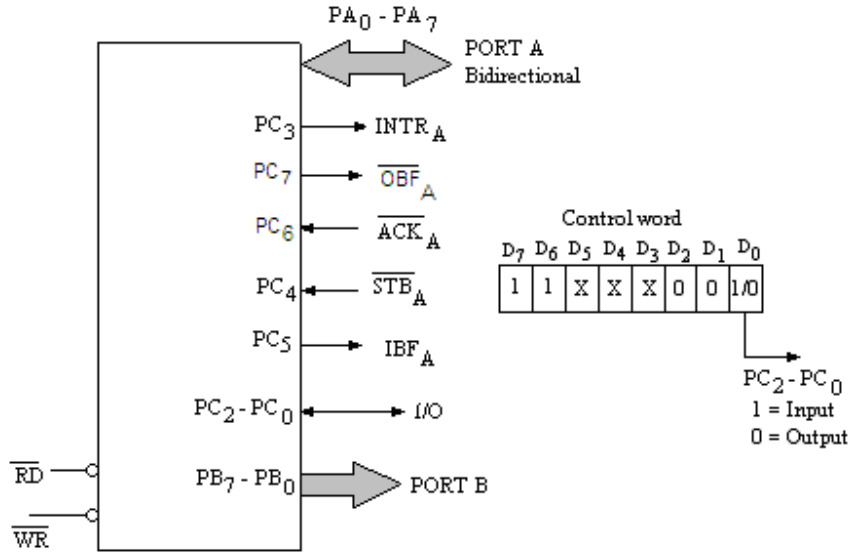


Fig. 8.26

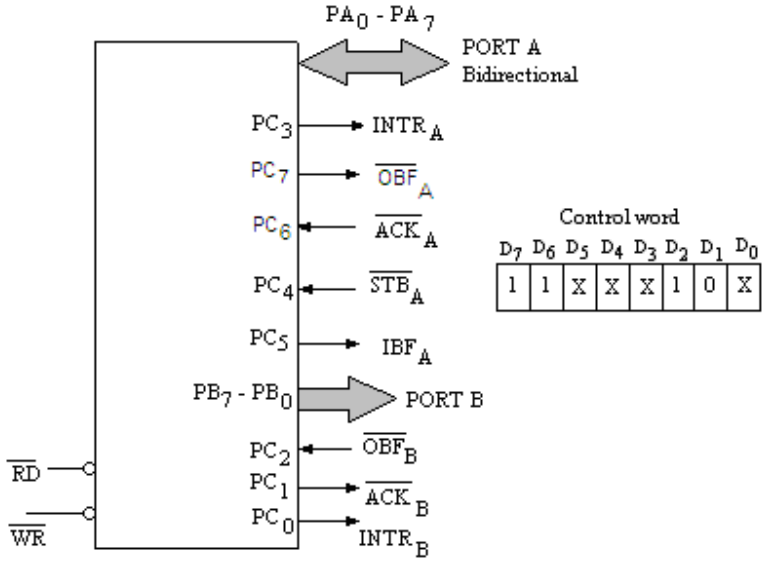


Fig. 8.27

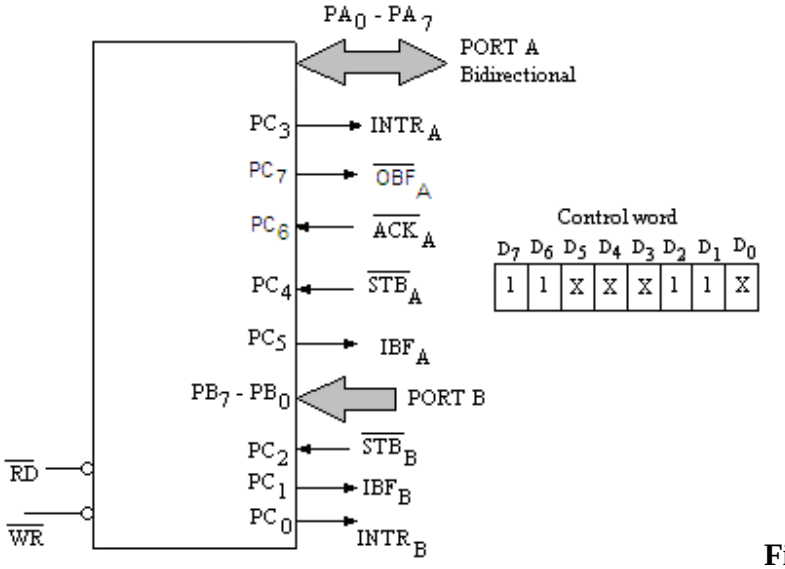


Fig. 8.28

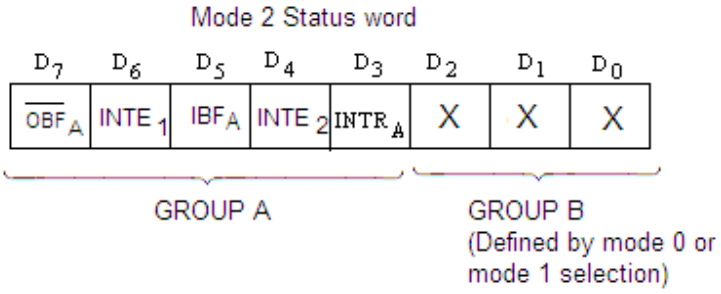


Fig. 8.29

Example 8.11. *Obtain the control word for the following configuration of the ports of 8255 A:*

Port A – as bidirectional,
Mode for Port A – mode 2
Port B – input port,
Mode for Port B – mode 0,
Port C_{Lower} – input port

Solution. . It may be remembered that the 8255 A is used in mode 2 only for port A. When port A is used to operate in mode 2; port B can be used either in mode 0 or mode 1. The pins of port C (PC_3 to PC_7) are used for the control signals for port A. For port B to operate in mode 0, Port C_{Lower} can be used as input or output pins as per the D_0 bit of the control word. On the other hand for the port B to operate in mode 1, PC_0, PC_1 and PC_2 bits will be used for control signals.

The control word for this case is given as (fig. 8.30):

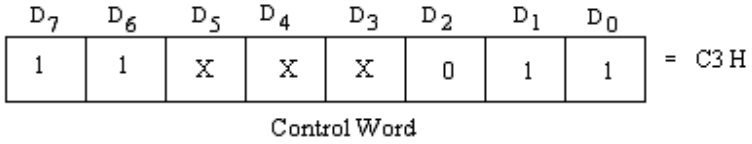


Fig. 8.30

Example 8.12. *Obtain the control word for the following configuration of the ports of 8255 A:*

Port A – as bidirectional,
Mode for Port A – mode 2
Port B – input port,
Mode for Port B – mode 1,

Solution.

The control word for this case is given as (fig. 8.31):

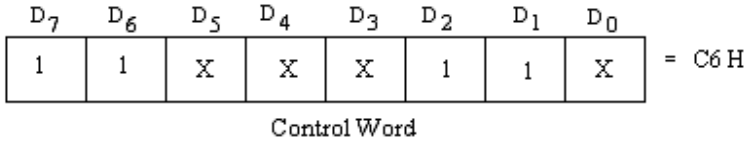


Fig. 8.31

8.7 BIT SET/RESET (BSR) MODE

As discussed in the earlier sections of this chapter that 8255A PPI can be used in bit set/reset (BSR) mode. To use the PPI in BSR mode D_7 bit of the control word should be set to zero. However, for other mode of operations this bit should be set to 1. In BSR mode eight bits of Port C can be set or reset by writing the control word in the control word register. If a control word with bit D_7 as 0 is recognized as a BSR mode then it does not alter any previously transmitted control word with D_7 as 1; thus I/O operations are not affected by a BSR control word. In this mode of operation individual bit of port C can be set or reset or in other words a BSR control word affects only one bit of port C. The control word format for this mode of operation is shown in figure 8.32.

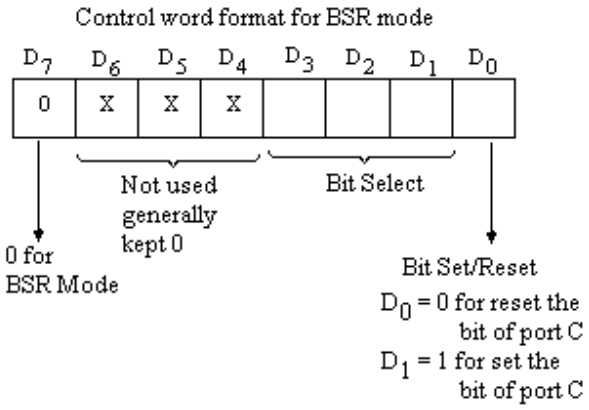


Fig. 8.32

The bits of the control word are defined as:

Bit D_7 This bit should be zero, to use 8255A in bit set/reset mode.

Bits D_4-D_6 These bits are don't care bits, which may be kept 0s or 1s but generally kept 0.

Bits D_1-D_3 The bits D_3, D_2 and D_1 are known as bit select pins. Table 8.5 shows which bit of port C will be affected by the D_3, D_2 and D_1 bits of bit select pins.

Table 8.5

D3	D2	D1	Bit of port C
0	0	0	PC_0
0	0	1	PC_1
0	1	0	PC_2
0	1	1	PC_3
1	0	0	PC_4
1	0	1	PC_5
1	1	0	PC_6
1	1	1	PC_7

Bit D_0 This is known as Set reset bit. To set a bit of port C, D_0 bit of the control word should be 1; and to reset a bit of port C, D_0 of the control word should be 0.

For example to reset PC_3 (bit 3 of port C), the control word format will be given as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	1	1	0	= 06 H

Bit 3 (PC_3)

Similarly, to set PC3, the control word format is given as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	1	1	1	= 07 H

Bit 3 (PC_3)

Example 8.13. *A relay circuit, to switch on or off an a.c. mains bulb, is connected to PC_4 (4^{th} bit of port C). The 8255A is connected microprocessor whose address for control*

word is 03 H. Using BSR mode it is desired to switch on and off the bulb after a regular interval of time. W rite a program in assembly language of 8085, to implement the above.

Solution. The relay circuit may be connected to PC_4 of 8255A as shown in figure 8.33. To switch on and off the bulb regularly using BSR, the program in assembly language of 8085 is written as:

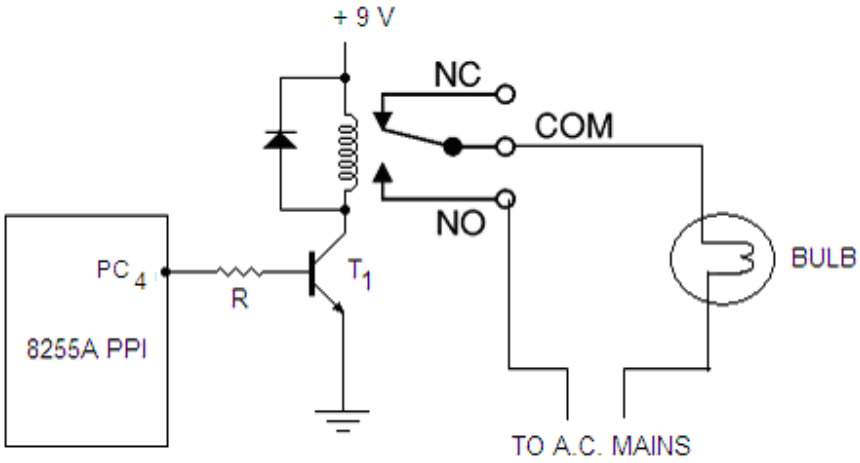


Fig. 8.33

The control word in BSR mode to reset PC_4 bit is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	1	0	0	0	= 08 H

The control word in BSR mode to set PC_4 bit is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	1	0	0	1	= 09 H

MAIN PROGRAM:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	; Initialize stack pointer.
START	MVI A,	08 H	; Load byte in accumulator to reset PC_4.
	OUT	03 H	; Reset PC_4 (the bulb will be off).
	MVI A,	08 H	; Load 08 H to accumulator.
	OUT	03 H	; Reset PC_4 (the bulb will be off).
	CALL	DELAY	; Jump to delay subroutine - to introduce a regular delay of constant time.
	MVI A,	09 H	; Load 09 to accumulator to set PC_4.
	OUT	03 H	; Set PC_4 (the bulb will be on).
	CALL	DELAY	; Jump to delay subroutine - to introduce a regular delay of constant time.
	JMP	START	; Jump to repeat the process.

SUBROUTINE PROGRAM:

Label	**Mnemonics**	**Operand**	**Comments**

```
DELAY   LXI  D,     XXXX H    ; Loads DE register pair with a 16-
                               bit number.
LOOP    DCX  D                ; Decrements DE register pair by 1.
        MOV  A,     E         ; Moves the contents of E register to
                               accumulator.
        ORA  D                ; ORing of the contents of D and E
                               registers are performed to set the
                               zero flag.
        JNZ         LOOP      ; If result is not zero than jump to
                               LOOP1.
        RET                   ; Go back to main program.
```

XXXX H may be taken any hexadecimal number for introducing any desired delay.

Example 8.14. *Write a program in assembly language of 8085 to send 01H, 02H, 03H . . . . . FFH data to port A of 8255A PPI with a time delay of 1msec to each output.*

Solution. The program to implement the above is given as:

```
Label     Mnemonics   Operand    Comments
          LXI  SP,    XXXX H     ; Initialize stack pointer.
          MVI  A,     80 H       ; Initialize 8255-I to work all the
                                   ports as output ports.
          OUT         03 H       ; Write the control word in the
                                   control word register of 8255-I.
          MVI  A,     00 H       ; Load 00 H to accumulator.
          MVI  B,     FF H       ; Load FF H to B-reg (last data).
LOOP      INR  A                 ; Increment acc.
          OUT         00 H       ; Send the number to port A.
          CALL        DELAY      ; Jump to delay subroutine - to
                                   introduce a delay of 1 msec.
          DCR  B                 ; Decrement B.
          JNZ         LOOP       ; If B is not zero jump to LOOP.
          HLT
```

SUBROUTINE PROGRAM:

```
Label     Mnemonics   Operand    Comments
DELAY     PUSH PSW               ; Push the program status word to
                                   stack.
          PUSH B                 ; Push the contents of B-C reg pair
                                   to stack.
          LXI  D,     XXXX H     ; Loads DE register pair with a 16-
                                   bit number to introduce a delay of
                                   1msec.
LOOP      DCX  D                 ; Decrements DE register pair by 1.
          MOV  A,     E          ; Moves the contents of E register to
                                   accumulator.
          ORA  D                 ; ORing of the contents of D and E
                                   registers are performed to set the
                                   zero flag.
```

	JNZ	LOOP	; If result is not zero than jump to LOOP1.
	POP B		; Pop the contents from stack to B-C reg pair back.
	POP PSW		; Pop program status word from stack.
	RET		; Go back to main program.

Example 8.15. *Write a program in assembly language of 8085 that inputs 256 bytes of data from port B of 8255A PPI with a time delay of 1msec to each input and store these bytes of data at memory locations starting at 2500 H.*

Solution. The program to implement the above is given as:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	; Initialize stack pointer.
	MVI A,	82 H	; Initialize 8255-I to work port B as input port and ports A and C as as output ports.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	LXI H,	2500 H	; Initialize H-L reg pair with first address of destination location.
	MVI C,	FF H	; Load FF H to C reg. as pointer.
LOOP	IN	01 H	; Input from port B.
	MOV M,	A	; Load the accumulator content to memory location.
	CALL	DELAY	; Jump to delay subroutine - to introduce a delay of 1 msec.
	INX H		; Increment H-L pair.
	DCR C		; Decrement C.
	JNZ	LOOP	; If C is not zero jump to LOOP.
	HLT		

SUBROUTINE PROGRAM:

The delay subroutine program is the same as given in the previous example.

Example 8.16. *Suppose 4096 bytes of data are stored in 2000 H to 2FFF H memory locations. Write a program in assembly language of 8085 that outputs these bytes of data to port A of 8255A PPI with a time delay of 1msec to each output.*

Solution. The program to implement the above is given as:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	; Initialize stack pointer.
	MVI A,	80 H	; Initialize 8255-I to work all the ports as output ports.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	LXI H,	19FF H	; Initialize H-L reg pair as pointer.

```
        INX  H                  ; Increment H-L reg. pair.
        MOV  A,     M           ; Load the data to accumulator.
LOOP    OUT         00 H        ; Output the data to port A.
        CALL        DELAY       ; Jump to delay subroutine - to
                                  introduce a delay of 1 msec.
        CPI         2F H        ; Compare accumulator contents to
                                  2F H.
        JNZ         LOOP        ; If not zero jump to LOOP.
        MOV  A,     L           ; Load L-reg data to accumulator.
        CPI         FF H        ; Compare with FF H.
        JNZ         LOOP        ; If not zero jump to LOOP.
        HLT
```

SUBROUTINE PROGRAM:

The delay subroutine program is the same as given in example 8.14.

Example 8.17. *A peripheral device is sending serial data to D_7 bit of port B of 8255A at an interval of 1msec. Write a program in assembly language of 8085 that converts 8 bit serial data stream to 8 bit parallel word, which is then sent to port A of 8255A.*

Solution. The program to implement the above is given as:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	; Initialize stack pointer.
	MVI A,	82 H	; Initialize 8255-I to work port B as input port and ports A and C as output ports.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	MVI B,	00 H	; Store 00 H in register B.
	MVI C,	07 H	; Store 07 H in register C as counter.
	IN	01 H	; Inputs a byte through port B.
	ORA	B	; ORing for taking 7^{th} bit.
	RAR		; Change the position.
	MOV B,	A	; Store to B register.
	CALL	DELAY	; Jump to delay subroutine - to introduce a delay of 1 msec.
	DCR C		; Decrement C.
	JNZ	LOOP	; If not zero jump to LOOP.
	OUT	00 H	; Outputs the data to port A.
	HLT		

SUBROUTINE PROGRAM:

The delay subroutine program is the same as given in example 8.14.

Example 8.18. *Suppose 256 bytes of data are stored in the memory locations 2000 H to 20FF H. Write a program in assembly language of 8085 that converts each of these bytes into serial data stream. The serial data stream is sent (bit by bit) through D_0 bit of port A of 8255A at the rate of approximately 1000 Bits/sec. (i.e. 1msec time interval between two bits).*

Solution. The program to implement the above is given as:

Label	Mnemonics	Operand	Comments
	LXI SP,	XXXX H	; Initialize stack pointer.
	MVI A,	80 H	; Initialize 8255-I to work all the ports as output ports.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	LXI H,	1FFF H	; Initialize the H-L register pair.
LOOP	INX H		; Increment the H-L register pair.
	MOV A,	M	; Store the byte in accumulator.
	MVI B,	00 H	; Store 00 H in register B.
AGAIN	OUT	00 H	; Outputs the bit through D_0 bit of port A.
	CALL	DELAY	; Jump to delay subroutine - to introduce a delay of 1 msec.
	RAR		; Rotate right for the next bit.
	DCR B		; Decrement B.
	JNZ	AGAIN	; If not zero jump to AGAIN.
	MOV A,	L	; Move the contents of L reg. to accumulator.
	CPI	FF H	; Compare with FF H
	JNZ	LOOP	; If not equal jump to LOOP.
	HLT		

SUBROUTINE PROGRAM:

The delay subroutine program is the same as given in example 8.14.

Example 8.19. *Write a program in assembly language of 8085 that inputs a byte from port A of 8255A and determines if the decimal equivalent of the byte is even or odd. If byte is even, then send ASCII E (45 H) to port B and send ASCII O (4F H) to port B if the byte is odd.*

Solution. The program to implement the above is given as:

Label	Mnemonics	Operand	Comments
	MVI A,	90 H	; Initialize 8255-I to work port A as input port and ports B and C as output ports.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	IN	00 H	; Inputs the byte through port A.
	ANI	01 H	; ANDed with 01 H to check D_0 bit of the Byte.
	JNZ	ODD	; If not zero (odd byte) jump to ODD.
	MVI A,	45 H	; Load accumulator for ASCII E.
	JMP	END	; Jump to END.
ODD	MVI A,	4F H	; Load accumulator for ASCII O.

END	OUT	01 H	; Output ASCII E or ASCII O to port B.
	HLT		

Example 8.20. *Write a program in assembly language of 8085 that inputs a byte from port A of 8255A and determines if the decimal equivalent of the byte is even or odd. If byte is even, then send ASCII E (45 H) to D_0 bit of port B in serial fashion; if odd send ASCII O (4F H) to D_0 bit of port B in serial fashion. The time interval between two consecutive bits should be 1msec.*

Solution. The program to implement the above is given as:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	90 H	; Initialize 8255-I to work port A as input port and ports B and C as output ports.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	IN	00 H	; Inputs the byte through port A.
	ANI	01 H	; ANDed with 01 H to check D_0 bit of the Byte.
	JNZ	ODD	; If not zero (odd byte) jump to ODD.
	MVI A,	45 H	; Load accumulator for ASCII E.
	JMP	END	; Jump to END.
ODD	MVI A,	4F H	; Load accumulator for ASCII O.
END	MVI C,	08 H	; Store 08 H to C register.
REPEAT	OUT	01 H	; Output ASCII E or ASCII O to D0 bit of port B.
	CALL	DELAY	; Call Delay subroutine to introduce a time delay of 1msec.
	RAR		; Rotate right.
	DCR C		; Decrement C.
	JNZ	REPEAT	; If not zero jump to REPEAT.
	HLT		

SUBROUTINE PROGRAM:

The delay subroutine program is the same as given in example 8.14.

PROBLEMS

1. Using the functional block diagram of 8255A PPI, explain its detail.
2. Mention various modes of operations of 8255A PPI; explain its working in simple I/O mode.
3. Draw the schematic block diagram of 8255A and explain the function of each block.
4. Mention various modes of operations of 8255A and explain it working in BSR mode. Explain also its control word format.

5. Draw and explain the control word format of 8255A. What will be control word if 8255 is used in simple I/O mode with port A as input port, port B as output port and port C_{Lower} as input port and Port C_{Upper} as output port?
6. Discuss the mode 1 output configuration of 8255. Discuss also its control word, control signals, timing diagram and status word.
7. Discuss the mode 1 Input configuration of 8255. Discuss also its control word, control signals, timing diagram and status word.
8. Explain how 8255 can be used in mode 2. Give the format of the control word when port A is used to operate in mode 2 and port B is operated in mode 1.
9. Give the format of the control word when port A is used to operate in mode 2 and port B is operated in mode 0. Port C_{Lower} is used as simple input or output.
10. Write an assembly language program of 8085A to output the contents of B-register to port A of 8255 A PPI.
11. Eight LEDs are connected to port B of 8255 A. write an assembly language program of 8085 to glow LEDs 0, 2, 4, 6 for 1 sec and then glow LEDs 1, 3, 5, 7 for the same time and repeat.
12. Obtain the control word when the ports of 8255A are to be used in mode 0 with port A as input port, port B as output port, and port C_{Lower} as output port and port C_{upper} as input port.

(Ans. 98 H)

13. Make a control word when the ports of 8255 A are to be used in mode 0 with all the ports as input ports.

(Ans. 9B H)

14. Obtain the control word for the following configuration of the ports of 8255 A for Mode 0 operation:
    Port A and port B as output ports and port C_{Lower} as output port; and port C_{upper} as input port.

(Ans. 88 H)

15. Obtain the control word for the following configuration of the ports of 8255 A for mode 1 operation:
    Port A – as input port,
    Mode for Port A – mode 1
    Port B – input port
    Mode for Port B – mode 1,
    The remaining pins of port C_{Upper} are to be used input pins.

(Ans. BE H or BF H)

15. Obtain the control word for the following configuration of the ports of 8255 A for mode 1 operation:
    Port A – output port in mode 1,
    Port B – output port in mode 1,
    The remaining pins PC_4 and PC_5 of port C are to be used output pins.

(Ans. A4 H or A5 H)

16. Obtain the control word for the following configuration of the ports of 8255 A:
    Port A – as bidirectional,
    Mode for Port A – mode 2
    Port B – input port,

Mode for Port B – mode 0,
Port C_{Lower} –output port.

(Ans. C2 H)

17. Write an assembly language program of 8085A to generate square wave of 100 Hz frequency using 8255A PPI. The wave should be available at PB_0 terminal of port B. Give the program of delay subroutine also.
18. Generate square wave of 100 Hz using BSR mode of 8255A. The wave should be available at PC4 terminal (D_4 bit of port C).
19. Write a program in 8085 assembly language to blink an LED at a regular interval of time. Assume LED is connected to PA_0 (D_0 bit of port A of 8255A PPI).
20. Write a program in 8085 assembly language to blink an LED at a regular interval of time using Bit Set/Reset (BSR) mode. Assume LED is connected to PC_0 (D_0 bit of port C of 8255A PPI).

9

Programmable Interval Timer/Counter: 8253

In many industrial applications precise time delays are needed. In discrete systems the time delays are generated using electronic timer based on logic gates and circuits or mechanical timers. However, in microprocessor based systems the time delays can be generated using software. The generation of accurate time delays by software control is very commonly used method as discussed in the earlier chapters of this book. In this method the processor has to wait for some time to generate a time delay, as microprocessor becomes unnecessarily busy in looping. The programmable interval timer/counter chip 8253 solves this problem. It is one of the most supporting chips for 8085 microprocessor. It is used for teal time applications. It can generate accurate time delays and wave forms using the software control. This chip includes three identical 16-bit pre-settable down counters that can operate independently, as each counter has a clock input, gate input and one output. The counters can count in binary or BCD. In this chapter a detailed discussion on this chip and its applications will be carried out.

9.1 INTEL PROGRAMMABLE INTERVAL TIMER 8253

Intel 8253 Programmable Interval Timer/Counter is an NMOS 24 pin plastic dual in line package (DIP). It operates at +5 volt d.c. source. It can generate accurate time delays and wave forms using the software control. This chip includes three identical 16-bit pre-settable down counters that can operate independently; namely counter 0, counter 1 and counter 2. The counters can count in binary or BCD. Each counter has a clock input, gate input and one output. The gate input enables the counting process. The counting may be started by applying the gate signal. The 8253 works at maximum clock frequency of 2 MHz. There is another programmable interval time which is upgraded version of 8253. It has pin compatible and software compatible with 8253. The 8254 can operate with high clock frequency up to about 8 MHz. This chip 8254 is compatible to 8086, 8088 and other microprocessors. However, 8253 is compatible to 8085microprocessor. We shall discuss the details of the chip 8253.

The 8253 can operate on the following six modes:

Mode 0: Interrupt on Terminal Count
Mode 1: Programmable one shot
Mode 2: Rate Generator
Mode 3: Square Wave Generator
Mode 4: Software Triggered Mode
Mode 5: Hardware Triggered Mode

The three counters of this chip can operate independently in any of the six modes.

9.2 BLOCK DIAGRAM OF 8253

Figures 9.1 and 9.2 show the pin diagram and functional block diagram of 8253. It has a data bus to be connected to the data bus of the microprocessor, a control word register, five control signals and three 16-bit presettable down counters namely:

Counter 0
Counter 1
Counter 2.

Control word Register:

The control word register of 8253 is accessed when A_0 and A_1 terminals are at logic 1. It is used to write a command word which specifies the counter to be used, its mode, and either Read or Write operation,

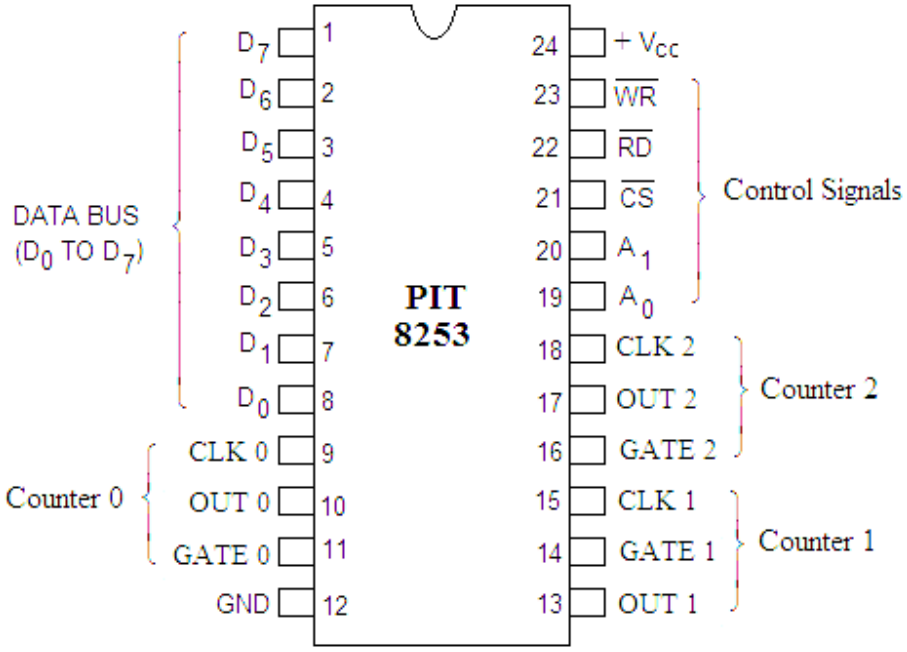


Fig. 9.1

The description of pins of 8253 Programmable Interval Timer (PIT) is given as follows:

Pin Nos. 1-8: form the bidirectional data bus buffer (D_7 to D_0) to be connected to the control bus of the microprocessor.

Pin Nos. 9-11: are provided for counter 0. Pin 9 is CLK 0 (Clock input terminal for counter 0), Pin 10 is OUT 0 (Output terminal for counter 0) and Pin 11 is GATE 0 (Gate input terminal for counter 0).

Pin No. 12: is the ground terminal.

Pin Nos. 13-15 are allotted to counter 1. Pin 13 is OUT 1 (Output terminal for counter 1), Pin 14 is GATE 1 (Gate input terminal for counter 1) and pin 15 is CLK1 (Clock input for counter 1).

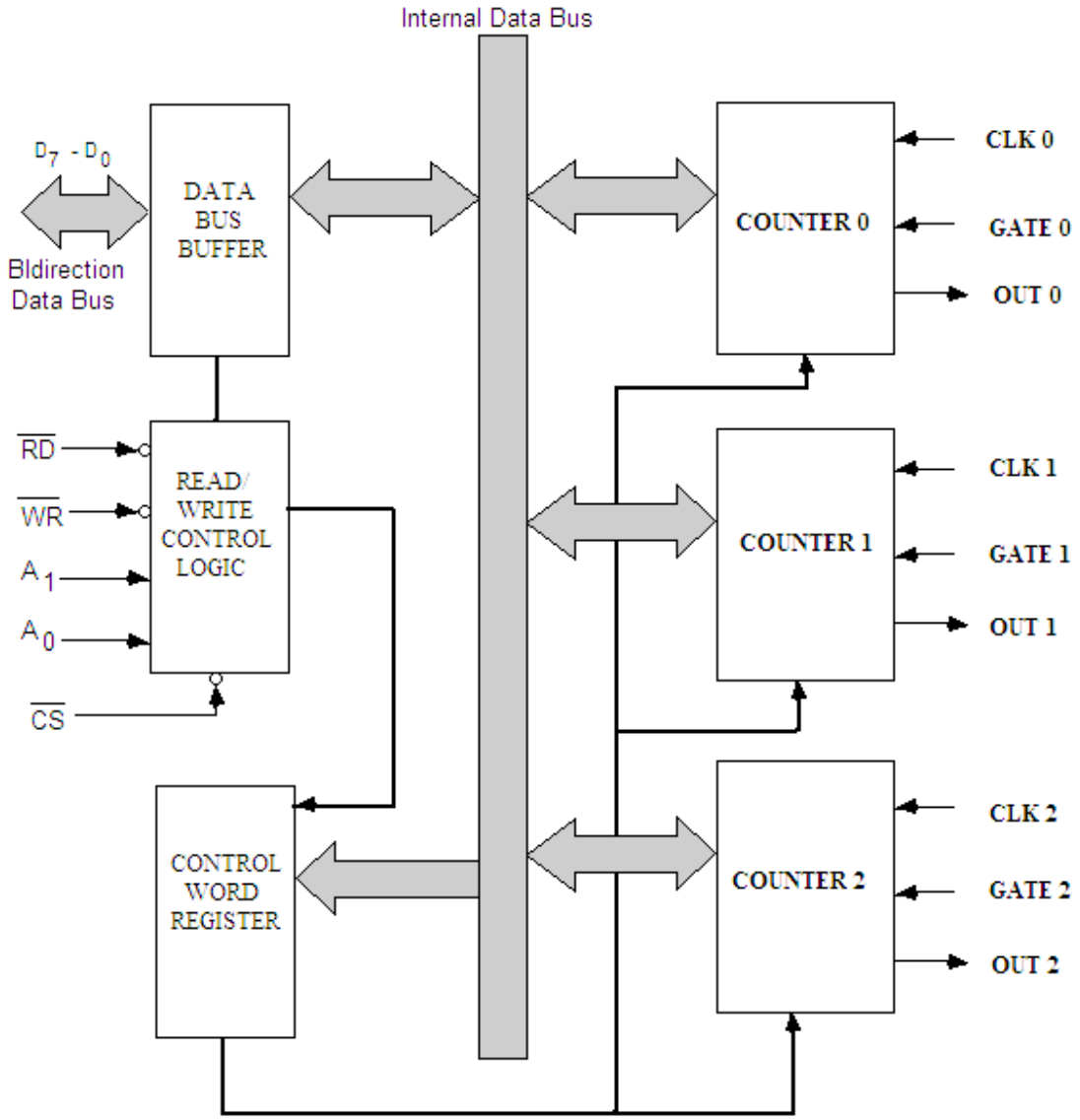


Fig. 9.2

Pin Nos. 16-18 are for counter 2. Pin 16 is GATE 2 (Gate input terminal for counter 2), Pin 17 is OUT 2 (Output terminal for counter 2) and pin 18 is CLK 2 (Clock input for counter 2).

Pin Nos. 19-20 are A_0 and A_1 terminals respectively. These terminals are used to select any one of the three counters and the control word register of 8253. These lines are to be connected to the address bus.
The counters and control word register are selected as given in table 9.1.

Table 9.1

A_0	A_1	Selection for Counters
0	0	Counter 0 is selected.
0	1	Counter 1 is selected.
1	0	Counter 2 is selected.
1	1	Counter 3 is selected.

Pin No. 21 is $\overline{CS}$ (Chip select terminal). It is an active low signal which allows enabling the device.

Pin No. 22 is $\overline{RD}$ (Read terminal). It is also an active low signal. A low to this pin informs the 8253; that the CPU is ready to receive the data in the form of count value.

Pin No. 23 is $\overline{WR}$ (Write terminal). When this pin is low, the microprocessor outputs data in the form of mode operation or loading of counter.

Pin No. 24 is + V_{CC} (+5 V) terminal.

9.3 LOGICS FOR COUNTERS

As already discussed, the 8253 has five control signals $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, A_1 and A_0. The table 9.2 shows the action of different combination of these control signals.

Table 9.2

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	Actions
0	1	0	0	0	Load counter 0
0	1	0	0	1	Load counter 1
0	1	0	1	0	Load counter 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read counter 0
0	0	1	0	1	Read counter 1
0	0	1	1	0	Read counter 2
0	0	1	1	1	No operation 3 state
0	1	1	X	X	No operation 3-state
1	X	X	X	X	Disable 3-state

Note: X indicates undefined state, i.e. it does not matter whether it is 0 or 1.

From this table it is clear that when $\overline{CS} = \overline{WR} = 0$ and $\overline{RD} = 1$ with different combination of A_1 and A_0; any counter can be selected and loaded with 16 counter values. Once the gate is enabled by applying a high signal (logic 1) to it, counter values starts decrementing at each of the trailing edge of the clock pulse. However, if $\overline{CS} = \overline{RD} = 0$ and $\overline{WR} = 1$, the counter read operation is performed by the different counters.

The clock pulse is to be applied to CLK terminal of each counter of 8253. Though the clock signal of 3 MHz frequency is available with 8085 microprocessor but it can not directly be applied to the CLK terminal of each counter. Since 8253 can work with a maximum of 2 MHz clock frequency. So the clock frequency available with 8085 microprocessor is divided by a factor of 2 using an edge triggered D-flip flop (7474) as shown in figure 9.3. So 1.5 MHz clock frequency available at the output of D-flip flop can be used as CLK input of the counters of 8253. In some microprocessor kits available in the market for use in laboratories, the CLK terminals of the counters are left open so that any clock signal of desired frequency may be connected to these terminals.

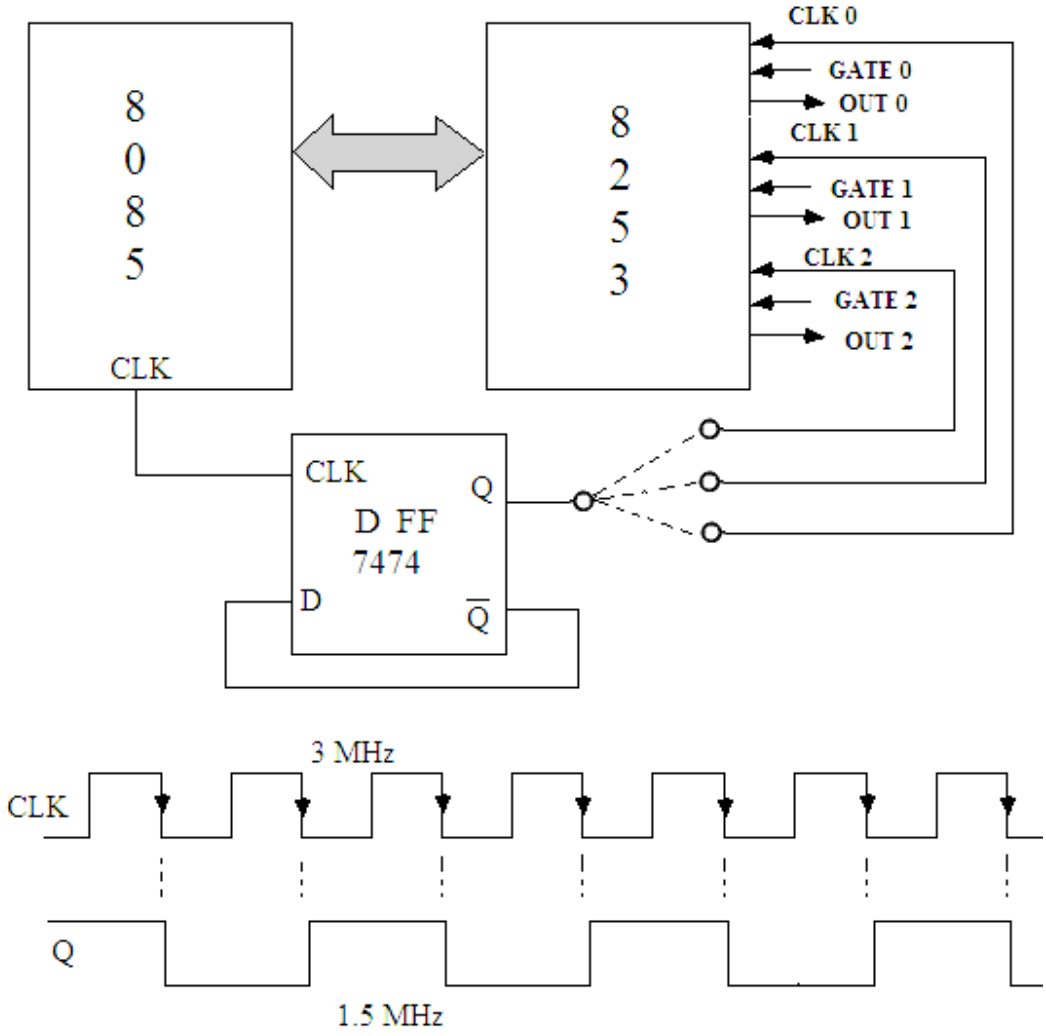


Fig. 9.3

9.4 CONTROL WORD FORMAT OF 8253

As already discussed, any of the three counters can be programmed to operate in any of the 6 modes. The control word decides the selection of counter, its mode of operation, loading, sequence of the count and selection of binary or BCD counting. The format for the control word of 8253 is shown in figure 9.4. The control word is written into the control word register of 8253, as per the requirement.

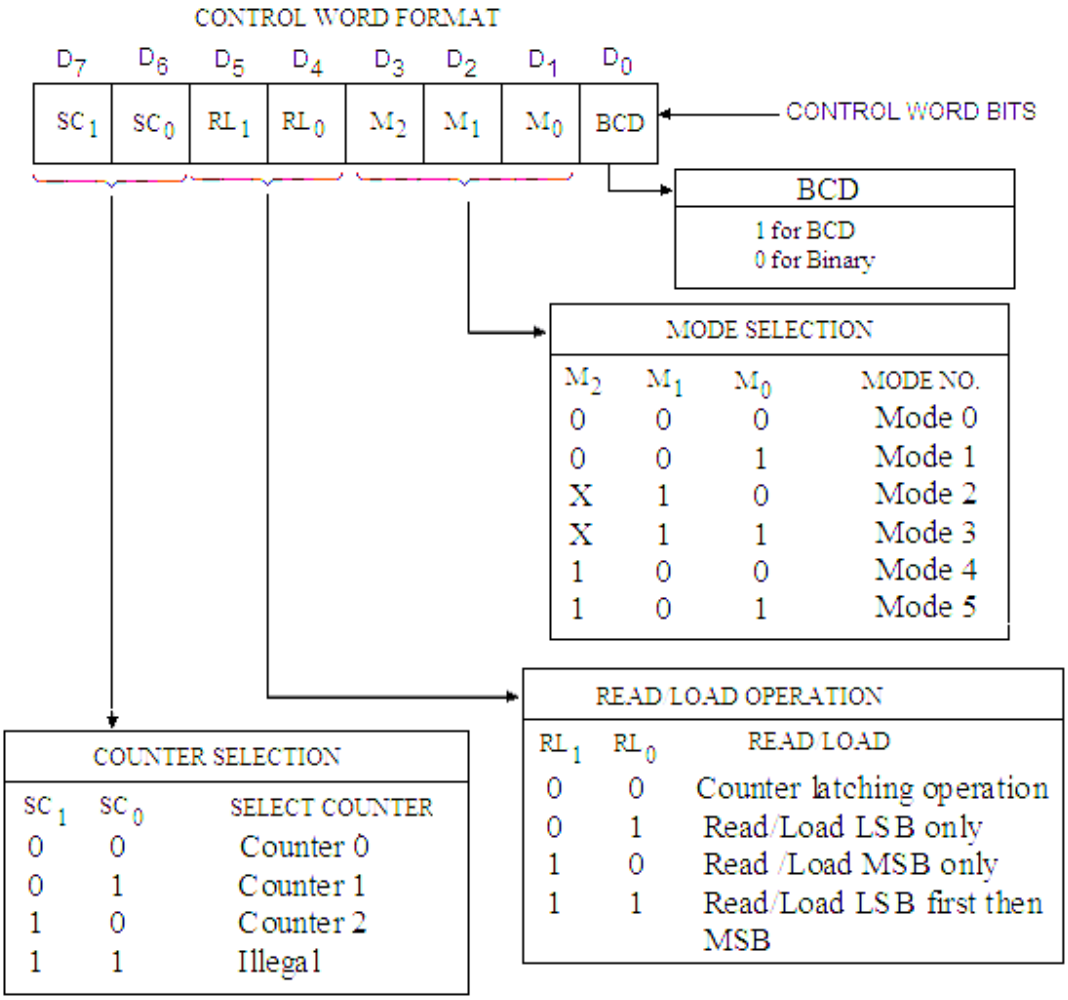


Fig. 9.4

The description of the bits of control word format is given as:

Bit D_0 Sets the counting of the counter in binary or BCD. To make the counting in BCD this bit is set to 1, and to make the counting in binary this bit is set to 0.

Bits D_1-D_3 These three bits are called mode selection bits M_0, M_1 and M_2 respectively; and are used for mode selection of the counters which are given in table 9.3.

Table 9.3

D_3 (M_2)	D_2 (M_1)	D_1 (M_0)	Mode
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

Bits D_4-D_5 These bits are known as Read/Load bits (RL_0 and RL_1) and are used for read/load of the count values in the counters. Their operations are given in table 9.4.

Table 9.4

D_5 (RL_1)	D_4 (RL_0)	Mode
0	0	Counter latching operation
0	1	Read/Load LSB only
1	0	Read /Load MSB only
1	1	Read/Load LSB first then MSB

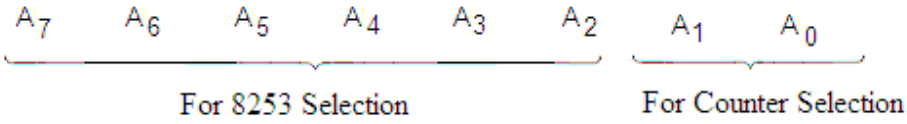
Bits D_6-D_7 These bits (SC_0 and SC_1) are used for counter selection as given in table 9.5.

Table 9.5

D_7 (SC_1)	D_6 (SC_0)	Select counter
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Illegal

9.5 INTERFACING AND PROGRAMMING OF 8253

Figure 9.5 shows interfacing of 8253 with the 8085 system bus. The eight data lines D_0 to D_7 of 8253 are connected to the data bus (Address data bus AD_0-AD_7) of the microprocessor. The chip is selected with $\overline{CS}$ terminal. So higher order address bus of the system is used for decoder circuit to select the chip. The A_8 and A_9 (two LSBs of the address bus) terminals of the microprocessor address bus are connected to A_0 and A_1 terminals of 8253 to select the proper counter. i.e.



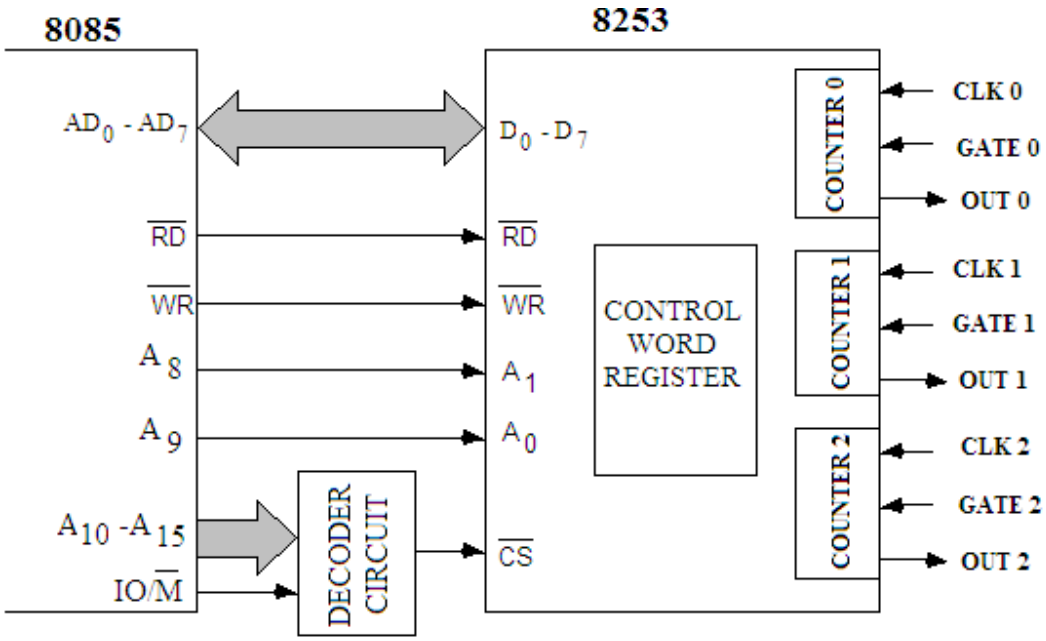


Fig. 9.5

Figure 9.6 shows the chip-select logic for 8253. From this logic diagram it is clear that if $A_2 – A_3$, $A_5 – A_7$ are all at logic 0 and A_3 is at logic 1, then it enables $\overline{CS}$ to select this chip. The counter selection will depend on the bits $A_1 – A_0$ as shown in table 9.6.

Table 9. 6

$\overline{CS}$									
A_7	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**	**HEX ADDRESS**	**Counter Selection**
0	0	0	1	0	0	0	0	10 H	Counter 0
0	0	0	1	0	0	0	1	11 H	Counter 1
0	0	0	1	0	0	1	0	12 H	Counter 2
0	0	0	1	0	0	1	1	13 H	Control word Register

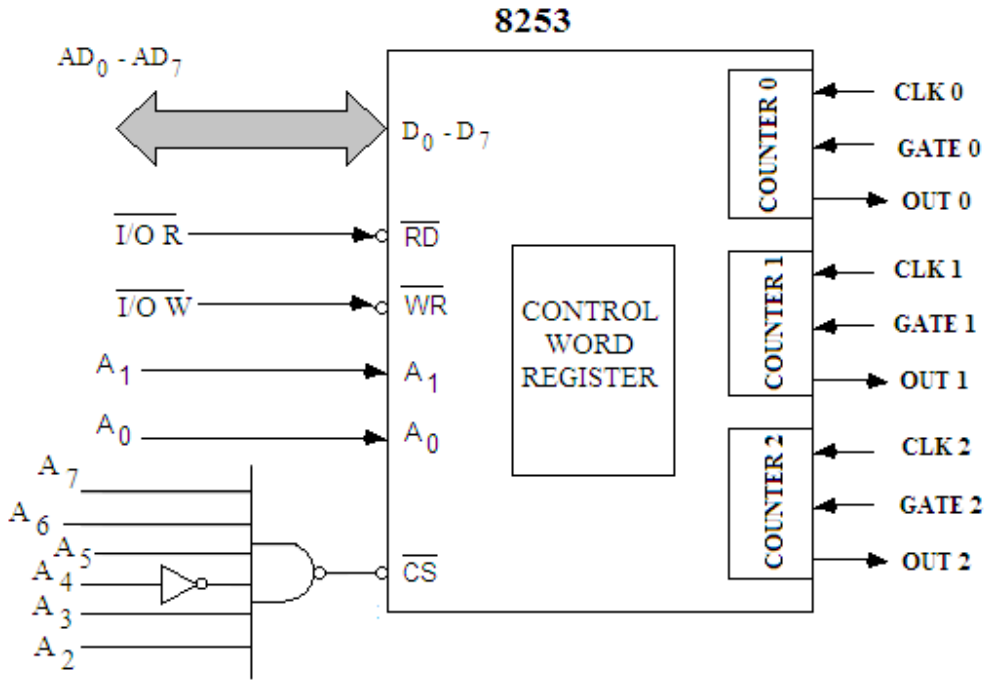


Fig. 9.6

As usual with IN and OUT instructions, the 8085 duplicates the port address on the address bus and data bus. The OUT instruction loads the control word in the control word register. As discussed above we may choose the hexadecimal address as

10 H	for counter 0
11 H	for counter 1
12 H	for counter 2
13 H	for control word register (CWR).

The OUT 13 H will load the accumulator content to the control word register. The accumulator is therefore, to be loaded to the proper control word (as decided by the control word format) before OUT 13 H instruction. This way 8253 is initialized. To load the count values to the counter number initialized by the OUT 13 instruction, again OUT instruction is used having the proper control word.

The IN instruction with proper port address will read the count values of the counter without stopping the counting. The 8253 can thus be programmed through write operations to generate various types of wave forms (which will be discussed in the succeeding section of this chapter) by loading the proper count values to any of the counters used in different modes.

So for programming the 8253 we proceed as given in the flow chart (figure 9.7).

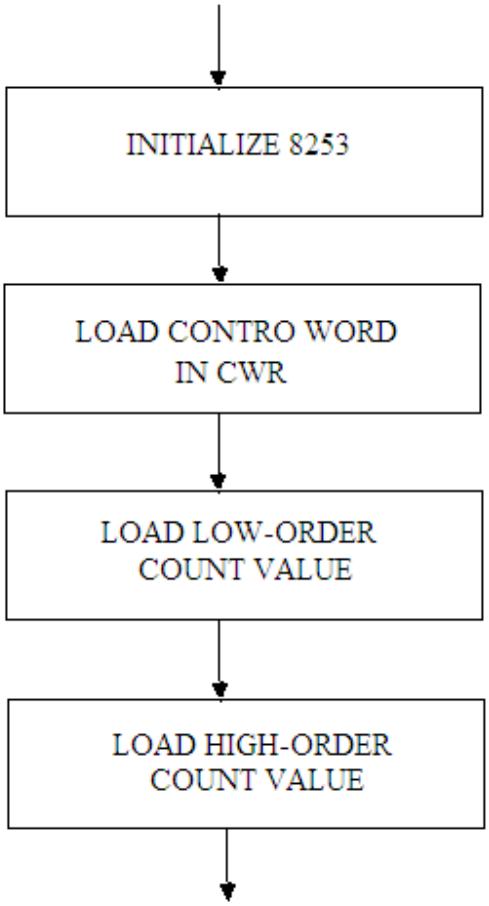


Fig. 9.7

For example to load counter 0 in mode 0 for the count value 1639 H in binary, we follow the program given below. The control word, to initialize the 8253 to load the given count value, first LS byte and then MS byte to the counter 0 in mode 0, is shown in figure 9.8.

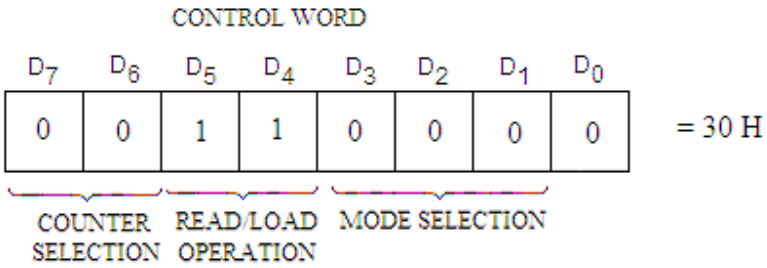


Fig. 9.8

Program:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	30 H	; Accumulator is loaded with the control word 30 H to Initialize 8253.

```
OUT        13 H      ; Write the control word in the
                       control word register of 8253.
MVI  A,    39 H      ; Load accumulator with LS byte
                       (39) of the count value.
OUT        10 H      ; First LS byte of the count value is
                       loaded to the counter 0 of 8253.
MVI  A,    16 H      ; Load accumulator with MS byte
                       (16) of the count value.
OUT        10 H      ; MS byte of the count value is then
                       loaded to the counter 0 of 8253.
. . . . . .
. . . . . .
. . . . . .
```

The count value of any counter of 8253 can also be read without stopping the counting. The bit pattern for the control word (figure 9.9) for latching operation is as follows:

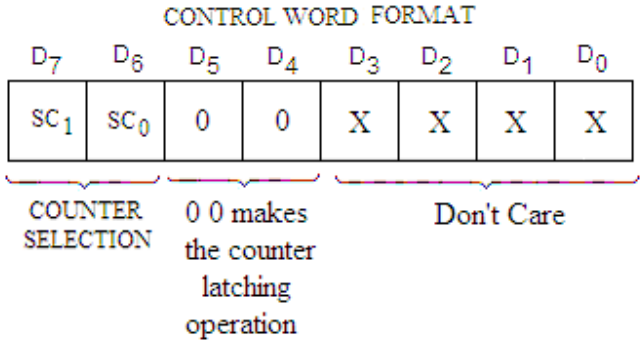


Fig. 9.9

The control word for latching operation for counter 0 is 00 H as shown in figure 9.10.

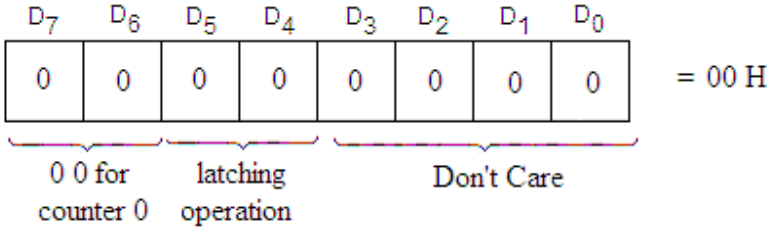


Fig. 9.10

So the following program can be used to read the count value of counter 0 while counting is in program.

Program:

```
Label      Mnemonics   Operand     Comments
           MVI  A,     00 H      ; Control word 00 H to read the
                                   count value of the counter 0 is
                                   loaded to accumulator.
           OUT         13 H      ; The control word is loaded in the
                                   control word register of 8253.
```

```
IN        10 H    ; First read the LS byte of the count
                    value.
MOV  B,   A       ; Load this value to B-register.
IN        10 H    ; Read the MS byte of the count
                    value.
MOV  C,   A       ; Load this value in C-register.
MVI  A,   16 H    ; Load accumulator with MS byte of
                    the count value.
. . . . . .
. . . . . .
```

Example 9.1. *Initialize 8253 to load counter 1 in mode 0 with 8bit binary number 06 H.*
Solution. To initialize 8253 we have the control word as (Figure 9.11):

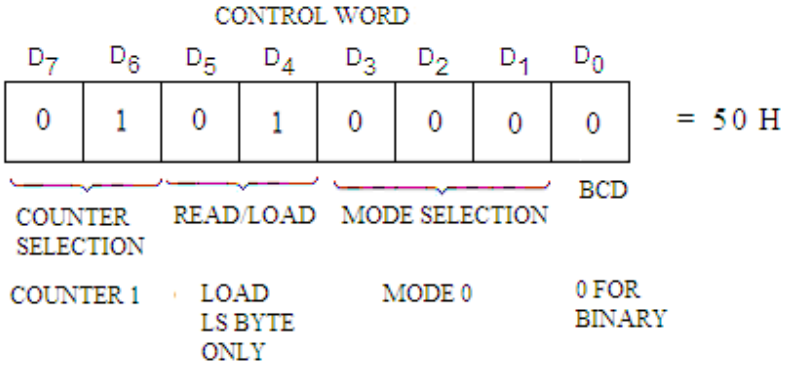


Fig. 9.11

The program for the same is written as:

Program:

Label	Mnemonics	Operand	Comments
	MVI A,	50 H	; Accumulator is loaded with the control word 50 H to Initialize 8253.
	OUT	13 H	; Write the control word in the control word register of 8253.
	MVI A,	06 H	; Load accumulator with LS byte (06 H) only of the count value.
	OUT	11 H	; 06 H of the count value is loaded to the counter 1 of 8253.
	HLT		; Stop processing.

Example 9.2. *Initialize 8253 to load counter 0 in mode 1 with 1632 H BCD number.*
Solution. To initialize 8253 we have the control word as (Figure 9.12):

The program for the same is written as:

Program:

Label	Mnemonics	Operand	Comments
	MVI A,	33 H	; Accumulator is loaded with the control word 33 H to Initialize 8253.
	OUT	13 H	; Write the control word in the control word register of 8253.

```
MVI  A,     32 H     ; Load accumulator first with LS byte (32 H) of the count value 1632 H.
OUT         10 H     ; 32 H of the count value is loaded to the counter 0 of 8253.
MVI  A,     16 H     ; Load accumulator then with MS byte (16 H) of the count value.
OUT         10 H     ; 16 H of the count value is loaded to the counter 0 of 8253.
HLT                  ; Stop processing.
```

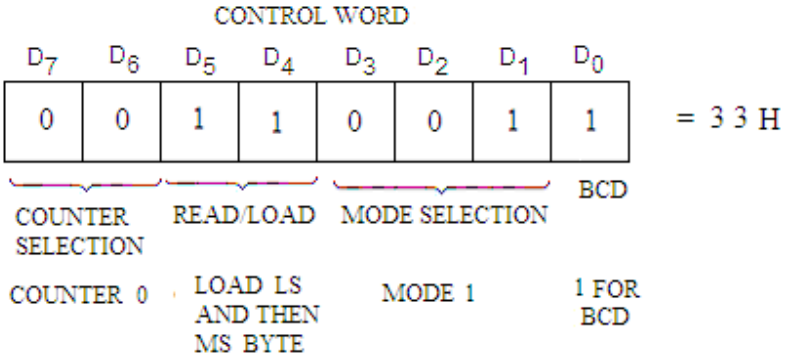


Fig. 9.12

Example 9.3. *Initialize 8253 to load counter 2 in mode 1 with a count value 5000_{10} in mode 0. Read the count value on the fly.*

Solution. To initialize 8253 we have the control word as (Figure 9.13):

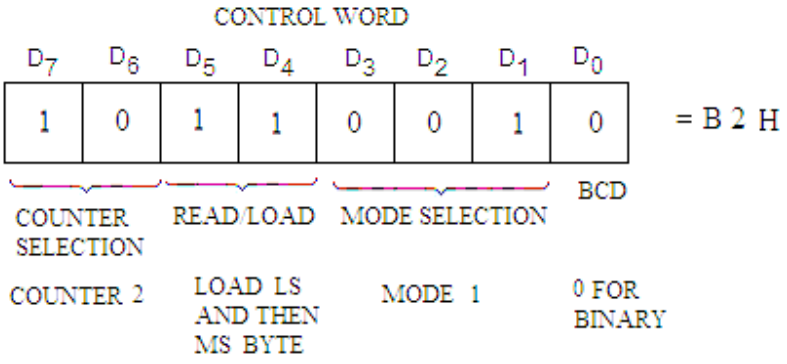


Fig. 9.13

The control word for latching operation for counter 2 is 80 H as shown in figure 9.14.

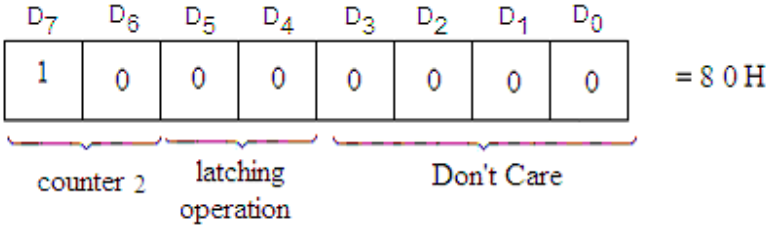


Fig. 9.14

The program for the same is written as:

The hexadecimal equivalent of 5000_{10} is 1388 H.

Program:

Label	Mnemonics	Operand	Comments
	MVI A,	B2 H	; Accumulator is loaded with the control word B2 H to Initialize 8253.
	OUT	13 H	; Write the control word in the control word register of 8253.
	MVI A,	88 H	; Load accumulator first with LS byte (88 H) of the count value 1632 H.
	OUT	12 H	; 88 H of the count value is loaded to the counter 2 of 8253.
	MVI A,	13 H	; Load accumulator then with MS byte (13 H) of the count value.
	OUT	12 H	; 16 H of the count value is loaded to the counter 2 of 8253.
	MVI D,	00 H	; Store 00 H to D-register.
LOOP	MVI A,	80 H	; Control word 80 H to read the count value of the counter 2 is loaded to accumulator.
	OUT	13 H	; The control word is loaded in the control word register of 8253.
	IN	12 H	; First read the LS byte of the count value.
	MOV B,	A	; Load this value to B-register.
	IN	12 H	; Read the MS byte of the count value.
	ORA B		; Logical OR the LS and MS byte to set zero flag.
	JNZ	LOOP	; If not zero jump to LOOP.
	HLT		; Stop processing.

9.6 PROGRAMMING OF 8253 IN MODE 0: INTERRUPT ON TERMINAL COUNT

The MODE 0 is interrupt on terminal count. The output of the desired counter goes low on setting the mode, and goes high after the loaded counts are complete. It is used for the generation of precise time delay under software control. After the selection of desired counter, a suitable count value is loaded in the counter. The GATE terminal of the selected counter is made high from low. The down counting of the counter will be started. At the beginning of the counter, OUT terminal becomes low and remains low till the counting is stopped at the zero count value. On reaching the terminal count (i.e. count value becomes zero), the OUT terminal becomes high (logic 1) and remains high until and unless the selected counter is reloaded or a new count value is loaded or the mode of operation is changed. Further, if the count value is reloaded in count register while the counting is in progress, then after LS byte of the count value is written, the current counting is stopped; and after MS byte of the count value is written, the counting restarts

with the new count value. The timing diagram for mode 0 operation is shown in figure 9.15.

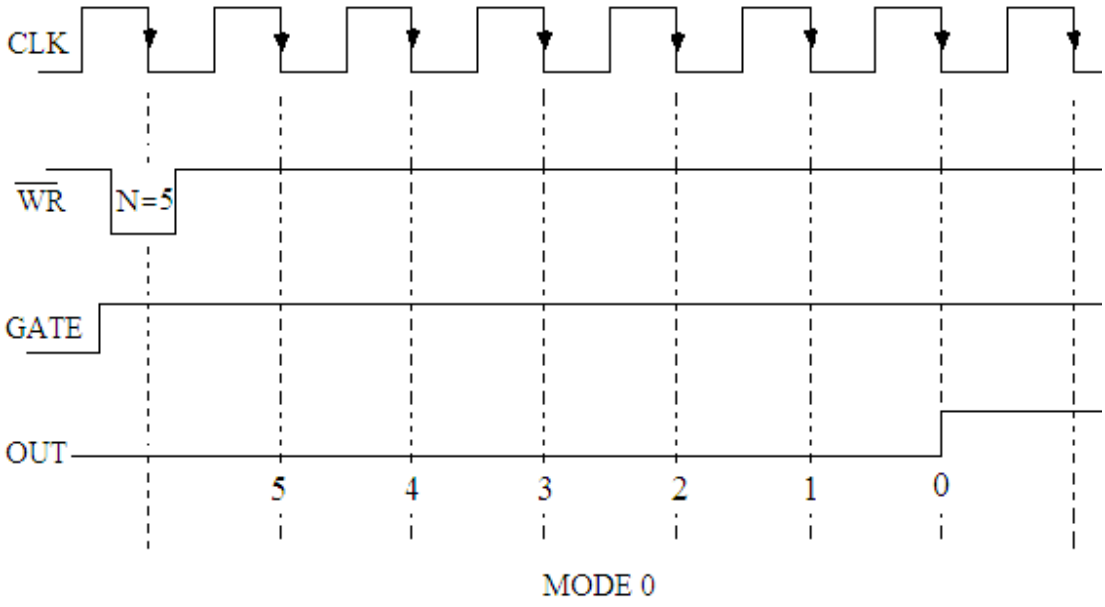


Fig. 9.15

If during the counting operation, GATE terminal is made low from high, the counting will be suspended. If the GATE terminal is again made high the counting may be resumed at the trailing edge of the clock pulse from the value where the counting was suspended. This is depicted in figure 9.16. The OUT terminal can be used to interrupt the microprocessor.

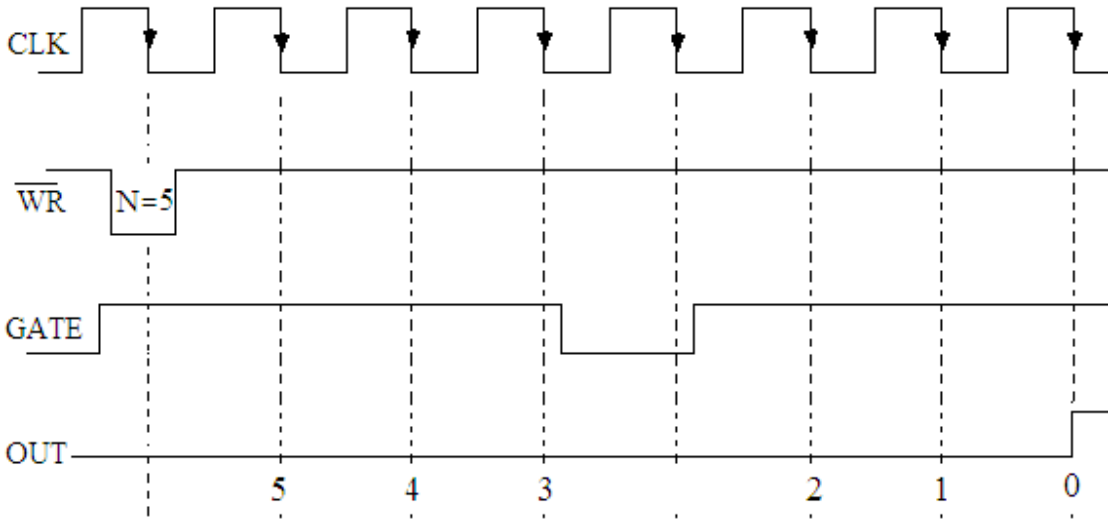


Fig. 9.16

9.7 PROGRAMMING OF 8253 IN MODE 1: PROGRAMMABLE ONE SHOT

The MODE 1 is Programmable or retriggerable one shot. In this mode the OUT terminal of the counter goes low on the triggering of corresponding GATE input (low to high transition); which (OUT terminal) goes high on terminal count. The OUT terminal remains low for the count value loaded to the counter.

Initially control word and count values are loaded to the device register. A leading edge trigger signal is applied to the GATE input of the selected counter. At the next trailing edge of the clock the OUT terminal goes low. The down counting of the counter will now start and at the terminal count (i.e. count value becomes zero) the OUT terminal becomes high. In other word we can say that the OUT terminal remains low for the number of counts loaded in the selected counter. In this mode the counter thus be used as a triggered mono (or one) shot of desired width. This width can be varied by the count value loaded for the selected counter. It is illustrated in figure 9.17. The changing signal from high to low to GATE terminal does not affect the signal of OUT terminal, once the counting is started.

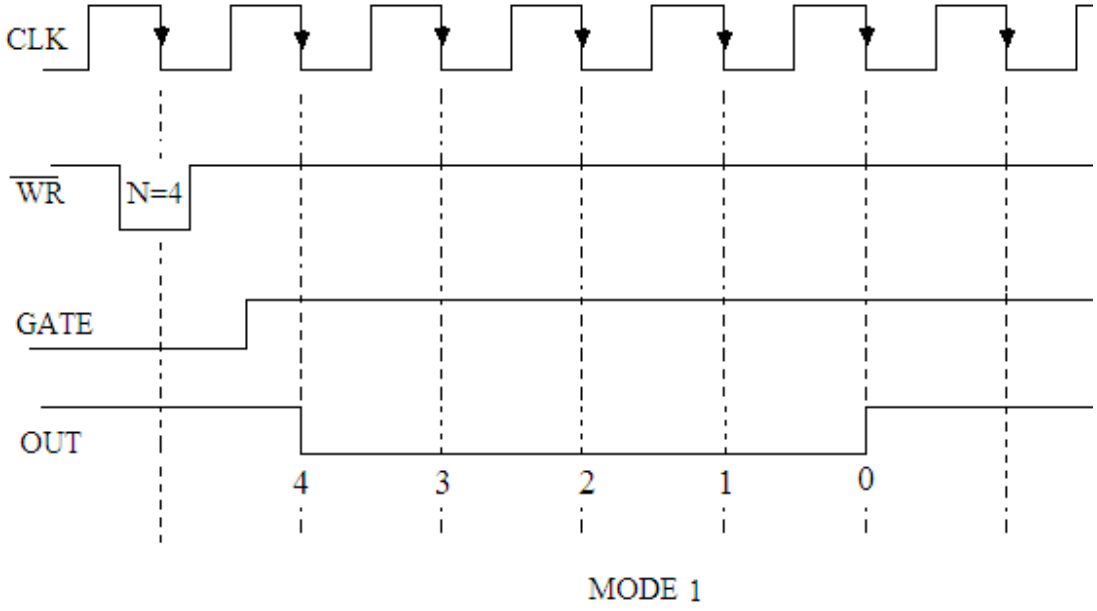


Fig. 9.17

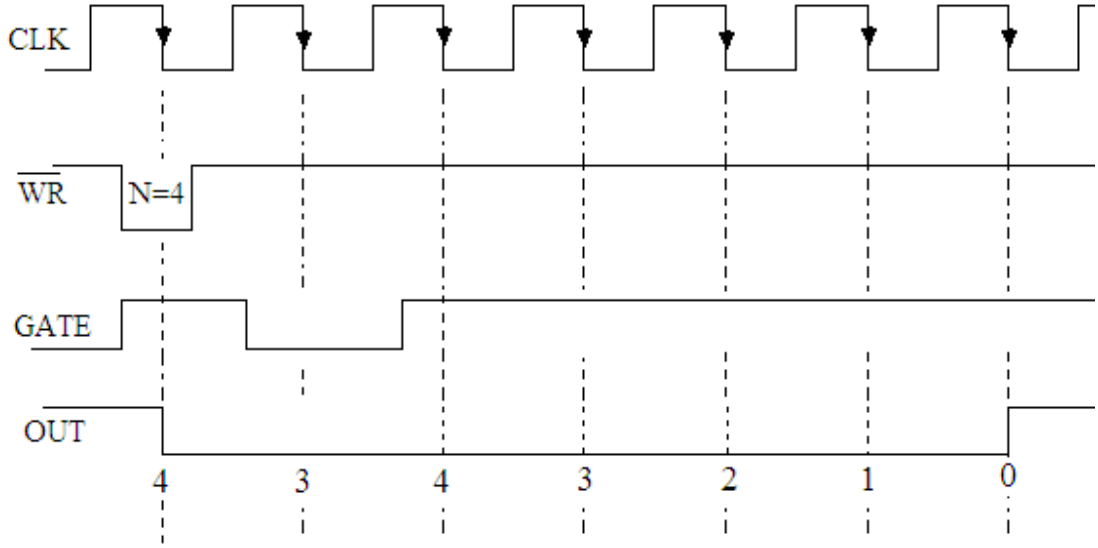


Fig. 9.18

If the signal at the GATE terminal is changed from low to high when the counting is in progress, then the counting will be stopped for exactly one clock pulse. After one

clock pulse, the recounting will be started from the beginning i.e. the counter will be reloaded with the same count value and the down counting will be started again and the OUT terminal goes low for the same count value. Thus it is also called retriggerable one shot. This situation is shown in figure 9.18.

A new count may be loaded to the counter while OUT terminal is low; it will not affect the width of the low OUT signal until the next trigger is applied to the GATE.

9.8 PROGRAMMING OF 8253 IN MODE 2: RATE GENERATOR

Any counter of 8253 may be programmed to operate in mode 2 as rate generator which is also called as divide by N counter.

Initially control word for mode 2 and count value N are loaded into the selected counter. The down counting of the counter will be started as soon as a high signal to the GATE terminal of the counter is applied. The OUT terminal stays high for (N-1) clock pulses and becomes low for exactly one clock pulse. After this OUT signal goes high again for (N-1) clock pulses and low for one pulse as the count value N is reloaded into the selected counter. This process continues as long as the GATE terminal is high. It, therefore, works as divide by N counter. Further, if the counter is working in this mode and a new count value is loaded, the present period will not be affected but the subsequent period will be as per the new count value. The timing diagram is shown in figure 9.19.

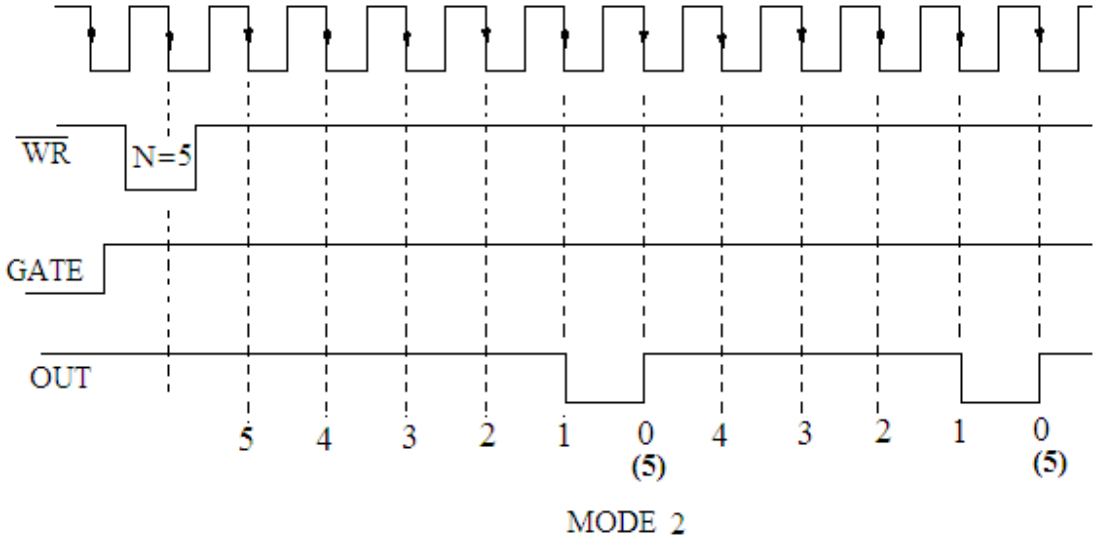


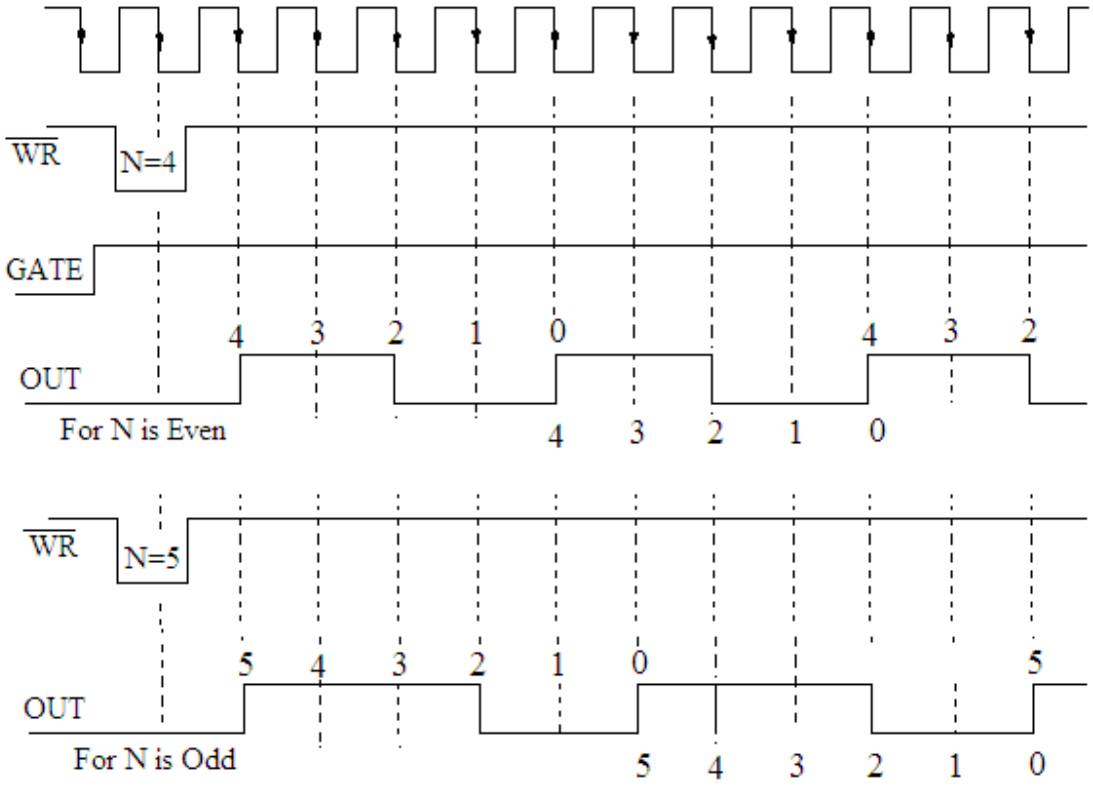
Fig. 9.19

The GATE input when low, will force the OUT terminal to high. When the GATE input goes high, the counter will start from the initial count. Thus the GATE can be used to synchronize the counter.

9.9 PROGRAMMING OF 8253 IN MODE 3: SQUARE WAVE GENERATOR

When 8253 operate in mode 3, the counter acts as a square wave generator. This is similar to mode 2 with the difference that the OUT terminal of the selected counter in mode 3 remains high for half the count value and is low for the other half, if the count value is an even number. However, if the count value is an odd value, OUT terminal remains high for $\left(\frac{N+1}{2}\right)$ clock pulses and is low for $\left(\frac{N-1}{2}\right)$ clock pulses.

When control word for mode 3 and count value N are loaded into the selected counter, a high signal at the GATE terminal starts the down counting. The OUT terminal remains high for $\left(\frac{N}{2}\right)$ counts and is low for the next $\left(\frac{N}{2}\right)$ counts, if the count value N is even; and for odd value of N, the OUT signal is high for $\left(\frac{N+1}{2}\right)$ pulses and is low for $\left(\frac{N-1}{2}\right)$ pulses as mentioned above. The count value is reloaded with full count and the process is repeated till the GATE is high. Figure 9.20 shows the timing diagram for this mode of operation for both even and odd count value.



MODE 3

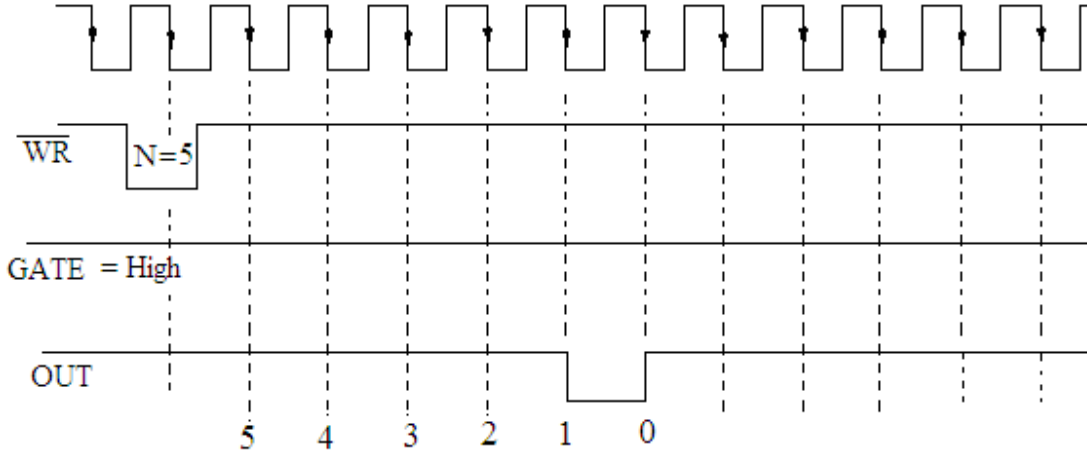
Fig. 9.20

The mode of operation of 8253 generates a square wave of period specified by the count value loaded to the counter. Any change in count value becomes effective, after the present counts are over.

9.10 PROGRAMMING OF 8253 IN MODE 4: SOFTWARE TRIGGERED STROBE

In this mode of operation software controlled delayed negative pulse of one clock period duration is generated with and without synchronization. After the mode of operation of 8253 is set, the OUT terminal will be high, and the down counting of the counter will start as soon as the count value is loaded into the counter. On terminal count,

the OUT terminal goes low for one clock pulse then will go high again as shown in figure 9.21.



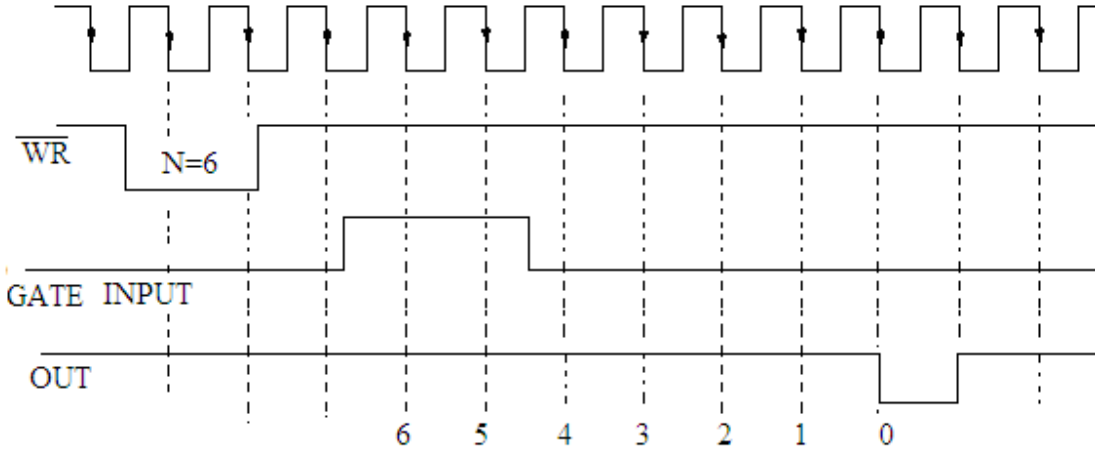
MODE 4

Fig.9.21

If the count register is reloaded between the output pulses, the present period will not be affected, but the subsequent period will reflect the new count value. The counter will be inhibited when the GATE terminal goes low. The GATE input can, therefore, be used for synchronization. The generation of strobe signal at OUT terminal is triggered by loading the count value in the counter that is why this mode is called software triggered mode.

9.11 PROGRAMMING OF 8253 IN MODE 5: HARDWARE TRIGGERED STROBE

In this mode of operation a delayed negative pulse of one clock period duration is generated if a positive going trigger input is applied at the GATE terminal. After the mode of operation of 8253 is set, and the count value is loaded into the counter OUT



MODE 5

Fig. 9.22

terminal will be initially high. As the rising edge signal is applied to GATE terminal, the down counting of the counter will be started at the first trailing edge of the clock pulse after the rising edge of the GATE input. On terminal count, the OUT terminal goes low for one clock pulse then will go high again. The timing diagram for this mode of operation is shown in figure 9.22. As the low to high transition of the GATE terminal causes triggering hence this mode is referred to as hardware triggered strobe.

If the GATE input is made low to high again the counter is reloaded by the same count value. The down counting of the count value once again started and on the terminal count the OUT terminal goes low for one clock period.

Example 9.4. *An LED is connected to PA_0 of 8255 PPI. It should show ON and OFF regularly with a delay of one second. The delay should be introduced with 8253 used in mode 0 of counter 1. Assume the clock frequency applied to the counter of 8253 is 2 MHz.*

Solution. The control word to initialize 8253 is (Figure 9.23):

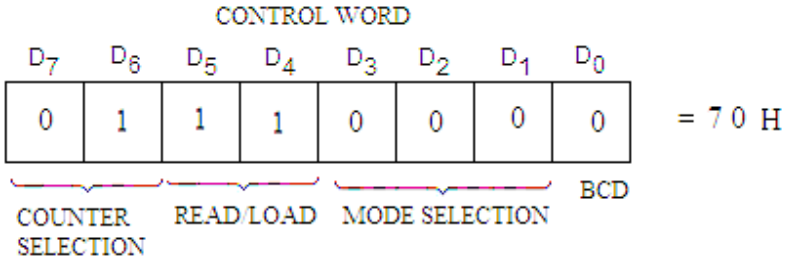


Fig. 9.23

The control word for latching operation for counter 2 is 80 H as shown in figure 9.24.

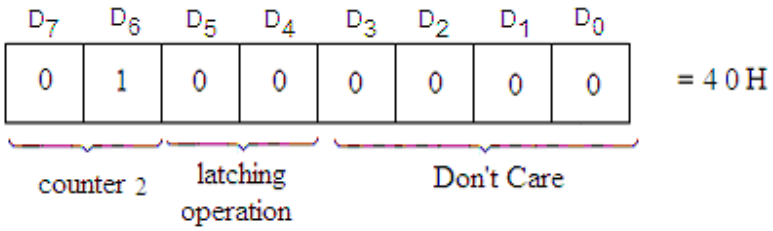


Fig. 9.24

Since the frequency of the clock signal connected to CLK terminal is 2 MHz, so the time of one clock pulse is 0.5 μ sec. If a count value of 50000_{10} (C350 H) is loaded to the counter, then a delay of 25 msec may be introduced when the counter is used in mode 0. Further to introduce a total time delay of 1 sec, then the subroutine program used for 25 msec may be executed 40 (28 H) times. Thus the program having the subroutine is given below:

Main Program:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	; Initialize stack pointer.
	MVI A,	80 H	; Initialize 8255 PPI with all the ports as output ports.
	OUT	03 H	; The control word is loaded to the control word register of 8255.
	MVI B,	00 H	; Load the register B with 00 H.

```
        MOV  A,    B        ; 00 H is stored in accumulator.
LOOP1   OUT        00 H     ; 00 is send to port A of 8255 so that
                            LED is OFF.
        MVI  C,    28 H     ; C register is used as counter so that
                            the delay subroutine is executed 40
                            times.
LOOP    CALL       DELAY    ; Delay program for the delay of 25
                            msec is called.
        DCR C               ; Decrement C.
        JNZ        LOOP     ; If the content in C becomes zero,
                            jump to LOOP.
        MOV  A,    B        ; Move the content of B in
                            accumulator.
        CMA                 ; Complement the accumulator
                            contents.
        MOV  B,    A        ; Complemented content of
                            accumulator is stored in B register.
        JMP        LOOP1    ; Jump to LOOP1 to change the state
                            of LED.
```

Subroutine Program:

```
DELAY   MVI  A,    70 H     ; Accumulator is loaded with the
                            control word 70 H to Initialize
                            counter 1 of  8253.
        OUT        13 H     ; Write the control word in the
                            control word register of 8253.
        MVI  A,    50 H     ; Load accumulator first with LS
                            byte (50 H) of the count value
                            C350 H.
        OUT        11 H     ; 50 H of the count value is loaded
                            to the counter 1 of 8253.
        MVI  A,    C3 H     ; Load accumulator then with MS
                            byte (C3 H) of the count value.
        OUT        11 H     ; C3 H of the count value is loaded
                            to the counter 1 of 8253.
AGAIN   MVI  A,    40 H     ; Control word 40 H is loaded to
                            accumulator to read on fly the
                            count value of the counter 1.
        OUT        13 H     ; The control word is loaded in the
                            control word register of 8253.
        IN         11 H     ; First read the LS byte of the count
                            value.
        MOV  D,    A        ; Load this value to D-register.
        IN         11 H     ; Read the MS byte of the count
                            value.
        ORA  D              ; Logical OR the LS and MS byte to
                            set zero flag.
```

```
        JNZ     AGAIN       ; If not zero jump to AGAIN.
        RET                 ; Return to main program.
```

Example 9.5. *The OUT terminal of counter 0 of 8253 is connected to RST 7.5, which is used to interrupt the microprocessor to clear the memory locations 2100 H to 21FF H. Use counter 0 of 8253 in mode 0 to enable RST 7.5 after a delay of 32.5 msec. Assume the clock frequency applied to the counter 0 of 8253 is 2 MHz.*

Solution. The control word to initialize counter 0 of 8253 is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	1	0	0	0	0	= 30 H

The accumulator contents for SIM to enable RST 7.5, RST 6.5 and RST 5.5 are given by:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SOD	SOE	X	R7.5	MSE	M7.5	M6.5	M5.5	
0	0	0	0	1	0	0	0	= 08 H

The clock frequency used for counter 0 is 2 MHz, so time for one clock pulse is 0.5 μsec. To introduce a delay of 32.5 *msec*, count value of 65000 (65000 x 0.5 μsec = 32.5 *msec*) is to be loaded to the counter 0. The hexadecimal equivalent of 65000_{10} is FDE8 H.

Main Program:

```
Label   Mnemonics   Operand     Comments
        EI                      ; Enable interrupts.
        MVI  A,     08 H        ; Bit pattern is loaded to
                                  accumulator to enable interrupts.
        SIM                     ; Enable interrupts.
        MVI  A,     30 H        ; Accumulator is loaded with the
                                  control word 30 H to Initialize
                                  counter 0 of 8253 in mode 0.
        OUT         13 H        ; Write the control word in the
                                  control word register of 8253.
        MVI  A,     E8 H        ; Load accumulator first with LS
                                  byte (E8 H) of the count value
                                  FDE8 H.
        OUT         10 H        ; E8 H of the count value is loaded
                                  to the counter 0 of 8253.
        MVI  A,     FD H        ; Load accumulator then with MS
                                  byte (FD H) of the count value.
        OUT         10 H        ; FD H of the count value is loaded
                                  to the counter 0 of 8253.
HERE    JMP         HERE        ; Wait for interrupt.
```

As soon as the counting is over, the OUT terminal of counter 0 becomes high which enables RST 7.5. The RST 7.5 interrupts the microprocessor and the program jumps to its vector location 003C H. At this vector location say it is stored JMP FFBD H. So the monitor will transfer the program from FFBD H. Now the user transfers the program from FFBD H to a memory location where service subroutine program for RST 7.5 is

stored. The program in service subroutine will be stored to clear the memory location 2100 H to 21FF H.

Interrupt Service Subroutine:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI H,	2100 H	; H-L pair is loaded with 2100 H, starting address of the memory location.
	MVI C,	FF H	; Load FF H to C-register which is used as counter.
	XRA A		; Clear the accumulator.
LOOP	MOV M,	A	; Clear the memory location.
	INX H		; Increment the H-L register pair.
	DCR C		; Decrement the content of C-register for next byte.
	JNZ	LOOP	; If the contents of C-register are not zero then jump to LOOP for next byte.
	EI		; Enable interrupts.
	RET		; Return.

Example 9.6. *A positive going pulse is applied (figure 9.25) to the GATE terminal of counter 1 of 8253 used in mode 0. The RST 7.5 is used to interrupt the microprocessor. Write a program to measure the width of this pulse which is of approximately 1 second. The clock signal applied to CLK terminal of counter 1 is 10 KHz. The width should be stored in terms of counts in memory locations 2100 H and 2101 H.*

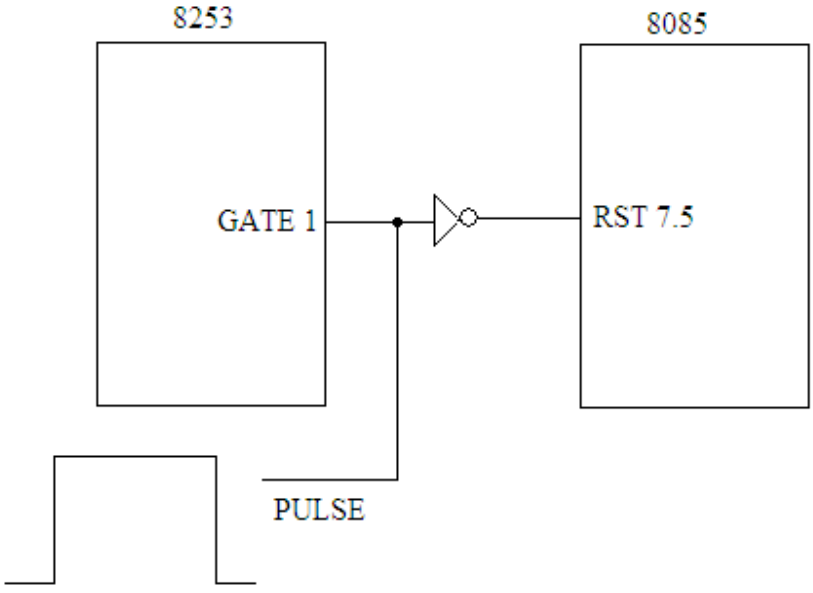


Fig. 9.25

Solution. The control word to initialize counter 1 of 8253 is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	1	1	0	0	0	0	= 70 H

The accumulator contents for SIM to enable RST 7.5, RST 6.5 and RST 5.5 are given by:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SOD	SOE	X	R 7.5	MSE	M 7.5	M 6.5	M 5.5	
0	0	0	0	1	0	0	0	= 08 H

The clock frequency used for counter 1 is 10 KMHz, so time for one clock pulse is 0.1 *msec*. This will also be measurement accuracy. Let us initialize the counter 1 with the count value corresponding to 2 second (1 sec for high and 1 sec for low). The count value will be 20000_{10}, (as 20000x0.1 msec = 2 sec). The hexadecimal equivalent of 20000_{10} is 4E20 H. So 4E20 H will be loaded as count value in counter 1.

To measure the pulse width the counter 1 is initialized in mode 0 in which signal (High pulse) on GATE terminal enables the counter; and RST 7. 5 is disabled. As soon as the pulse ends, the counting will be stopped and RST 7.5 will interrupt the microprocessor. The interrupt service subroutine will store the required number of counts in 2100 H and 2101 H memory locations. The required number of counts will be obtained by subtracting the remaining counts from the initial counts.

The program is therefore given by:

Main Program:

Label	**Mnemonics**	**Operand**	**Comments**
	EI		; Enable interrupts.
	MVI A,	08 H	; Bit pattern is loaded to accumulator to enable interrupts.
	SIM		; Enable interrupts.
	MVI A,	70 H	; Accumulator is loaded with the control word 70 H to Initialize counter 1 of 8253 in mode 0.
	OUT	13 H	; Write the control word in the control word register of 8253.
	MVI A,	20 H	; Load accumulator first with LS byte (20 H) of the count value 4E20 H.
	OUT	11 H	; 20 H of the count value is loaded to the counter 1 of 8253.
	MVI A,	4E H	; Load accumulator then with MS byte (4E H) of the count value.
	OUT	11 H	; 24E H of the count value is loaded to the counter 1 of 8253.
HERE	JMP	HERE	; Wait for interrupt.

Interrupt Service Subroutine:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI H,	2100 H	; H-L pair is loaded with 2100 H, starting address of the memory location.
	IN	11 H	; Read the LS byte of the remaining counts.

```
MOV  D,     A        ; Store it to D-register.
IN          11 H     ; Read the MS byte of the remaining
                       counts.
MOV  E,     A        ; Store it to E-register. So the
                       remaining counts are stored in D-E
                       register pair.
MVI  A,     20 H     ; Initial maximum LS byte (20 H) is
                       stored in accumulator.
SUB  D               ; Subtract the remaining LS byte
                       from maximum LS byte.
MOV  M,     A        ; Store the counts in 2100 H.
INX  H               ; Increment the H-L register pair.
MVI A,      4E H     ; Initial maximum MS byte is stored
                       in accumulator.
SBB  E               ; Subtract with borrow the
                       remaining MS byte from the
                       maximum MS byte.
MOV  M,     A        ; Store the counts in 2101 H.

EI                   ; Enable interrupts.
RET                  ; Return.
```

Example 9.7. *The D_0 bit of 8255 A PPI is connected trigger input (GATE terminal) of counter 2 of 8253 as shown in figure 9.26. The counter 2 is used in mode 1. The count to be loaded to the counter 2 is 0A H. Write a program so that a wave, having low for 10 pulse duration and repeats, is generated. The CRO may be connected to the OUT terminal of counter 2 to see the wave shape.*

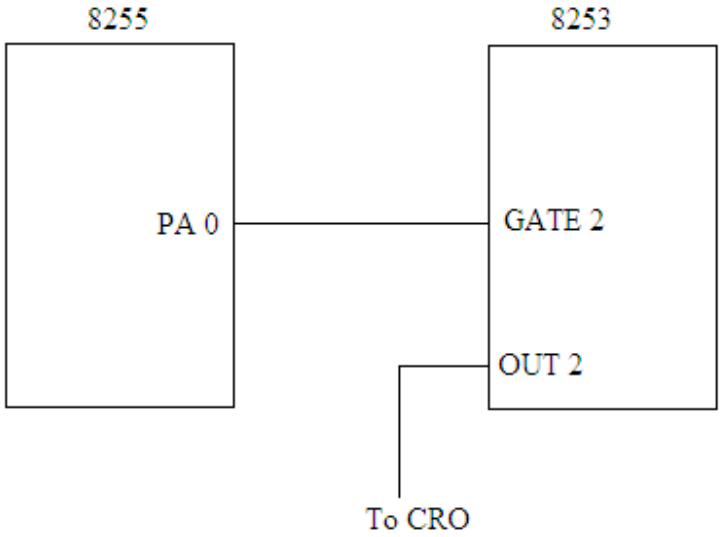


Fig. 9.26

Solution. The control word to initialize counter 0 of 8253 is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	1	0	0	1	0	= A2 H

Main Program:

Label	Mnemonics	Operand	Comments
	MVI A,	30 H	; Accumulator is loaded with the control word 30 H to Initialize counter 2 of 8253 in mode 1.
	OUT	13 H	; Write the control word in the control word register of 8253.
	MVI A,	0A H	; Load accumulator 0A H as count value.
	OUT	12 H	; 0A H of the count value is loaded to the counter 2 of 8253.
	MVI A,	80 H	; Initialize 8255 A PPI with all the ports to use as output ports.
	OUT	03 H	; 80 H is loaded to control word register of 8255.
	MVI A,	00 H	; Store 00 H to accumulator.
	OUT	00 H	; PA0 is zero.
	MVI A,	01	; Store 01 H to accumulator to make PA0 as 1.
	OUT	00 H	; PA0 is 1.
	CALL	DELAY	; Call a delay program to introduce a small delay.
	JMP	LOOP	; Jump to LOOP for repetition.

DELAY SUBROUTINE:

Label	Mnemonics	Operand	Comments
DELAY	MVI C,	FF H	; Store FF H to C register to use it as counter.
LOOP	DCR C		; Decrement counter.
	JNZ	LOOP	; If count is not zero jump to LOOP.
	RET		; Return to main program.

Example 9.8. *Write a program that will count the approximate number of T-states in a program given below. The SOD line of 8085 is connected to the GATE terminal of counter 1 of 8253. The 8253 is used in mode 1. The crystal of 8085 is of 2 MHZ which is directly connected to CLK input of the counter. The number of T-states used is to be stored in memory locations 2200 H and 2201 H.*

Label	***Mnemonics***	***Operand***
	LHLD	*2101 H*
	XCHG	
	LHLD	*2103 H*
	MVI C,	*00 H*
	DAD D	
	JNC	*NXT*
	INR C	
NXT	*SHLD*	*2105 H*

```
MOV A,      C
STA         2107 H
HLT
```

Solution. The control word to initialize counter 1 of 8253 in mode 1 is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	1	1	0	0	1	0	= 72 H

Let us load FFFF H as count value in counter 2.

Main Program:

Label	Mnemonics	Operand	Comments
	MVI A,	72 H	; Accumulator is loaded with the control word 72 H to Initialize counter 1 of 8253 in mode 1.
	OUT	13 H	; Write the control word in the control word register of 8253.
	MVI A,	FF H	; Load accumulator with low byte of count value.
	OUT	11 H	; Low byte FF H of the count value is loaded to the counter 1 of 8253.
	MVI A,	FF H	; Load accumulator with High byte of count value.
	OUT	11 H	; High byte FF H of the count value is loaded to the counter 1 of 8253.
	CALL	MODPRG	; Call the modified program.
	LXI H,	2200 H	; H-L pair is loaded with 2200 H, starting address of the memory location.
	IN	11 H	; Read the LS byte of the remaining counts.
	MOV D,	A	; Store it to D-register.
	IN	11 H	; Read the MS byte of the remaining counts.
	MOV E,	A	; Store it to E-register. So the remaining counts are stored in D-E register pair.
	MVI A,	FF H	; Initial maximum LS byte (FF H) is stored in accumulator.
	SUB D		; Subtract the remaining LS byte from maximum LS byte.
	MOV M,	A	; Store the counts in 2100 H.
	INX H		; Increment the H-L register pair.
	MVI A,	FF H	; Initial maximum MS byte is stored in accumulator.
	SBB E		; Subtract with borrow the remaining MS byte from the maximum MS byte.

```
        MOV  M,     A          ; Store the counts in 2101 H.
        EI                     ; Enable interrupts.
        HLT                    ; Stop processing.
```

Label	Mnemonics	Operand	Comments
MODPRG	MVI A,	C0 H	; Control word for enabling SOE and SOD.
	SIM		; Set SOD line.
	IN	11 H	; Read the LS byte of the remaining counts.
	MOV D,	A	; Store it to D-register.
	LHLD	2101 H	;
	XCHG		;
	LHLD	2103 H	;
	MVI C,	00 H	; Given program
	DAD D		;
	JNC	NXT	;
	INR C		;
NXT	SHLD	2105 H	;
	MOV A,	C	;
	STA	2107 H	;
	MVI A,	40 H	; Control word for enabling SOE and resetting SOD.
	SIM		; Reset SOD line.
	RET		; Return to main program.

Example 9.9. *Write a program to generate a negative pulse of approximately one T state after every 4 msec using 8253. Consider 1 MHz clock signal is applied to CLK terminal of the counter 1 of 8253.*

Solution. A negative pulse of approximately one T state is to be generated. This is the problem of rate generator, so 8253 may be used in mode 2. The counter 1 is to be initialized for this purpose. The GATE terminal is kept high. Further, clock frequency is 1 MHz so the time for one T state is 1 μsec.

The number of counts to be loaded for 2 msec delay is given by:

$$= \frac{4m\sec}{1\mu\sec} = 4000_{10}.$$

The number 4000 may be counted in BCD.

The control word to initialize counter 1 of 8253 in mode 2 is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	1	1	0	1	0	1	= 7A H

The program is, therefore, written as:

Main Program:

Label	Mnemonics	Operand	Comments

	MVI A,	7A H	; Accumulator is loaded with the control word 7A H to Initialize counter 1 of 8253 in mode 2.
	OUT	13 H	; Write the control word in the control word register of 8253.
	MVI A,	00 H	; Load accumulator with low byte of count value.
	OUT	11 H	; Low byte 00 H of the count value is loaded to the counter 1 of 8253.
	MVI A,	40 H	; Load accumulator with High byte of count value.
	OUT	11 H	; High byte 40 H of the count value is loaded to the counter 1 of 8253.
	HLT		; Stop processing.

Example 9.10. *Use counter 1 of 8253 in mode 2 as divide-by-8 counter.*

Solution. The GATE terminal is kept high.

The control word to initialize counter 1 of 8253 in mode 2 is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	0	1	0	1	0	0	= 54 H

The program is written as:

Main Program:

Label	Mnemonics	Operand	Comments
	MVI A,	54 H	; Accumulator is loaded with the control word 54 H to Initialize counter 1 of 8253 in mode 2.
	OUT	13 H	; Write the control word in the control word register of 8253.
	MVI A,	08 H	; Load accumulator with count value 08 H.
	OUT	11 H	; Count Value is loaded to the counter 1 of 8253.
	HLT		; Stop processing.

Example 9.11. *Write a program to generate a square wave of 3 KHz frequency using counter 1 of 8253. Consider 1.5 MHz clock signal is applied to CLK terminal of the counter.*

Solution. To generate square wave, we use the 8253 in mode 3. The count value to be loaded to the counter 1 is given by:

$$= \frac{1.5MHz}{3KHz} = 500_{10}.$$

So load 0500 H in BCD to counter 1. The GATE terminal of counter 1 kept high.

The control word to initialize counter 1 of 8253 in mode 2 is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	1	1	0	1	1	1	= 77 H

The program is, therefore, written as:

Main Program:

Label	Mnemonics	Operand	Comments
	MVI A,	77 H	; Accumulator is loaded with the control word 77 H to Initialize counter 1 of 8253 in mode 3.
	OUT	13 H	; Write the control word in the control word register of 8253.
	MVI A,	00 H	; Load accumulator with low byte of count value.
	OUT	11 H	; Low byte 00 H of the count value is loaded to the counter 1 of 8253.
	MVI A,	50 H	; Load accumulator with High byte of count value.
	OUT	11 H	; High byte 50 H of the count value is loaded to the counter 1 of 8253.
	HLT		; Stop processing.

PROBLEMS

1. Draw the schematic block diagram of programmable interval timer (PIT) 8253 and explain its detail..
2. Discuss the control word format for 8253.
3. Describe Read/Write control logic for 8253.
4. Explain the different modes of 8253.
5. Discuss different applications of 8253.
6. Discuss how 8253 can be used as rate generator.
7. Discuss how 8253 can be used as programmable mono shot.
8. Discuss how 8253 can be used as square wave generator.
9. Discuss how 8253 can be used as software triggered strobe.
10. Discuss how 8253 can be used as hardware triggered strobe.
11. Explain how to generate a delay using 8253 in mode 0.
12. Explain how the count value loaded to a counter of 8253 is read while the counting is in progress.
13. Write a program to generate a square wave of 1 KHz using 8253 in mode 3. The CLK terminal of the counter used is connected to a clock frequency of 1 MHz.
14. Assume that CLK terminal of the counter 0 of 8253 is connected to clock frequency of 1 KHz. It is desired to generate square wave of 1 Hz frequency using this counter. Write a program to implement this.
15. Set up 8253 as a square wave generator of 1 msec time period, if the input frequency connected to CLK terminal of counter 0 is 1 MHz.
16. Write a program to generate a negative pulse of approximately one T-state after every 1 msec. Consider 2 MHz clock frequency is applied to the CLK terminal of the counter 1.
17. Connect PC_0 of 8255 A PPI to the trigger input (GATE 0) of counter 0 of 8253 used in mode 1. Load the count 0024 H (binary). Write a program to generate a wave having low for 36 pulses and repeat.

18. The OUT terminal of counter 0 of 8253 is connected to RST 7.5. It is used to interrupt the microprocessor to set SOD line of 8085 after a delay of 40 msec. Use counter 0 in mode 0 and clock frequency connected to the CLK terminal is 2 MHz.
19. An LED is connected to SOD line of 8085. The should glow ON/OFF regularly with an interval of 1 sec. The delay should be introduced by counter 2 of 8253 used in mode 0. Consider clock frequency connected to the CLK terminal is 10 KHz.
20 Use 8253 in mode 3 as a square wave generator. Use count value as 10 H (BCD). The counter 1 is to be used for this purpose.
21. Use counter 0 of 8253 to generate a square wave of 10 KHz and simultaneously counter 1 as divide by N counter. (N = 10 H BCD). The clock frequency connected to the CLK terminals is 1 MHz.
22. Use counter 0 of 8253 as a simple counter in mode 0. After certain delay, read the counts when the counting is in progress.
23. The 8253 is interfaced with 8085 microprocessor as shown in figure 9.27. Specify the port addresses of the three counters and control word register.

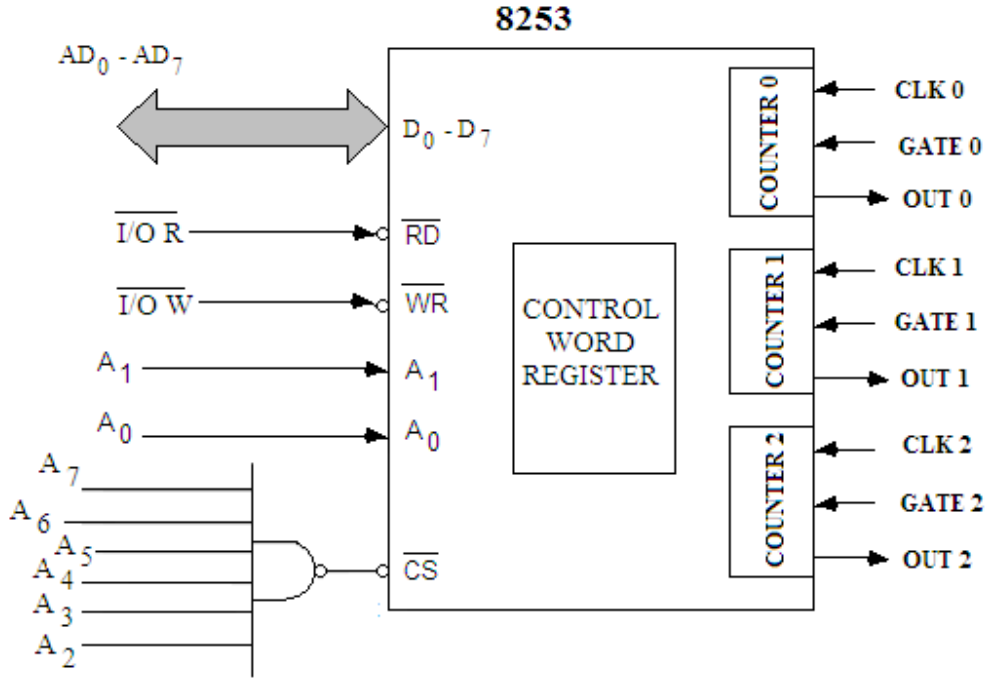


Fig. 9.27

(**Ans.: 00 H for counter 0**
01 H for counter 1
02 H for counter 2
03 H for CWR)

24. Specify the control words for 8253:
    (i) to program counter 1 in mode 5 having a 16-bit count value in BCD;
    (ii) to count the counts on fly for the counter 1.

(**Ans.: (i) 7B H, (ii) 40 H**)

25. Write a program to use counter 0 of 8253 in mode 0 for BCD operation with an initial count value 3648_{10}, and counter 2 in mode 3 for binary operation with an initial count value FF H.

(**Ans.: Program** **MVI A, 31 H**
OUT 13 H
MVI A, 96 H
OUT 13 H
MVI A, 48 H
OUT 10 H
MVI A, 36 H
OUT 10 H
MVI A, FF H
OUT 12 H
HLT)

26. Explain what will be the meaning of the following program for 8253;

(i)
```
MVI A, 94 H
OUT 13 H
MVI A, FF H
OUT 12 H
HLT
```

(ii)
```
MVI A, 38 H
OUT 13 H
MVI A, 04 H
OUT 10 H
MVI A, 02 H
OUT 10 H
HLT
```

(**Ans.: (i) Counter 1 in mode 2 with binary counting,**
(ii) Counter 0 in mode 4 with binary counting)

27. Write a program to use counter 0 of 8253 in mode 4 for BCD operation with an initial count value 0224_{10}.

28. Write a program to use counter 1 of 8253 in mode 5 for Binary operation with an initial count value $2AB6_{10.}$

———————

10

Programmable Keyboard and Display Interface: 8279

The entering of program or data in the microprocessor base systems is most commonly done by a keyboard. Hence, keyboard is the most versatile input device. The keyboards with hexadecimal or ASCII characters are basically a combination of switches placed in a matrix with rows and columns. Similarly, display devices are to be connected to the system to output the data or the result, and the most common output device is seven segment display. The keyboard has to be constantly scanned to detect a key-press. The display, however, has to be supplied with the data to hold it ready. If the CPU is required to do all these operations itself, it will be heavily burdened and will have lesser time for other processes. These operations are, therefore, carried out by another device so that CPU is relieved from all these burdens. For this purpose, Intel 8279 keyboard/display interface is designed to directly connect to 8085 microprocessor. It performs both keyboard scan and output display operations repetitively and very short time of CPU is utilized for the data transfer between the device and CPU. This chapter will entirely deal with the details of this programmable keyboard and display device 8279.

10.1 INTEL PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE 8279

Intel 8279 Programmable Key Board/Display Interface is available in the form of 40 pin IC in plastic dual in line package (DIP). It has been designed to interface the key board (an input device) and display device (an output device) with microprocessor. The 8279 constantly scans to detect a key press and transmit the information of characteristics of the key press to the CPU. It also displays or outputs the data received from CPU to the display devices. These two operations keyboard scan and display are performed repetitively and independently without utilizing the time of CPU except for relatively short time when the data is actually transferred to and from the burden of scanning the keyboard or refreshing the display repetitively.

There are three input modes:

Scanned keyboard mode
Scanned Sensor matrix mode
Strobed input mode.

The keyboard can provide a scanned interface to 64 contact key matrix or array of sensors or a srobed interface keyboard. Key depressions can be 2 key lock out or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO (First In First Out) and set the interrupt lines. If more than 8 characters are entered, overrun

interrupt status is set. The display provides a scanned display interface for LED, incandescent or other popular display technologies.

10.2 BLOCK DIAGRAM OF 8279

The pin diagram, logic diagram and functional block diagram of this 40 pin 8279 IC are depicted in figures 10.1, 10.2 and 10.3 respectively.

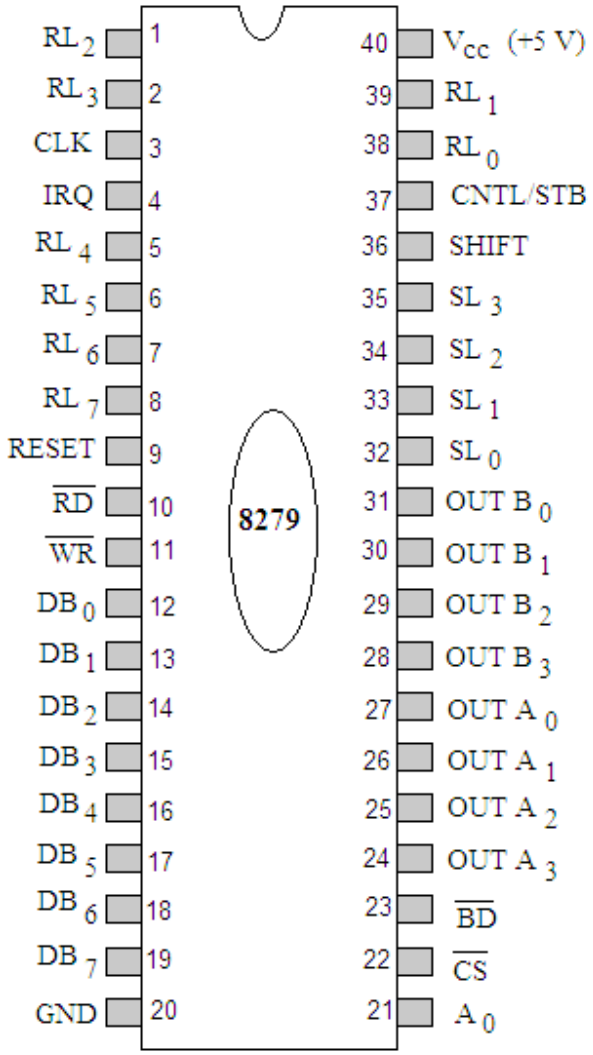


Fig. 10.1

The description of pins of 8279 Programmable keyboard/display interface is given as follows:

DB_0-DB_7: These pins form the bidirectional 8-bit data bus. The commands are also transmitted on this data bus.

CLK: It is a clock terminal to be connected to the external clock terminal of the system clock. It is used to generate internal timing. Maximum frequency to be used is 3 MHz.

$\overline{CS}$: This is a chip select terminal. A low signal to this terminal enables the chip for programming, reading the keyboard, etc.

A_0: It is buffer address pin. A low signal on this pin indicates data and a high on this pin indicates the command.

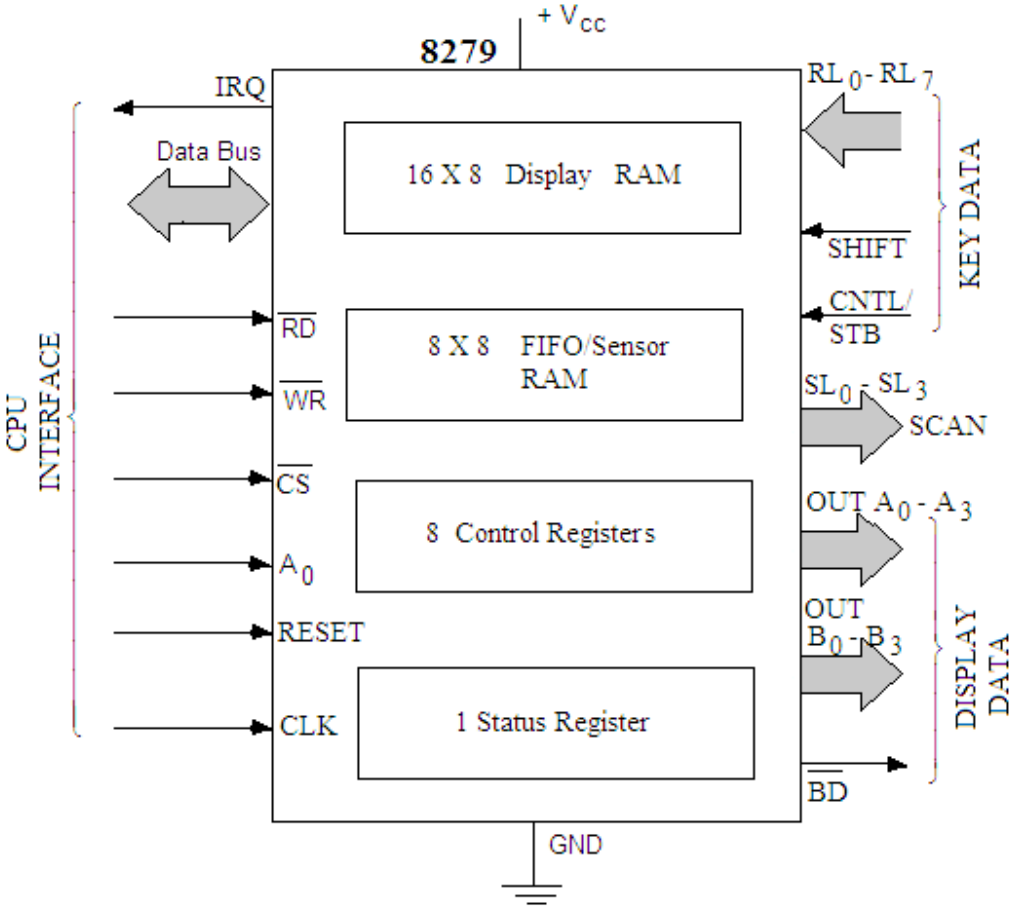


Fig. 10.2

IRQ: It is an interrupt request output terminal. Interrupt request, becomes 1 when a key is pressed, data is available.

SL_0-SL_3 : These four scan lines are used to scan the key switch or sensor matrix and the display digits. Scan line outputs scan both the keyboard and displays.

RL_0-RL_7: These are 8 Return line inputs which are connected to the scan lines through the key or sensor switches. These lines have internal pull-ups to keep high until a switch closure pulls one of the lines low.

SHIFT: This is an input terminal used in the scanned matrix keyboard modes. The SHIFT input status is stored along with the key position on key closure. SHIFT connects to shift key on keyboard.

CNTL/STB: This is control and strobe line, connected to the control key on the keyboard. It is high until a switch closure pulls it low.

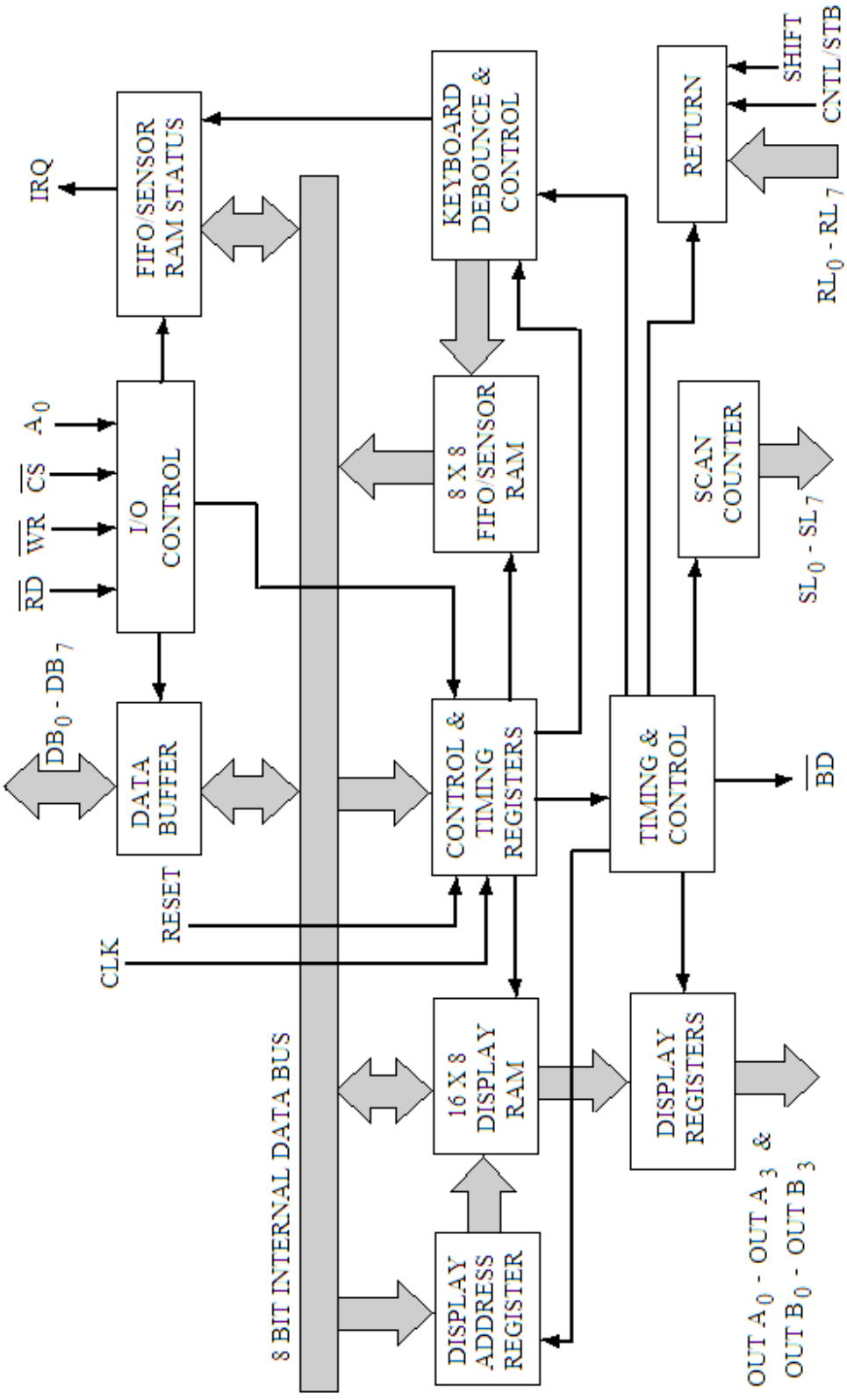

IRQ
FIFO/SENSOR RAM STATUS
KEYBOARD DEBOUNCE & CONTROL
RETURN
SHIFT
CNTL/STB
RL0 - RL7
A0
CS
WR
RD
I/O CONTROL
8 X 8 FIFO/SENSOR RAM
SCAN COUNTER
SL0 - SL7
DB0 - DB7
DATA BUFFER
CONTROL & TIMING REGISTERS
TIMING & CONTROL
BD
RESET
CLK
8 BIT INTERNAL DATA BUS
16 X 8 DISPLAY RAM
DISPLAY REGISTERS
DISPLAY ADDRESS REGISTER
OUT A0 - OUT A3 &
OUT B0 - OUT B3

Fig. 10.3

SHIFT: This is an input terminal used in the scanned matrix keyboard modes. The SHIFT input status is stored along with the key position on key closure.

$\overline{RD}$: It is read input terminal and is connected to microprocessor's RD signal. It reads data/status registers.

$\overline{WR}$: It is write input terminal and is connected to microprocessor's write terminal.

OUT A_0 –OUT A_3 & OUT B_0-OUT B_3: These are two 4-bit ports. The display output is through two 4-bit ports (OUT A_0 –OUT A_3 and OUT B_0-OUT B_3). These two ports can be combined to form an 8 bit port.

$\overline{BD}$: This is a blanking display terminal, used to blank the display during the digit switching or by a display blanking command.

10.3 FUNCTIONAL DESCRIPTION OF 8279

The functional description of programmable keyboard and display interface 8279 will now be discussed with reference to figures 10.2 and 10.3. The 8279 has mainly been divided into four sections:

- Keyboard Section
- Scan Section
- Display Section
- Microprocessor Interface Section

Keyboard Section

The keyboard section includes Return buffer, keyboard debounce and control. Return buffers are to buffer the 8-input lines (RL_0-RL_7). In the keyboard mode, these lines are scanned, looking for the key closure in that row. If the debounce circuit detects a closed switch, it waits about 10 msec to check whether the switch remains closed. If it does, the address of the switch in the matrix in addition to other information is transferred to the FIFO.

The block FIFO/Sensor RAM and Status contains dual function 8 x 8 RAM. It will act as a FIFO in keyboard or strobed input modes. Each new entry is written into successive RAM positions and each is then read in the order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by $\overline{RD}$ and $\overline{CS}$ low and A_0 high. The status logic also provides an IRQ signal when the FIFO is not empty.

Scan Section

The scan section has a scan counter and four scan lines SL_0-SL_7 which are used o scan the key switch or sensor matrix and display digits. This counter can be operated in two ways: Encoded mode and Decoded mode. In the encoded mode, the counter provides a binary count which can be decoded to provide the scan lines for the keyboard or display. However, in case of decoded mode, the scan counter itself is a decoder providing 1 of 4 scan.

Display Section

This section contains the display address registers and display RAM. The display address registers hold the address of the word currently being written or read by the CPU and the two nibbles being displayed. The read/write addresses are programmed by CPU

command. They can also be set to auto increment after each read or write. The display RAM can be directly read by the CPU after the correct mode and address is set. These are two 4-bit ports. The display output is through two 4-bit ports (OUT A_0-OUT A_3 and OUT B_0-OUT B_3). These two ports can be combined to form an 8 bit port. The data from these lines are synchronized to the scan lines (SL_0-SL_3) for the multiplexed digit displays. The two ports may be blanked independently by the blanking display terminal $\overline{BD}$. This section also includes 16 x 8 display RAM and the processor can read from or write into any of these registers.

Microprocessor Interface Section

The microprocessor interface sections contains 8-bit bidirectional data lines (DB_0-DB_7), one interrupt request line (IRQ), one address buffer line A_0, and five lines ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$, RESET, CLK) for interfacing. When a high signal appears on A_0 terminal, the command or status registers inside the 8279 can be accessed i.e. it works as command word or status. A low, however, on this line indicates that the data register, e.g. display RAM or the FIFO/Sensor RAM can be accessed. The interrupt request line IRQ becomes high whenever data entries are stored in the FIFO. This signal is used to interrupt the microprocessor to indicate the availability of the data. The data and commands are communicated between the microprocessor and the 8279 through the data bus (DB_0-DB_7). The RESET pin resets the 8279, when a high signal appears on this pin. The two signals $\overline{RD}$ and $\overline{WR}$ enable the data bus to either send data to external bus or receive it from the external bus.

10.4 KEYBOARD SCAN

As already discussed, the 8279 provides 4 scan lines (SL_0-SL_3) and 8 return lines (RL_0-RL_7). The scan lines can be operated either in encoded mode or in decoded mode.

Encoded Mode:

In this encoded mode, four scan lines (SL_0-SL_3) can be used to generate 16 lines with the help of external decoder. Usually 3 to 8 line decoder is used with the scan lines. The most significant scan line SL_3 is not recommended to use in keyboard decoder. So with three scan lines (SL_0-SL_2) and a 3 to 8 line decoder, 8 decoded scan lines are generated. These 8 decoded scan lines in conjunction with the 8 return lines can form an 8 x 8 keyboard matrix as shown in figure 10.4. Two more extra lines SHIFT and CNTL (control) can also provide four more different combinations as shown in table 10.1.

Table 10.1

SHIFT	CNTL
0	0
0	1
1	0
1	1

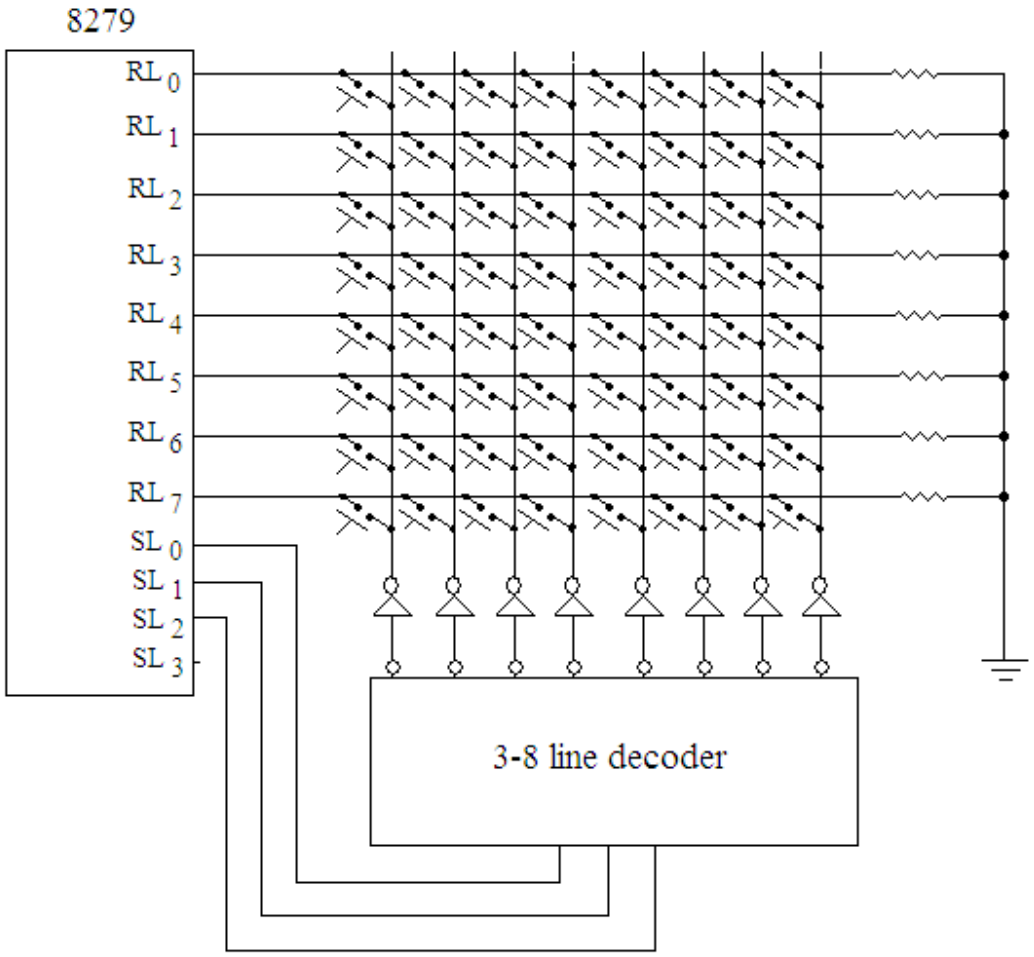


Fig. 10.4

With these four combinations and 8 x 8 matrix 4 x 8 x 8 = 256 character definitions are possible.

Decoded Mode:

In the decoded mode, an internal decoder is used to provide all the four scan lines (SL0-SL3) as the decoded lines as shown in table 10.2.

Table 10.2

SL_3	SL_2	SL_1	SL_0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

These four decoded lines with 8 return lines form the keyboard matrix as shown in figure 10.5. The SHIFT and CNTL lines in combination with the keyboard matrix can provide 4 x 4 x 8 128 character definitions.

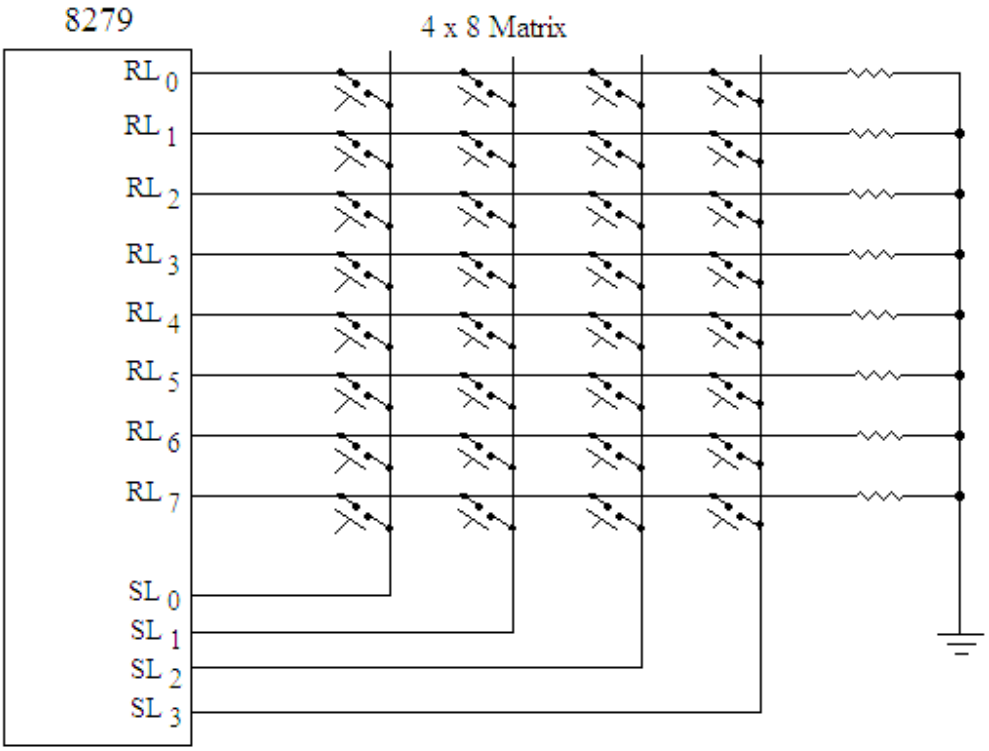
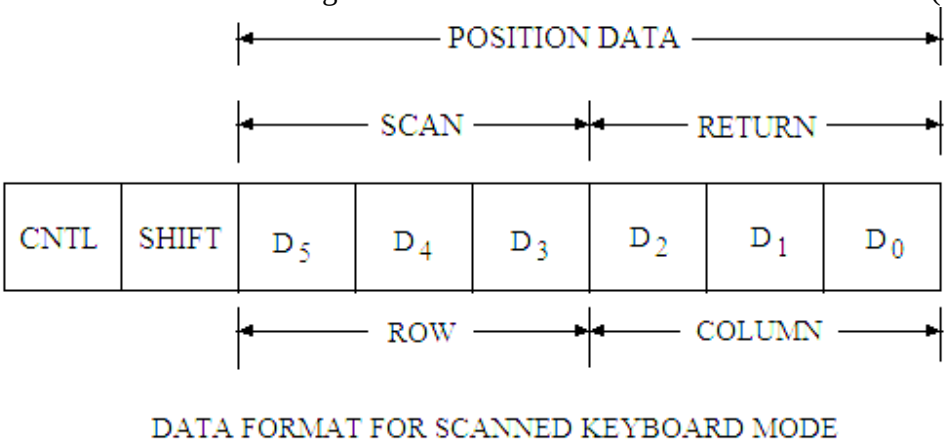


Fig. 10.5

The disadvantage of the decoded scan is that the number of combinations of the scan lines as given in table is only four. Hence, only four rows can be used in case of keyboard and only four digits can be used for the display.

10.5 SCANNED KEYBOARD

In this mode, when a key is pressed, a unique 6 bit data is generated characteristic of the key position. An 8-bit word is formed, with the 6-bit position data for the key pressed and two bits for control (CNTL) and SHIFT lines. The format for the scanned keyboard mode is shown in figure 10.6. The scan counter has three scan bits (D_5-D_3) as



DATA FORMAT FOR SCANNED KEYBOARD MODE

Fig. 10.6

000 to 111 for the row on which the pressed key is located. The column counter also has three bits (D_2-D_0) as 000 to 111 for column on which the pressed key is located. Figure

10.7 shows the 8 x 8 keyboard matrix that has the 8 return lines and 8 scan lines. It has 64 key positions (0 to 63) which generate 8 bit data word for the key pressed. The 8 bit data word generated for a key pressed may be understood if we consider CNTL and SHIFT lines are 00. Suppose a key number 15 is pressed. The row line corresponding to the key number 15 is 1 (three bit binary is 001) and the column line for this pressed key is 7 (three bit binary is 111) so the 8 bit data formed for this pressed key becomes:

00 001 111

which is the binary equivalent of 15 (0F H). Similarly, the 8 bit data word for the key pressed 32 is 00 10 000 (20 H).

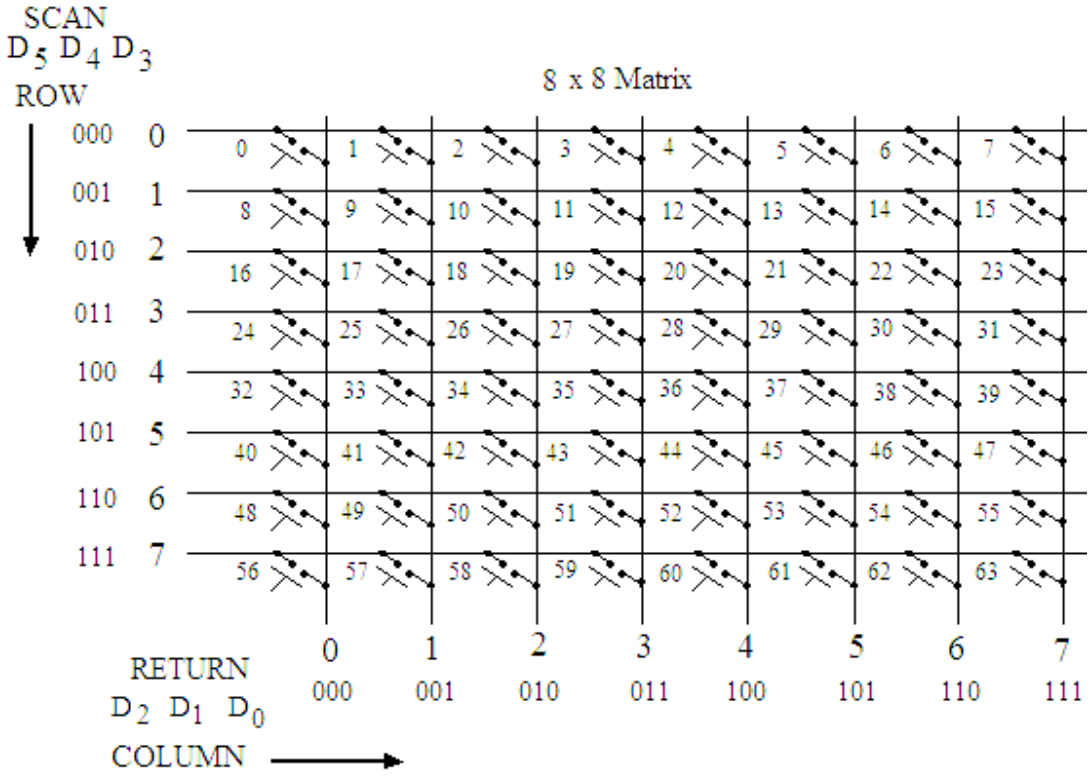


Fig. 10.7

The 8-bit data thus formed corresponding to the key pressed is stored in FIFO (first in first out) RAM inside the 8279. As soon as the 8-bit word is stored in FIFO RAM, IRQ terminal of 8279 goes high. This terminal is connected to one of the hardware interrupt line of CPU. With the high IRQ, the interrupt line of the CPU will be activated and the service subroutine program of the hardware interrupt will read the data from the FIFO RAM. As the data is read from the FIFO RAM, the IRQ terminal goes low; and it becomes high again if FIFO RAM contains further data.

The scanned keyboard mode has further two alternative ways of operation:

- Two-key lockout
- N-key rollover

10.5.1 Two-key Lockout

The mechanical keys used in the keyboard have a problem. When a key is pressed the contacts bounce back and forth and finally settle down after a small time. Due to this contact bounce problem multiple entries may be made for the same key. This can be avoided either by using the debouncing circuit using flip-flops etc. or by making the device to wait for few milliseconds after the key is pressed. In the 8279, second method is used for debouncing which is the built in feature of this IC.

In the two-key lockout operation, if two keys are depressed within the bounce cycle, it is called a simultaneous depression and neither key will be recognized until one of the keys is released. The last key released will be recognized and its corresponding 8-bit code will be entered in FIFO RAM.

If after the first key is depressed, and no other key is found to be depressed within next two scans, then it is taken as a single key depression and the code for the depressed key will be entered in FIFO RAM.

If after the first key depression, one or more additional key depression is detected within the next two scans, then there will be following two possibilities:

If all keys are released before the first pressed key, then the first key data will be entered in FIFO RAM.

If first key pressed is released before others, then the press key will be entirely ignored.

10.5.2 N-key Rollover

In this mode, each key depression is treated independent. If simultaneous key depression occurs then keys are recognized and entered in FIFO RAM according to the order of the key pressed. In fact when a key is pressed the debounce circuit inside the 8279 waits for two scans then checks if this key is still pressed. If this key is still pressed, the code for the key depressed is entered into FIFO RAM.

10.6 SCANNED SENSOR MATRIX

As discussed earlier, the keyboard matrix size is 8 x 8 in encoded scan lines and 4 x 8 in decoded scan lines. In this mode, the keys are placed in the form of matrix either in 8 x 8 encoded scan lines or 4 x 8 decoded scan lines, the scan lines form the columns and return lines form the rows of the keyboard matrix. The key status (open or closed) is stored in RAM which can be addressed by the CPU. The data on each of the eight lines enter directly in eight columns of sensor RAM; and each switch position maps to specific sensor RAM positions. The SHIFT and CNTL lines are not considered as inputs. The format for each row of the sensor RAM is shown in figure 10.8. The logic circuits can also be connected to the return lines which will be triggered by the scan lines. The debouncing circuit is not provided in this mode. It therefore has the advantage that the CPU knows how long the sensor was closed. The IRQ line goes high if a sensor value is found to have changed at the end of sensor matrix scan. The IRQ line is cleared by the first data read operation if the auto increment flag is set to zero or by the End Interrupt Command if the auto increment flag is set to one.

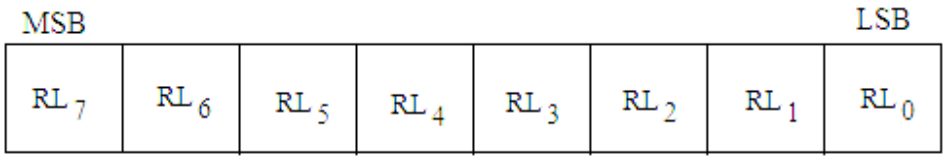


Fig. 10.8

10.7 STROBED INPUT

In this mode, the data is accepted from the return lines and go to FIFO RAM and entered at the rising edge of CNTL/STB line pulse. The data placed on the return lines can come from any source. The data is stored in the same format as shown in figure 10.8. Each scan line would lead to an 8-bit word.

10.8 DISPLAY INTERFACE

The interfacing of keyboard with the 8279 has been discussed in the preceding sections. The interfacing of display devices with the 8279 will now be discussed. Generally seven segment display devices are connected with 8279 using the multiplexing technique. In the multiplexing technique the seven segment code is sent to all the displays simultaneously, but the particular segment to be illuminated is only grounded (in case of common cathode displays).

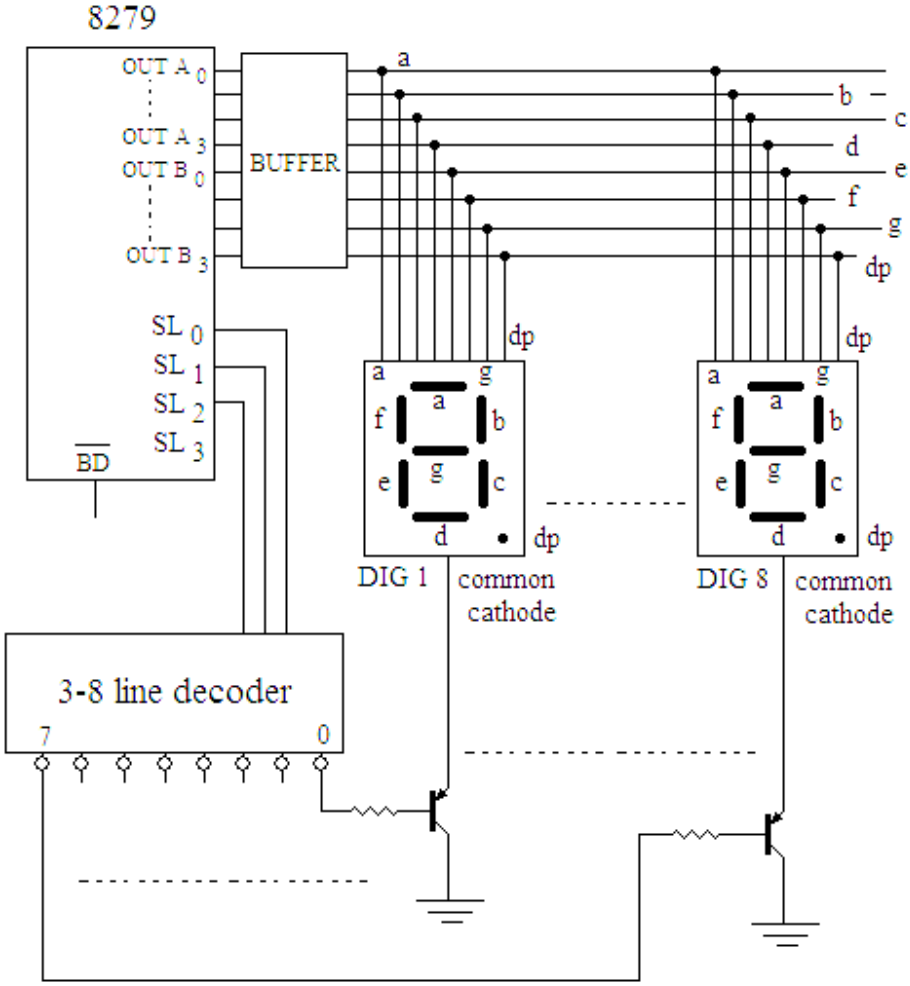


Fig. 10.9

In the 8279, eight output terminals, OUT A_0-A_3 and OUT B_0-B_3 are provided for the purpose of interfacing the seven segment displays. If a 4-to-16 line decoder is used with the scan lines (SL_0-SL_3), then a maximum of 16 display devices may be connected to the 8279. The internal FIFO RAM of 8279 can hold 8-bit data for 16 digits. If a 3-to-8

line decoder is used with three scan lines (SL_0-SL_2), then a maximum of 8 displays may be connected to this IC. All the segments of display devices are connected in parallel (i.e. a's segment of all the displays are tight together, similarly for b's to g's and dp's segments). These segments are then connected to the output terminals OUT A_0-A_3 and OUT B_0-B_3, as shown in figure 10.9. The OUT A_0, OUT A_1, OUT A_2 and OUT A_3 lines of the 8279 are connected to segments a, b, c and d respectively. The OUT B_0, OUT B_1, OUT B_2 and OUT B_3 lines of the 8279 are connected to segments e, f, g and dp respectively. The decoded outputs of scan lines provide 0 to 7 outputs in a periodic fashion, to select one digit of the display devices at a time. The $\overline{BD}$ line is used to blank all display digits.

When a given bit is 1, the corresponding segment is switched ON. For example the key code for the alphabet 'C' is obtained if the segments a, d, e and f are high. The data code for the alphabet 'C' is 93 as shown in figure 9.10.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	Out terminals of 8279
d	c	b	a	dp	g	f	e	Segments of display
1	0	0	1	0	0	1	1	= 93 H

Fig. 10.10

10.9 DISPLAY MODES

The interfacing of display devices with 8279 has been discussed in the earlier section. Further there are following two options for the display formats in the display modes of 8279:

- Left Entry Mode (Type Writer Mode)
- Right Entry Mode (Calculator Mode)

10.9.1 Left Entry Mode (Type Writer Mode)

In the left entry mode, the first location of the display RAM data is treated as the segment for the left most digit and second location of the display is treated as the segment data for the second digit from the left and so on. This is similar to typing the paper with

the type writer. In this mode there is auto-increment facility as in the type writer; the

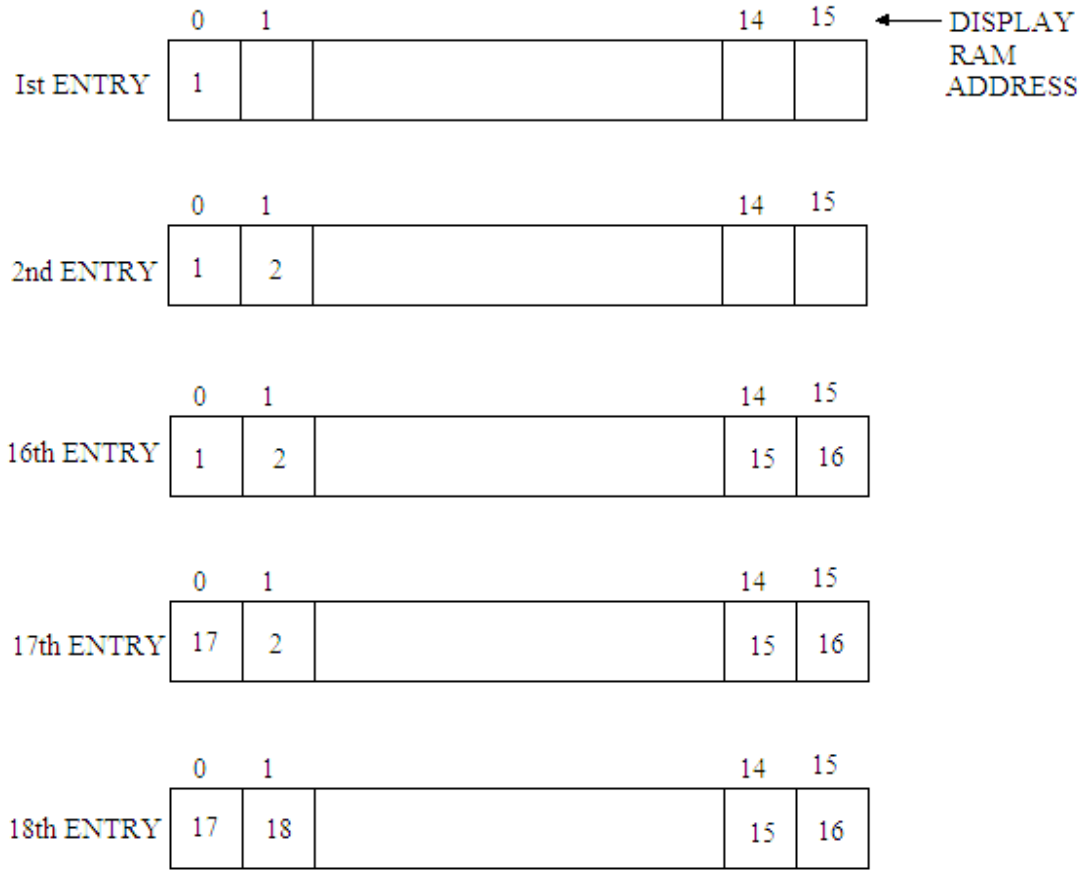


Fig. 10.11

carriage advances one step during typing. The left entry mode with auto increment facility is illustrated in figure 10.11 for 16 digits to be displayed on the display devices. From this figure it is clear that the first entry goes to the address 0 (first one of sixteen) for one word RAM and to the left most display position. The second entry goes to address 1 and the second display position and so on. The 16^{th} entry goes to the address 15 of the display RAM and position 16 of the display. The 17^{th} entry fill the left most position again.

There is a command (the details of which will be discussed in the next section) which allows entering data at an arbitrary address location of the display RAM. Figure 10.12 illustrates the result of a command that is used to display the next data to the 6^{th} position using 8-digit display. The command word given here is 10010110 which contains 8 bits, the three most significant bits 100 represents the "write display RAM": the next bit 1 is for auto increment and the next four bits 0110 (6^{th}) are for the position at which it should start filling. This command does not lead to any undesirable result.

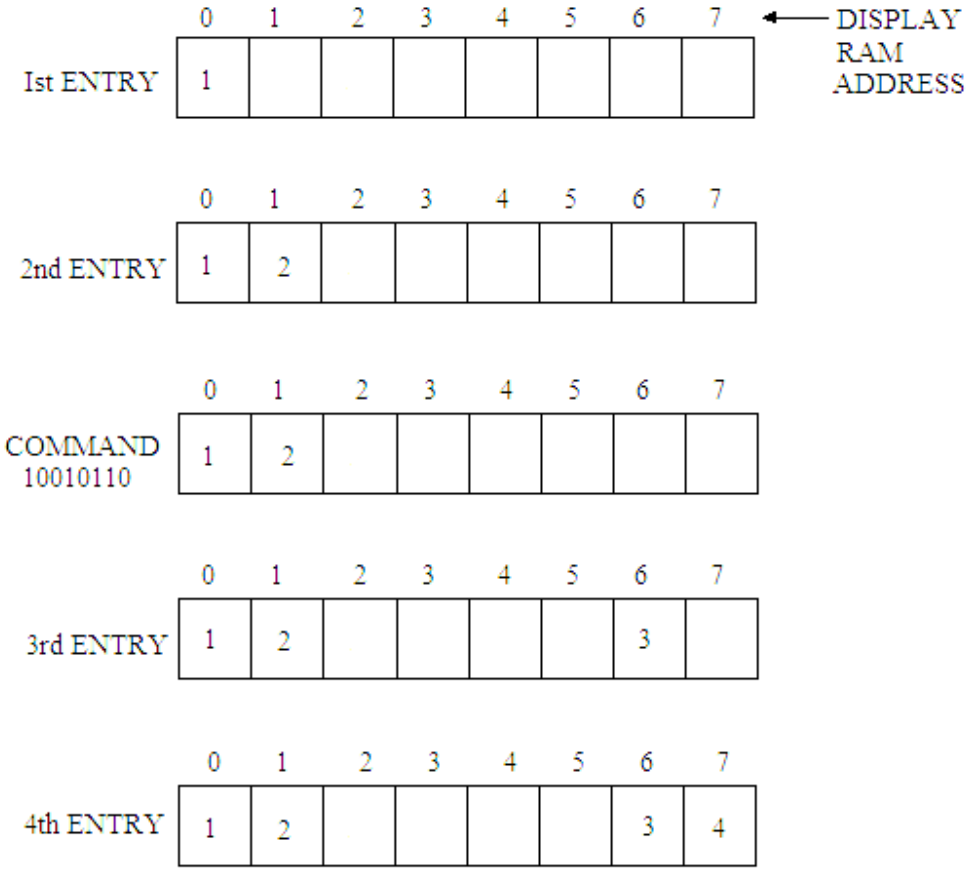


Fig. 10.12

10.9.2 Right Entry Mode (Calculator Mode)

The right entry mode is also known as calculator mode. In this mode the first location of the display RAM indicates the display data for the right most digit. Therefore, on the display, the data appears to start from the right and shift towards left as the digits move left in the calculator when the data is entered in to it. Figure 10.13 shows how the data are entered in this mode for 16 digits display with auto increment. The first character appears at the right display position. When a second character enters, the first character shifts towards the left by one place on the display. Similarly at the third entry, both the characters move by one place left each and the third character again takes the original right most position. It has been observed that a given character entered at a certain display position continues to remain there; but in case of right entry mode at every new entry each existing character moves one place to the left and finally the left most character shifts off the end, and is lost.

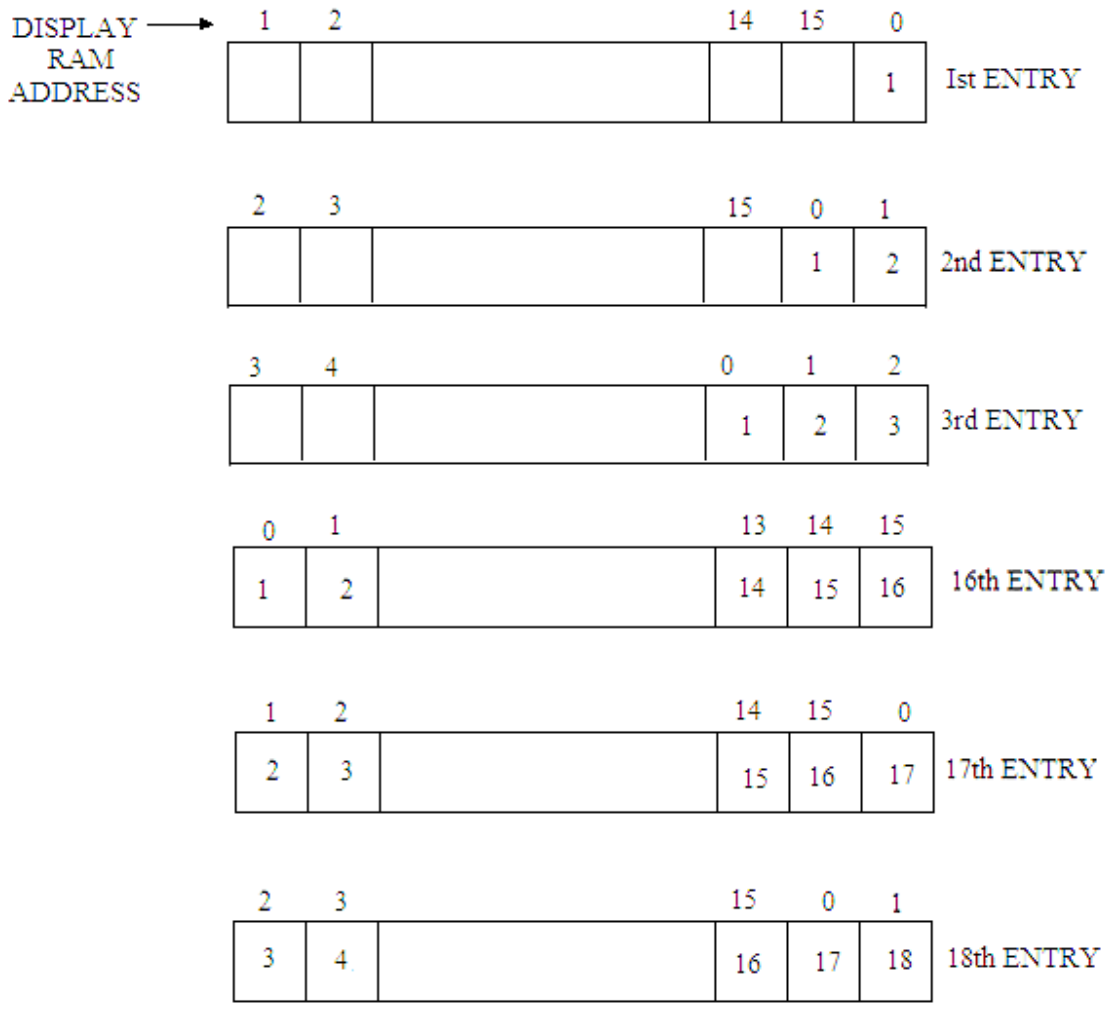


Fig. 10.13

Figure 10.14 shows the entering of the data in right entry mode with auto increment for 8 digit display using the command word for entering data at an arbitrary address location of the display RAM. The results obtained for entering data at an arbitrary address location in this mode are unexpected. Therefore, a starting display RAM address 0 and sequential entry are recommended in this right entry mode.

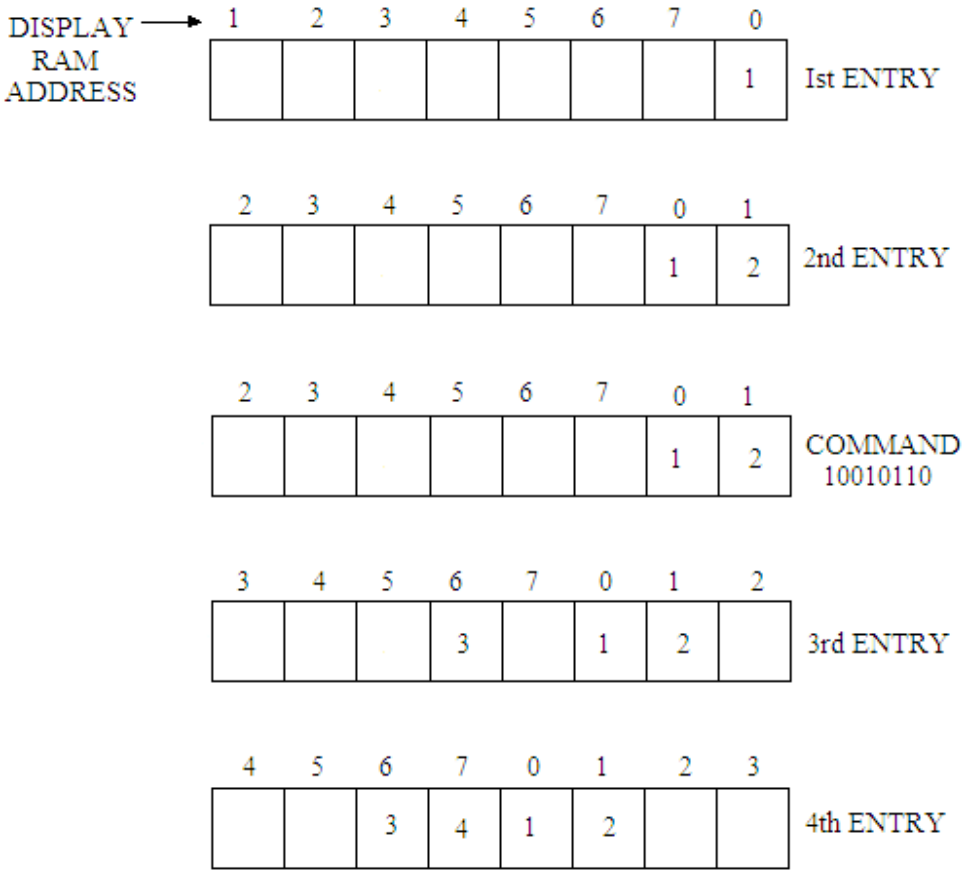


Fig. 10.14

10.10 PROGRAMMING OF 8279

The 8279 is a programmable keyboard and display interface device so it may be programmed for the desired operation. There are following 8 commands that can be used in the 8279:

- Keyboard/Display Mode Set
- Program Clock
- Read FIFO/Sensor RAM
- Read Display RAM
- Write Display RAM
- Display Write Inhibit/Blanking
- Clear
- End Interrupt/Error Mode Set

The command word, whose format is shown in figure 10.15, is sent on the data bus with $\overline{CS}$ low and A_0 high. The word is loaded to the 8279 using the write operation.

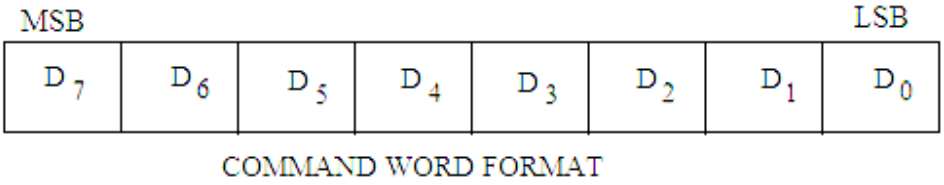


Fig.10.15

In this command word format, the three most significant bits selects the various operations which are given as:

D_7	D_6	D_5	**Function**
0	0	0	Keyboard/Display Mode Set
0	0	1	Program Clock
0	1	0	Read FIFO/Sensor RAM
0	1	1	Read Display RAM
1	0	0	Write Display RAM
1	0	1	Display Write Inhibit/Blanking
1	1	0	Clear
1	1	1	End Interrupt/Error Mode Set

10.10.1 Keybaord/Display Mode Set

This command sets up the operation of keyboard and display mode. The command bit pattern for the same is given as (figure 10.16):

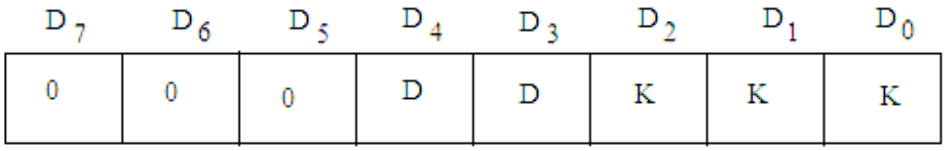


Fig. 10.16

D D represents the display mode and **K K K** represents the keyboard mode. **D D** in D_4 D_3 bit positions has the following four display mode options:

D_4	D_3	**Display Option**
0	0	Eight 8-bit character display with left entry
0	1	Sixteen 8-bit character display with left entry
1	0	Eight 8-bit character display with right entry
1	1	Sixteen 8-bit character display with right entry

K K K in D_2 D_1 D_0 bit positions has the following four display mode options:

D_2	D_1	D_0	Keyboard Option
0	0	0	Encoded Scan Keyboard with 2-key lockout
0	0	1	Decoded Scan Keyboard with 2-key lockout
0	1	0	Encoded Scan Keyboard with N-key roll over
0	1	1	Decoded Scan Keyboard with N-key roll over
1	0	0	Encoded Scan Sensor Matrix
1	0	1	Decoded Scan Sensor Matrix
1	1	0	Strobed Input Encoded Display Scan
1	1	1	Strobed Input Decoded Display Scan

The command word to set decoded scan keyboard with 2-key lockout and to have eight 8-bit characters with right entry in the display mode can be shown as:

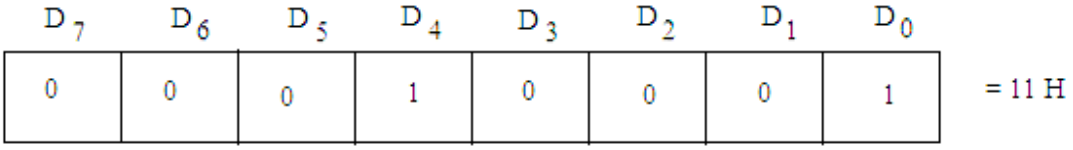


Fig. 10.17

If the address of the command port for 8279 is FF H, then by executing the following instructions, keyboard and display mode will be set as per the requirement discussed above:

```
MVI  A, 11 H
OUT  FF H
```

10.10.2 Program Clock

The different keys of the keyboard are scanned one by one in a sequence to detect a key press by the 8279; and also key debouncing is implemented by this device. All these functions require the timing signal. The 8279, therefore, needs an internal clock. The frequency of this clock should be around 100 KHz. But the clock of the system (i.e. the frequency of the external clock terminal of 8085) is 3 MHz. This system clock or any other external clock signal of known frequency may be applied to the CLK terminal (pin no. 3) of 8279. The 8279 internally provides an arrangement so that the clock applied at the pin no. 3 may be divided by a known factor to get the clock frequency of about 100 KHz. The frequency division is possible by the software.

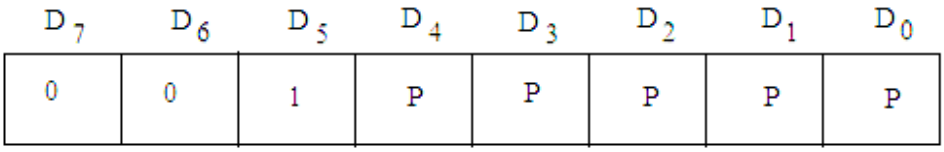


Fig. 10.18

A command word is generated to get the required clock frequency after division of the system clock or external clock connected to pin no. 3 of 8279. The command word format is given in figure 10.18.

The five bits (D_0 to D_4) of the command word define the scale factor. These five bits P P P P P may be set 0 0 0 0 0 to 1 1 1 1 1 whose decimal equivalents are 0 to 31. So the scale factor may be chosen up to 31.

The scale factor will be given by:

$$\text{Scale Factor} = \frac{\text{System} - \text{frequency}}{100\,\text{KHz}}$$

If the external clock frequency applied to pin 3 of 8279 is 2 MHz, then the scale factor will be:

$$\text{Scale Factor} = \frac{2\,\text{MHz}}{100\,\text{KHz}} = 20$$

The binary equivalent of 20 in five bits is 1 0 1 0 0 (represent P P P P P) and the command word will be given by (figure 10.19):

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	1	0	1	0	0	= 34 H

Fig. 10.19

Similarly, if the system clock frequency of 3 MHz is applied to 8279, then the scale factor is given by:

$$\text{Scale Factor} = \frac{3\,\text{MHz}}{100\,\text{KHz}} = 30$$

The binary equivalent of 30 is 1 1 1 1 0 and the command word will be given by (figure 10.20):

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	1	1	1	1	0	= 3E H

Fig. 10.20

The following instructions after execution will set the program clock to 100 KHz, if the system clock frequency of 3 MHz is applied to 8279. The command for this is 3E H as discussed above.

```
MVI  A, 3E H
OUT  FF H
```

The address of the command port for 8279 is assumed as FF H.

10.10.3 Read FIFO/Sensor RAM

The command word for setting up the 8279 to read FIFO/sensor RAM is shown in figure 10.21.

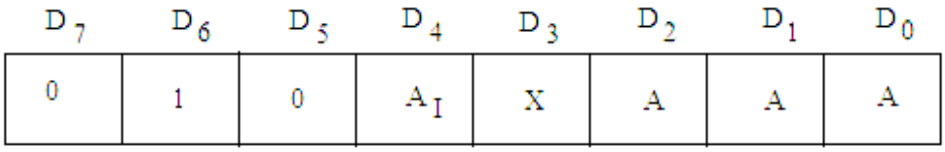


Fig. 10.21

The bits D_7 D_6 D_5 represent for command word 2 (0 1 0).

A_I represents Auto Increment.

X is don't care.

A A A (D_2 D_1 D_0) represent RAM address bits.

In the sensor matrix mode, the address bits A A A represent on the 8 rows of the sensor RAM. If A_I is set to 1, the successive Read operation is performed from the subsequent row of the sensor RAM. In the keyboard mode, auto increment bit AI and address bits A A A are irrelevant and the 8279 will automatically drive the data for each subsequent read (A_0 = 0) in the same sequence in which the data first entered in FIFO RAM.

If the data is to be read from the 3rd row of the sensor RAM, then the command word will be as:

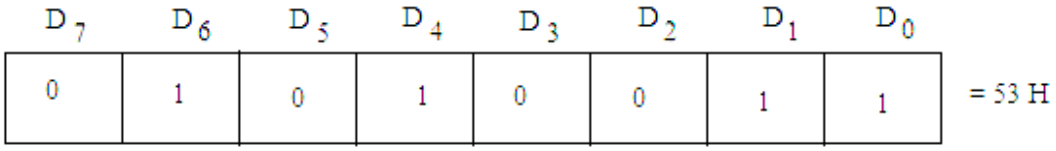


Fig. 10.22

10.10.4 Read Display RAM

The command word for setting up the 8279 to read display RAM is shown in figure 10.23.

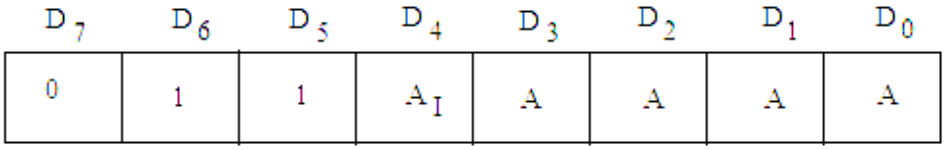


Fig. 10.23

The bits D_7 D_6 D_5 represent for command word 3 (0 1 1).

A_I represents Auto Increment.

A A A A (D_3 D_2 D_1 D_0) select one of the 16 rows of display RAM that is to be read.

If the auto increment bit A_I is set to 1, then the row address will automatically be incremented after each of the Read to display RAM. The command word 0 1 1 1 0 0 0 0 (= 70 H) represents a read from the display RAM.

10.10.5 Write Display RAM

To display information, the seven segment data is to be written in the internal display buffer. To enable writing seven segment data into display buffer, the write display RAM command has to be written into the command word register. The command word for the same is shown in figure 10.24.

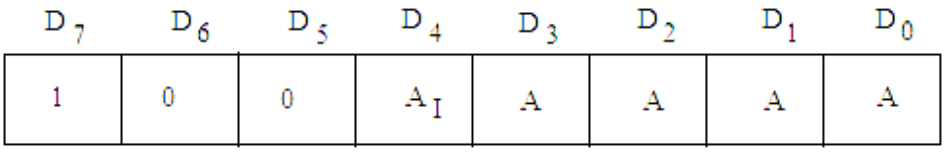


Fig. 10.24

The bits D_7 D_6 D_5	represent for command word 4 (1 0 0).
A_I	represents Auto Increment.
A A A A (D_3 D_2 D_1 D_0)	is the address of the digit that the CPU writes into the buffer. Four bit address is provided to select any one of the digits.

After the write command with A_0 = 1, all the subsequent writes with A_0 = 0 will be to the display RAM

For example, to write the segment codes 63 H and B5 H in the first and second locations of the display RAM, the command will be given as:

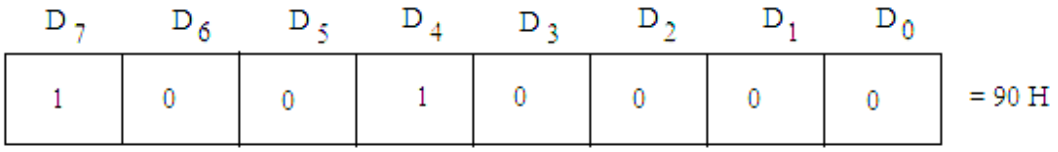


Fig. 10.25

The following instructions will write the segment codes 63 H and B5 H in the first and second locations of the display RAM:

```
MVI A, 90 H
OUT FF H
MVI A, 63 H
OUT FE H
MVI A, B5 H
OUT FE H
```

The first two instructions MVI A, 90 H and OUT FF H writes the command word in the command word register (FF H is the address of the command word register with A_0 is 1). The next two instructions MVI A, 63 H and OUT FE H writes the segment code 63 H in the first location of the display RAM (OUT FE H is the address for the same with A_0 = 0); and the last two instructions MVI A, B5 H and OUT FE H writes the segment data B5 H in the next location of display RAM as the in the command word auto increment A_I bit is set to 1.

10.10.6 Display Write Inhibit/Blanking

The control word for Display Write Inhibit/Blanking is shown in figure 10.26.

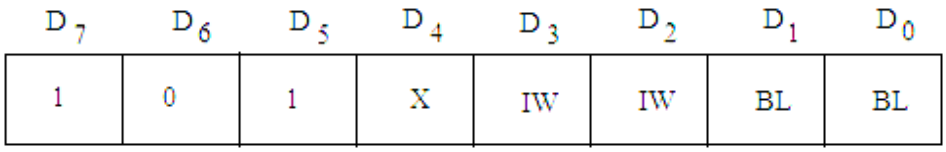


Fig. 10.26

The bits D_7 D_6 D_5 represent for command word 5 (1 0 1).

X represents don't care may be taken as 0.

IW Inhibit Write Flag.

BL Blank Display Flag.

The display write inhibit control word inhibits writing to either left most 4 bits of the display or right most 4 bits. By setting D_3 (IW) bit to 1, the left most significant 4 bits of the display will be masked or inhibited and similarly, by setting D_2 (IW) bit to 1, the right most significant 4 bits will be masked or inhibited.

Similarly, the blank flag (BL) blanks half of the output pins. By setting D1 (BL) bit to 1, the left most significant 4 bits will be blanked (or turned off); and by setting D0 (BL) bit to w, the right most significant 4 bits will be blanked.

10.10.7 Clear

The command word to clear the display, FIFO or both is shown in figure 10.27.

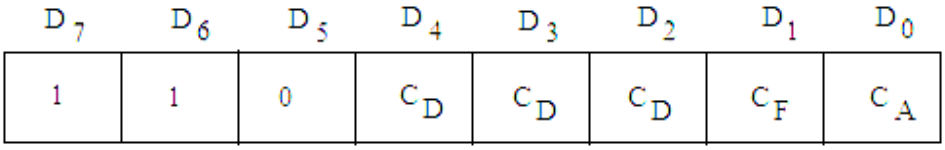


Fig. 10.27

The bits D_7 D_6 D_5 represent for command word 6 (1 1 0).

C_A Clears All (it clears both display RAM and FIFO).

C_F Clears FIFO Status and resets the IRQ line.

C_D C_D C_D Clears all rows of the display RAM to a selectable blanking code.

The selectable blanking code C_D C_D C_D is defined as follows:

C_D	C_D	C_D	
	0	X	All display RAM locations become 00000000 (all zeros).
	0	0	AB becomes 00100000 (20 H).
	1	1	All ones 11111111 (FF H).
			Enables clear display when 1.

If C_F bit is set to 1, it clears FIFO and the display RAM status, and sets address pointer to 000; it also resets the IRQ lne.

The bit C_A has the combined effect of CD and CF; if it is set to 1 it clears the display RAM and also clears FIFO status.

10.10.8 End Interrupt/Error Mode Set

The command word to end interrupt/error mode set is shown in figure 10.28.

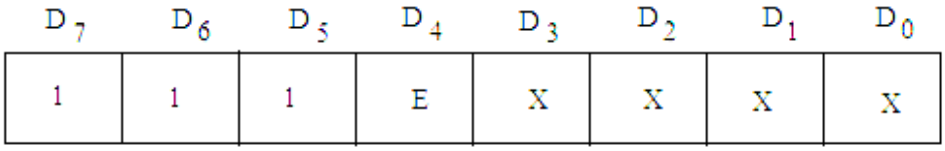


Fig. 10.28

The bits D_7 D_6 D_5 represent for command word 6 (1 1 0).

The bit D_4 represents Error (E) which may be programmed to set the error mode or clear the IRQ line.

The bits D_3 - D_0 are represented by X, means don't care and these may be considered as 0s.

For the sensor matrix mode this command lowers the IRQ line and enables writing into RAM.

For the N-key rollover if the bit E is programmed to 1 the chip will be operated in the special error mode.

10.11 STATUS REGISTER (IN OPERATION)

The status word gives the information about how many characters are in the FIFO RAM and whether an error has occurred. The status word can be read by the CPU when A_0 = 1 or in other words we can say that the status word can be read by the command register (IN FF H in the present case if FF H is the address of the command register).

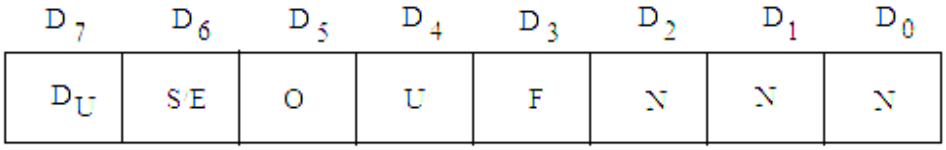


Fig. 10.29

The bit D_7 represents D_U - Display Unavailable.

The bit D_6 represents S/E - Sensor Closure/Error Flag for multiple closure.

The bit D_5 represents O- Over-run error.

The bit D_4 represents U- Under-run error.

The bit D_3 represents F- FIFO Full.

The bits D_2 to D_0 represent NNN as the number of key codes in FIFO RAM.

In this status word the first three bits D_2 to D_0 labeled as NNN identify the number of key codes or characters that are currently present in the FIFO.

The next bit F (D_3) indicates whether FIFO is Full or Empty. If F = 1, then it means FIFO is full. The FIFO is empty is F = 0.

The next two bits U and O (D_4 and D_5) are related to the under-run or over-run errors. Over-run error indicates that an attempt was made to enter the key code or character in FIFO RAM when it was already full. The other error under-run means the processor attempted to read the FIFO when it was empty.

In the special error mode, the bit S/E (D_6) is showing that an error flag gives an indication whether a simultaneous multiple closure error has occurred.

The most significant bit D_U (D_7) of the status word stands for display unavailable. When the clear display command is send, the 8279 clears the display and clearing requires some very small time. In this duration the display is unavailable and the data can not be read or written in display RAM. The bit D_U is 1 when clearing is going on and it is automatically reset when display RAM becomes available again for writing.

Read Operation

To read FIFO RAM for the keyboard data following steps are carried out:

Step I Send Keyboard/display command word to command register. Say:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	0	0	= 00 H

Eight 8-bit Character display - left entry.

Encoded scan keyboard - 2 Key lockout.

Step II A read FIFO RAM command is written into the command register as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	0	A_I	X	A	A	A	
0	1	0	0	0	0	0	0	= 40 H

(A_I and A A A are irrelevant in sensor matrix mode).

Step III Read the status word by using IN instruction and port address is for A_0= 0. Further, status word is checked, if D_0 bit is 1. If it is 1, which indicates one character or key code is available in the first location of FIFO RAM.

Step IV Read FIFO RAM command word is entered into the command register.

Step V Now the key data can be read from the data register with A_0 = 0.

Using these steps program may be written as given below:

Program:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	00 H	; Encoded scan, 2 key lockout mode.
	OUT	FF H	; Keyboard display code is written in command register.
LOOP	IN	FF H	; Read the status word.
	ANI	01 H	; Check if D_0 is 1.

```
JZ      LOOP    ; If D0 = 0 then read the status word
                  again till D0 is 1.
MVI  A, 40 H    ; Load FIFO RAM command word
OUT     FF H    ; in command word register.
IN      FE H    ; Read the data register to take the
                  key code.
```

10.12 INTERFACING OF 8279 WITH 8085

Since the 8279 is an I/O device, so it can be used as memory mapped I/O device or I/O mapped I/O device. The interfacing of this IC with 8085 microprocessor is shown in figure 10.29.

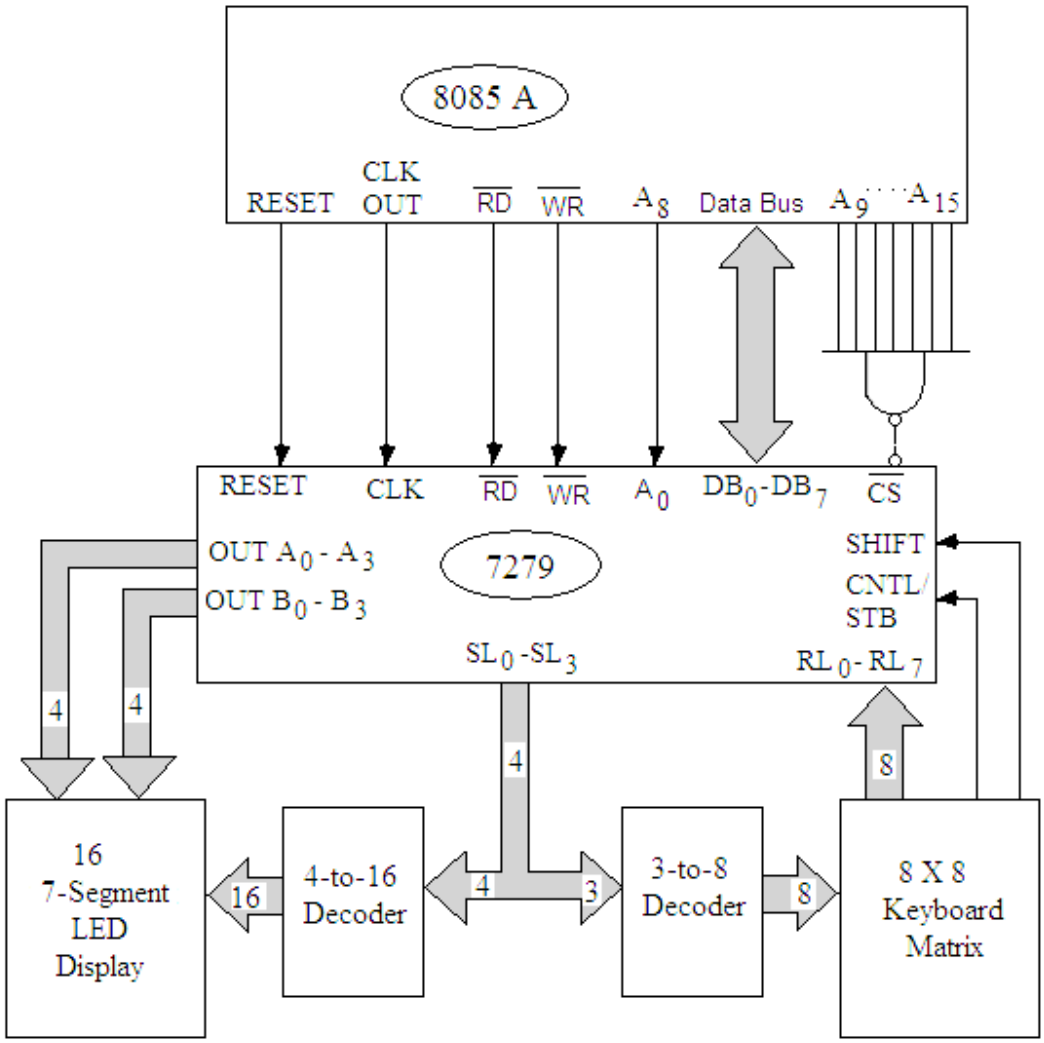


Fig. 10.29

The description of the connections of this IC with 8085 microprocessor is given below:

- The 8 data lines (DB_0-DB_7) are connected to the data bus of the microprocessor.

- The RESET signal of 8279 is connected to the RESET OUT terminal of the CPU.
- The active low terminals $\overline{RD}$ and $\overline{WR}$ of 8279 are connected to active low terminals $\overline{IOR}$ and $\overline{IOW}$ of the CPU.
- The CLK signal of this IC may be connected to either an external clock signal or the CLK OUT terminal of the processor.
- The buffer address terminal A_0 of 8279 is connected to the A_8 terminal of the address bus of the microprocessor. A low signal on this pin indicates data word and a high on this pin indicates the command word.
- The output of the address decoder circuit is connected to active low terminal $\overline{CS}$ of the IC 8279. The A_9-A_{15} terminals of the address bus of the microprocessor are connected to the inputs of the decoder circuit. These connections show that FF H is the address of the command register and FE H is the address of the data register.
- The Scan lines SL_0-SL_2 of the 8279 are connected to a 3 to 8 line decoder, the outputs of which are connected to the 8 x 8 keyboard matrix. The eight return lines RL_0-RL_7 along with the CNTL (control) and SHIFT terminals are also connected with the keyboard matrix.
- The sixteen 7-segment LED displays are also connected to this IC 8279 through the OUT A_0-A_3 and OUT B_0-B_3 terminals. The common terminals of the digit display are connected to the Scan lines SL0-SL3 through 4-to-16 line decoder.

Example 10.1. *A 4 x 4 keyboard matrix is to be interfaced with 8085 microprocessor using 8279. Give the necessary hardware for the same. Also give the software to enter the hex code of the key pressed and store this code to memory location 2500 H.*

Given frequency of the clock connected to the CLK terminal of 8279 is 2.5 MHz. Assume Decoded scan mode with N-key-roll over. Let control port address is 81 H and data port address is 80 H.

Solution. The hardware needed to interface 4 x 4 keyboard matrix with 8085 microprocessor is given in figure 10.30.

The initialization steps for providing the software are given below:

1. Send Keyboard/display command word to command register. Say:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	1	1	= 03 H

Eight 8-bit Character display - left entry.

decoded scan keyboard - N Key rollover.

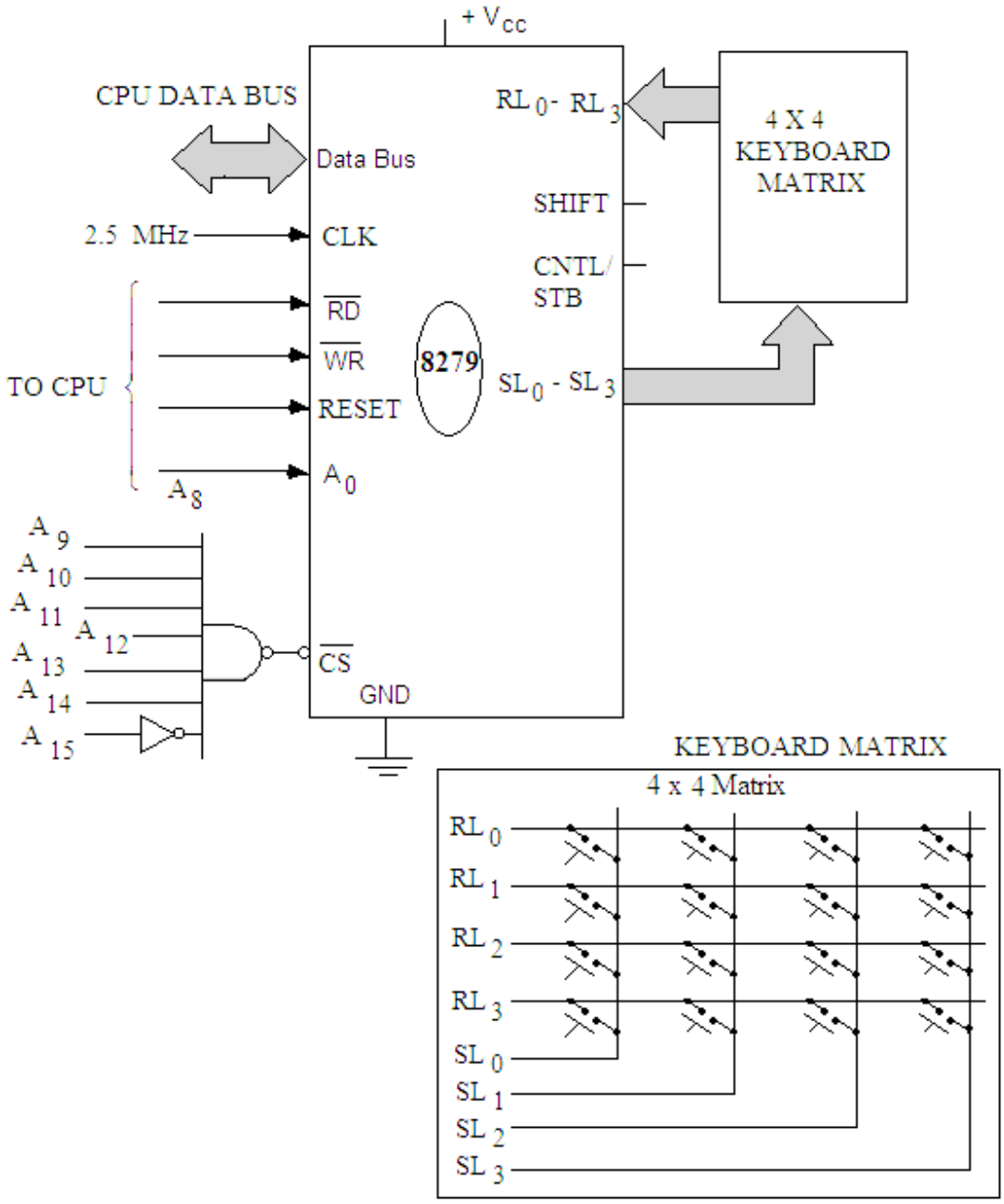


Fig. 10.30

2. The frequency of the clock is to be set around 100 KHz from 2.5 MHz. This frequency is to be divided by 25 generating the program clock word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	P	P	P	P	P	
0	0	1	1	1	0	0	1	= 39 H

11001 for PPPPP is 25.

3. Generate clear command word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	0	0	0	0	1	1	= C3 H

It will clear all.

4. A read FIFO RAM command is written into the command register as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	0	A_I	X	A	A	A	
0	1	0	0	0	0	0	0	= 40 H

(A_I and A A A are irrelevant in sensor matrix mode).

5. Check the status word if the key is pressed.

The program for the same is written as:

Program:

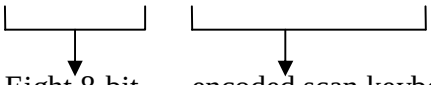
Label	**Mnemonics**	**0perand**	**Comments**
	MVI A,	03 H	; Decoded scan, N key roll-over mode.
	OUT	81 H	; Keyboard display code is written in command register.
	MVI A,	39 H	; Program clock
	OUT	81 H	; is set to 100 KHz.
	MVI A,	C3 H	; Clear FIFO.
	OUT	81 H	; Clear command word is loaded to command register.
LOOP	IN	81 H	; Read the status word.
	ANI	07 H	; Check if key pressed.
	JZ	LOOP	; If not then read the status word again till the key is pressed.
	MVI A,	40 H	; Load FIFO RAM command word
	OUT	81 H	; in command word register.
	IN	80 H	; Read the data register to take the key code.
	STA	2500 H	; Store the Hex code for the key pressed in memory location 2500 H.
	HLT		; Stop processing.

Example 10.2. *An 8279 is to be initialized with the following requirements:*
Keyboard encoded scan mode with N key rollover.
External clock frequency is 2 MHz.
Control port address is FF H and
Data port address is FE H.
The IRQ line of the 8279 is connected to the RST 7.5 interrupt of 8085A. Write an interrupt service routine to store the hex code of the key pressed in 2501 memory location.

Solution. The initialization steps for providing the software are given below:

1. Send Keyboard/display command word to command register. Say:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	1	0	= 02 H



Eight 8-bit Character display - left entry.

encoded scan keyboard - N Key rollover.

2. The frequency of the clock is to be set around 100 KHz from 2 MHz. This frequency is to be divided by 20 generating the program clock word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	P	P	P	P	P	
0	0	1	1	0	1	0	0	= 34 H

10100 for PPPPP is 20.

4. Generate clear command word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	0	0	0	0	1	1	= C3 H

It will clear all.

4. A read FIFO RAM command is written into the command register as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	0	A_I	X	A	A	A	
0	1	0	0	0	0	0	0	= 40 H

(A_I and A A A are irrelevant in sensor matrix mode).

5. Check the status word if the key is pressed.

Program:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	02 H	; Encoded scan, N key roll-over mode.
	OUT	FF H	; Keyboard display code is written in command register.
	MVI A,	34 H	; Program clock
	OUT	FF H	; is set to 100 KHz.
	MVI A,	C3 H	; Clear FIFO.
	OUT	FF H	; Clear command word is loaded to command register.
	EI		; Enable interrupts.
	MVI A,	0B H	; Enable RST 7.5.
	SIM		; Set interrupt mask.

In this case when key code is available in FIFO RAM the IRQ line becomes high and enables RST 7.5 interrupt. The program will jump to the vector location 003C H of this interrupt, from where it jump to the service routine.

003C H	JMP	ISR	; Jump to service routine.

```
ISR     MVI  A,     40 H      ; Load FIFO RAM command word
        OUT         FF H      ; in command word register.
        IN          FE H      ; Read the data register to take the
                              key code.
        STA         2501 H    ; Store the Hex code for the key
                              pressed in memory location 2500
                              H.
        RET                   ; Return
```

Example 10.3. *An 8279 keyboard/display interface is used to drive eight 7-segmnet display in multiplexed mode. Write a program to initialize 8279 to display a message 'HELLO 07'.*

External clock frequency is 1.5 MHz.
Control port address is 19 H and
Data port address is 18 H.

Solution. The initialization steps for providing the software are given below:

1. Send Keyboard/display command word to command register. Say:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	0	0	= 00 H

Eight 8-bit Character display - left entry.

2. The frequency of the clock is to be set around 100 KHz from 1.5 MHz. This frequency is to be divided by 15 generating the program clock word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	P	P	P	P	P	
0	0	1	0	1	1	1	1	= 2F H

01111 for PPPPP is 15.

3. Generate clear command word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	0	1	0	0	1	1	= D3 H

It will clear all.

4. Display RAM command is written into the command register as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	A_I	A	A	A	A	
1	0	0	1	0	0	0	0	= 90 H

(A_I =1 for auto increment and 0000 for A A A A as address bits).

Program:

```
Label   Mnemonics   Operand     Comments
        MVI  A,     00 H      ; 8 bit character display mode.
        OUT         19 H      ; Keyboard display code is written in
                              command register.
```

```
        MVI  A,     2F H    ; Program clock
        OUT         19 H    ; is set to 100 KHz.
        MVI  A,     D3 H    ; Clear FIFO.
        OUT         19 H    ; Clear command word is loaded to
                            command register.
UP      IN          19 H    ; Read the status word.
        ANI         80 H    ; Check if display available.
        JNZ         UP      ; If not available, read again.
        LXI  H,     2100 H  ; Point to the data in address
                            location.
        MVI  C,     08      ; C is used counter for eight
                            characters of 'HELLO 07'.
        MVI  A,     90 H    ; Write display command
        OUT         19 H    ; The command word is loaded in
                            command register.
REP     MOV  A,     M       ; Write the segment code in
        OUT         18 H    ; display RAM.
        INX  H              ; Increment H-L pair to point the
                            next code.
        DCR  C              ; Decrement C.
        JNZ         REP     ; If C is not zero jump to REP.
        HLT                 ; Stop processing.
```

Enter hex codes for HELLO 07 in the memory locations starting at 2100 H.

Location	**Hex code for**
2100 H	H
2101 H	E
2102 H	L
2103 H	L
2104 H	O
2105 H	- (Blank)
2106 H	0
2107 H	7

Example 10.4. *An 8 X 8 keyboard matrix (64 keys) and 8 common cathode seven segment displays are interfaced with 8085 through 8279. Write a program to initialize 8279 to display a message 'DISPLAY' on pressing key "0".*

External clock frequency is 2.0 MHz.
Control port address is 05 H and
Data port address is 04 H.

Solution. The initialization steps for providing the software are given below:

1. Send Keyboard/display command word to command register. Say:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	0	0	= 00 H

Eight 8-bit Character display - left entry.

2. The frequency of the clock is to be set around 100 KHz from 2.0 MHz. This frequency is to be divided by 20 generating the program clock word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	P	P	P	P	P	
0	0	1	1	0	1	0	0	= 34 H

10100 for PPPPP is 20.

3. Generate clear command word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	0	1	0	0	1	1	= D3 H

It will clear all.

4. Display RAM command is written into the command register as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	A_I	A	A	A	A	
1	0	0	1	0	0	0	0	= 90 H

(A_I =1 for auto increment and 0000 for A A A A as address bits).

Program:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	00 H	; 8 bit character display mode.
	OUT	05 H	; Keyboard display code is written in command register.
	MVI A,	34 H	; Program clock
	OUT	05 H	; is set to 100 KHz.
	MVI A,	D3 H	; Clear FIFO.
	OUT	05 H	; Clear command word is loaded to command register.
UP	IN	05 H	; Read the status word.
	ANI	80 H	; Check if display available.
	JNZ	UP	; If not available, read again.
UP1	IN	05	; Read the status word again if the "0" is pressed.
	ANI	07 H	; Check for 0.
	JZ	UP1	; If zero, key "0" is not pressed repeat.
	MVI A,	40 H	; Load FIFO command word
	OUT	05 H	; in command word register.
REP	IN	04 H	; Read the data register to take the key code.

```
        CPI     00 H    ; Key "0" is pressed or not.
        JNZ     REP     ; If not repeat.
        LXI H,  2100 H  ; Point to the data in address location.
        MVI C,  07      ; C is used counter for eight characters of 'DISPLAY'.
        MVI A,  90 H    ; Write display command
        OUT     05 H    ; The command word is loaded in command register.
LOOP    MOV A,  M       ; Write the segment code in
        OUT     04 H    ; display RAM.
        INX H           ; Increment H-L pair to point the next code.
        DCR C           ; Decrement C.
        JNZ     LOOP    ; If C is not zero jump to REP.
        HLT             ; Stop processing.
```

Enter hex codes for DISPLAY in the memory locations starting at 2100 H.

Location	**Hex code for**
2100 H	D
2101 H	I
2102 H	S
2103 H	P
2104 H	L
2105 H	A
2106 H	Y

Example 10.5. *An 8279 keyboard/display interface is used to drive sixteen 7-segmnet display in multiplexed mode. Write a program to initialize 8279 to display a blinking message "CONGRATULATIONS".*

External clock frequency is 1.5 MHz.
Control port address is 19 H and
Data port address is 18 H.

Solution. The initialization steps for providing the software are given below:

1. Send Keyboard/display command word to command register. Say:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	1	0	0	0	= 08 H

Sixteen 8-bit Character display - left entry.

2. The frequency of the clock is to be set around 100 KHz from 1.5 MHz. This frequency is to be divided by 15 generating the program clock word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	P	P	P	P	P	
0	0	1	0	1	1	1	1	= 2F H

01111 for PPPPP is 15.

3. Generate clear command word as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	0	1	0	0	1	1	= D3 H

It will clear all.

4. Display RAM command is written into the command register as:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	A_I	A	A	A	A	
1	0	0	1	0	0	0	0	= 90 H

(A_I =1 for auto increment and 0000 for A A A A as address bits).

Program:

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	08 H	; 8 bit character display mode.
	OUT	19 H	; Keyboard display code is written in command register.
	MVI A,	2F H	; Program clock
	OUT	19 H	; is set to 100 KHz.
	MVI A,	D3 H	; Clear FIFO.
	OUT	19 H	; Clear command word is loaded to command register.
UP	IN	19 H	; Read the status word.
	ANI	80 H	; Check if display available.
	JNZ	UP	; If not available, read again.
LOOP	LXI H,	2100 H	; Point to the data in address location.
	MVI C,	0F	; C is used counter for 15 characters of 'CONGRATULATIONS'.
	MVI A,	90 H	; Write display command
	OUT	19 H	; The command word is loaded in command register.
REP	MOV A,	M	; Write the segment code in
	OUT	18 H	; display RAM.
	INX H		; Increment H-L pair to point the next code.
	DCR C		; Decrement C.
	JNZ	REP	; If C is not zero jump to REP.
	CALL	DELAY	; Introduce some delay using a Delay subroutine.
	CALL	CLEAR	; A subroutine to clear the display is used.
	CALL	DELAY	; Delay is introduce so that display is clear for some time.
	JUMP	LOOP	; Jump to repeat the display.

HLT ; Stop processing.

Enter hex codes for “CONGRATULATIONS” in the memory locations starting at 2100 H.

Location	**Hex code for**
2100 H	C
2101 H	O
2102 H	N
2103 H	G
2104 H	R
2105 H	A
2106 H	T
2107 H	U
2108 H	L
2109 H	A
210A H	T
210B H	I
210C H	O
210D H	N
210E H	S

PROBLEMS

1. Draw the function block diagram of Programmable Keyboard/Display interface 8279 and also discuss the function of each block.
2. Discuss three input modes of Keyboard scan of 8279.
3. Describe Scanned keyboard mode of 8279 with 2-key lockout and N-key rollover.
4. Discuss Scanned sensor matrix mode for 8279.
5. Discuss encoded mode of Keyboard scan of 8279.
6. Discuss decoded mode of Keyboard scan of 8279.
7. Describe left entry mode for display of 8279.
8. Name 8 command words of 8279 and discuss the command word format for the program clock.
9. Discuss the command word format for Read display RAM.
10. Discuss the command word format for keyboard/display mode set.
11. Discuss the command word format for Read FIFO/Sensor RAM.
12. Discuss the command word format for Display write Inhibit/blanking.
13. Discuss the command word format for End Interrupt/Error mode set.
14. Explain status register of 8279. How the status word can be read.
15. Discuss how 8279 can be interfaced with 8085A.
16. Mention various steps to be carried out for the initialization of 8279 with following specifications:
    Keyboard encoded scan mode with N key rollover.
    External clock frequency is 2 MHz.
    Control port address is 11 H and
    Data port address is 10 H.

17. An 8279 keyboard/display interface is used to drive sixteen 7-segmnet display in multiplexed mode. Write a program to initialize 8279 to display a blinking message "HAPPY BIRTH DAY".
    External clock frequency is 2.0 MHz.
    Control port address is 05 H and
    Data port address is 04 H.

18. An 8 x 8 keyboard matrix (64 keys) and 8 common cathode seven segment displays are interfaced with 8085 through 8279. Write a program to initialize 8279 to display a message 'PLEASE" on pressing key "0".
    External clock frequency is 2.5 MHz.
    Control port address is FF H and
    Data port address is FE H.

19. An 8279 keyboard/display interface is used to drive sixteen 7-segmnet display in multiplexed mode. Write a program to initialize 8279 to display a message 'BEST WISHES'.
    External clock frequency is 2.5 MHz.
    Control port address is 01 H and
    Data port address is 00 H.

20. An 8 x 8 keyboard matrix is to be interfaced with 8085 microprocessor using 8279. Give the necessary hardware for the same. Also give the software to enter the hex code of the key pressed and store this code to memory location 2500 H.
    Given frequency of the clock connected to the CLK terminal of 8279 is 2.5 MHz. Assume Decoded scan mode with N-key-roll over. Let control port address is 19 H and data port address is 18 H.

11

Programmable Interrupt Controller: 8259

This chapter will confine to the detailed discussion on Programmable Interrupt Controller (PIC) – 8259. This device is also called priority interrupt controller. It is designed to work with Intel 8080A, 8085A, 8086 and 8088 microprocessors. It works as an overall manager in an interrupt driven system environment. This device can handle 8 external interrupts and the starting address of the interrupts service routine can be vectored to any location in the memory map unlike the software and hardware interrupts which point to the predetermined starting address. The 8259 can be set to accept level triggered or edge triggered interrupts. The interrupt can be expanded to 64 interrupt inputs by cascading many 8259 device.

11.1 PROGRAMMABLE INTERRUPT CONTROLLER 8259

It n an earlier chapter it has been discussed that the 8085A has four hardware interrupt terminals namely TRAP, RST 5.5, RST 6.5 and RST 7.5. When the I/O device sends an interrupt signal to the CPU, the CPU completes the current instruction and branches to the service routine of the interrupt. After the execution of the ISR, it returns to the main program. In addition to these inputs, the CPU can also be interrupted through its *INTR* input. When an interrupt input signal *INTR* is send to the microprocessor, the CPU then send an interrupt acknowledge signal $\overline{INTA}$ to the external device. In response to this acknowledge signal the op code of the CALL instruction is placed on the data bus. The CPU reads the op code and sends another $\overline{INTA}$ signal. This signal is used to place the low order eight bits address of the CALL instruction on to the data bus. After reading the low order address the CPU sends another $\overline{INTA}$ signal. It then places the high order eight bit address on to the data bus. The CPU then reads and executes the interrupt service routine available in that CALL address. The number of I/O devices that may be used in an interrupt driven environment may be made greater than four if an external device is used. The external device should be capable of accepting interrupt requests and generating unique CALL instructions for the different interrupt inputs. This device will be used to interrupt the microprocessor on the *INTR* line. A programmable interrupt controller (PIC) is such a device (ref. figure 11.1) which accepts interrupt requests from a number of I/O devices, resolves the priority of servicing the requests, and issues an interrupt on the *INTR* input of the 8085A as discussed above.

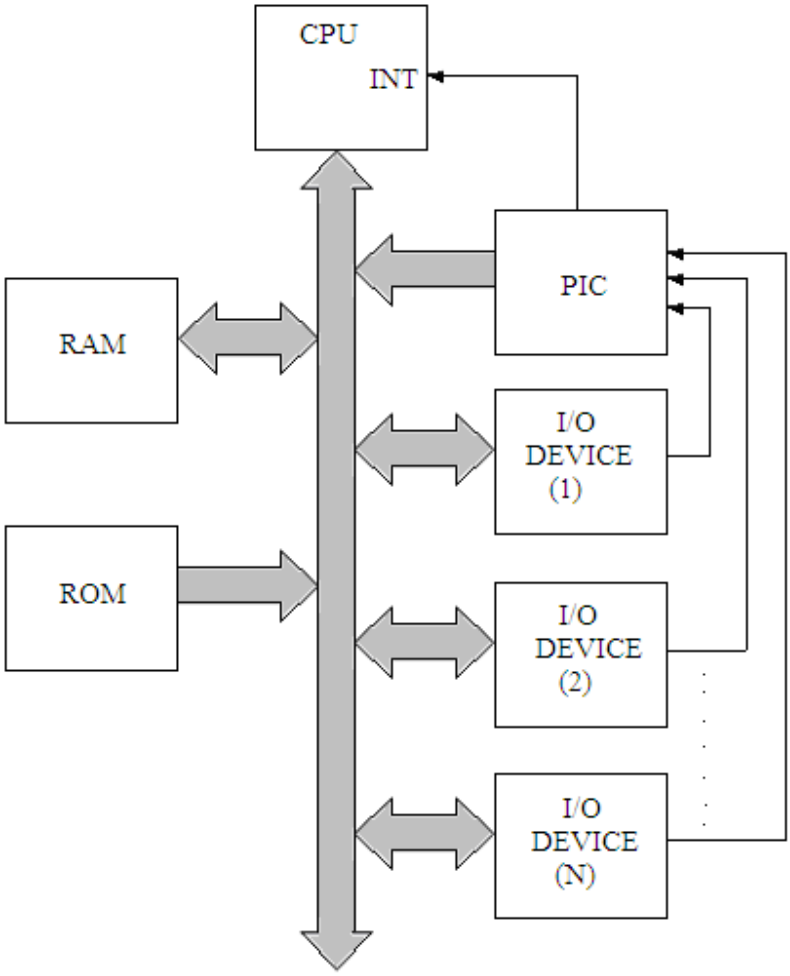


Fig. 11.1

11.2 BLOCK DIAGRAM OF 8259

The 8259 is a programmable interrupt controller which uses NMOS technology. It is available in 28 pin plastic dual in line package (DIP). It requires one power supply of +5 V but does not require any internal or external clock. A single PIC can accept interrupt requests from eight I/O devices, resolve priority among them and communicate to the microprocessor. Interrupt requests from all the I/O devices can individually be masked and a suitable priority mode can be selected by programming. Built in expandability has also been provided to cascade 9 such Programmable Interrupt Controller devices (8259s) to serve up to 64 I/O devices. Figure 11.2 shows the pin diagram of this IC.

The description of the pins of 8259 programmable interrupt controller is given as:

Pin No. 1: This pin is chip select terminal ($\overline{CS}$). It is active low. When a low signal is applied to this terminal, the device is chosen to work.

Pin No. 2: This is write input terminal ($\overline{WR}$), which is also active low. A low to this input enables the CPU to write Initialization Control Word (ICW) and Operation Command Word (OCW) to the 8259A.

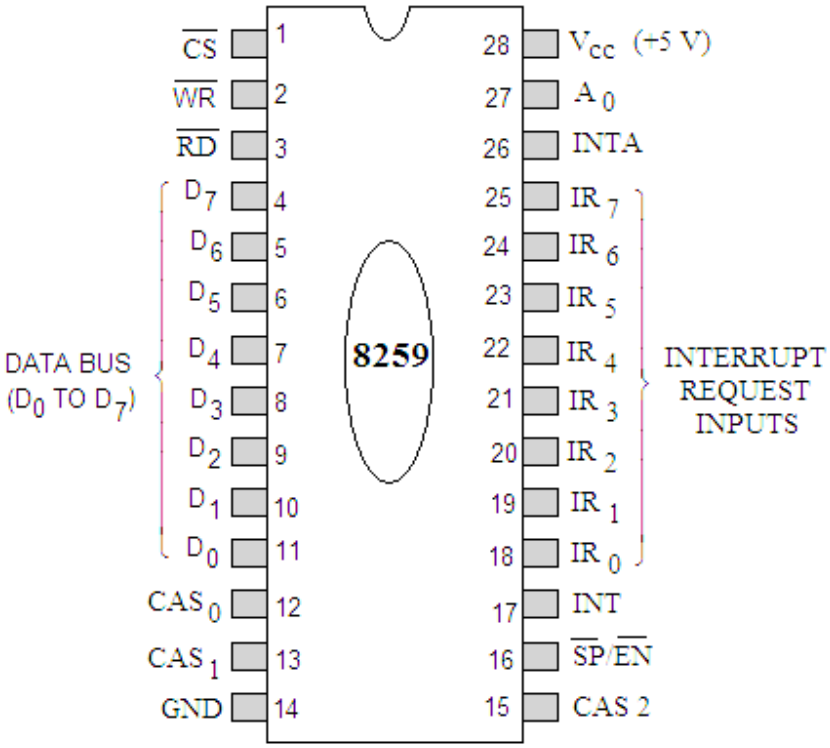


Fig. 11.2

Pin No. 3: This is read input terminal ($\overline{RD}$). A low to this pin enables the 8259A to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR) or the BCD of the interrupt level on the data bus.

Pin Nos. 4-11: These pins (D_0 to D_7) form the bidirectional data bus to be connected to the data bus of the system.

Pin Nos. 12, 13 &15: These pins (CS_0 to CS_2) are known as cascade lines. These lines are used to cascade a number of 8259A.

Pin No. 14: This is a common ground terminal to be connected to the common terminal of the system or the supply.

Pin No. 16: This is slave program/enable buffer ($\overline{SP}/\overline{EN}$). As already discussed more than one 8259A can be used with the system to expand the priority interrupt schemes up to 64 levels. In such cases one 8259A acts as the master and the others act as slaves. A high on this pin designates the 8259A as the master and a low to this pin designates as slave.

Pin No. 17: This is an interrupt output terminal (INT) to be directly connected to the interrupt terminal (INTR) of the CPU. When an interrupt signals is applied to the any of the interrupt request terminals of the 8259, an INT signal is generated for the CPU.

Pin Nos. 18 to 25: These are 8 interrupt request inputs (IR_0 to IR_7).

Pin No. 26: This is an interrupt acknowledge input signal ($\overline{INTA}$), connected to the $\overline{INTA}$ terminal of the CPU. A low acknowledgement signal is sent by the CPU to the 8259, whenever CPU receives an interrupt signal.

Pin No.27: This pin A_0 is the input signal used in conjunction with the signals $\overline{WR}$ and $\overline{RD}$ to write the commands into the various status registers of the device. This terminal can directly be connected to one of the address lines.

Pin No. 28: This is supply pin ($+V_{CC}$) connected to the positive 5 volts.

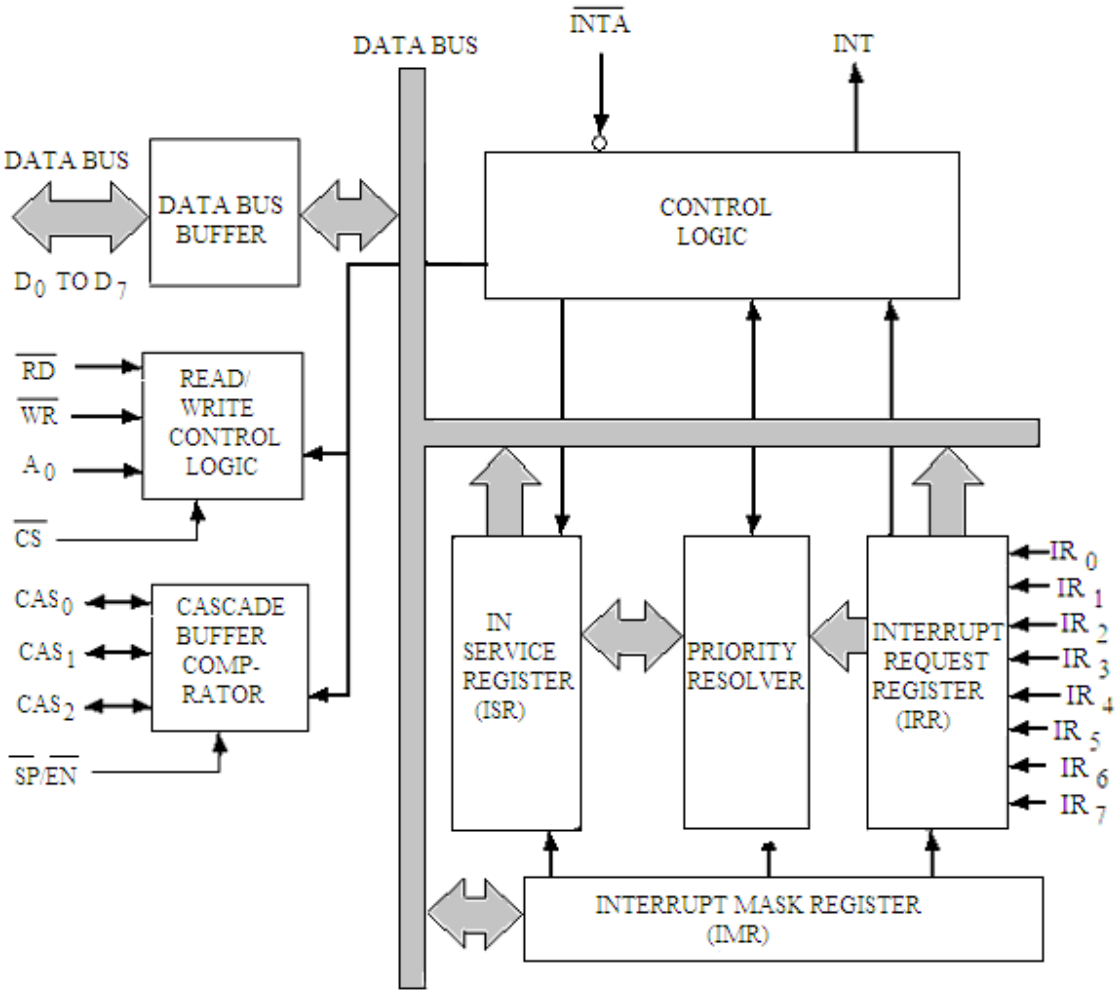


Fig. 11.3

Figure 11.3 shows the internal block diagram of the 8259A. The 8259A contains the following four sections:

1. Interrupt and Control Logic Section
2. Data Bus Buffer
3. Read/Write Control Logic Section
4. Cascade Buffer/Comparator Section

1. Interrupt and Control Logic Section

This section contains the following five sections:

- Interrupt Request Register (IRR)
- In-Service Register (ISR)
- Priority Resolver
- Interrupt Mask Register (IMR)
- Control Logic Section

Interrupt Request Register (IRR)

The interrupt request register (IRR) is used to store all the interrupt levels which are requesting services. It has 8-interrupt lines (IR_0 to IR_7). When any of these lines become high, the corresponding mask bit is checked and if it enabled, then the corresponding bit in the interrupt request register (IRR) is set.

In-Service Register (ISR)

The in-service register (ISR) is used to store information of all the interrupt levels which are currently being serviced.

Priority Resolver

This block of Interrupt and Control Logic Section determines the priorities of the bits set in the IRR. This block determines the priorities as dictated by the priority mode set by the Operation Command Words (OCWs). The bit corresponding to the highest priority interrupt is set in the ISR during the $\overline{INTA}$ input. It is a priority interrupt controller, this means, even while executing an interrupt, it will accept and service a higher priority; but it will reject a lower priority interrupt. The priority resolver does the job of judging whether to allow another interrupt to be executed in the middle of executing one interrupt service routine.

Interrupt Mask Register (IMR)

The interrupt mask register (IMR) stores the bits of the interrupt lines to be masked. The IMR operates on the ISR. In fact this register can be programmed by an operation command word (OCW) to store the bits of the interrupt lines to be masked. An interrupt which is masked by software will not be recognized and serviced even if it sets the corresponding bit in the IRR.

Control Logic Section

After the interrupt request priorities are resolved by the priority resolver, this block control logic sends an interrupt signal through its *INT* signal to the CPU. This terminal *INT* is connected to the *INTR* terminal of the microprocessor. The microprocessor responds to this request by putting an interrupt acknowledge signal $\overline{INTA}$ to the input terminal $\overline{INTA}$ of the 8259A. The PIC then places the op code of CALL instruction on the data bus. This is read by CPU, it places two additional $\overline{INTA}$ signals on the data bus. When the CPU receives the $\overline{INTA}$ signal out of these two additional $\overline{INTA}$ signals, it places the low order byte of the CALL address on the data bus. After receiving the last $\overline{INTA}$ signal, it places the high order byte of the CALL address on the data bus. The CALL address is the vector memory location for t he interrupt; this address is placed in the control register during the initialization of 8259.

2. Data Bus Buffer

This is a three state bidirectional 8-bit data bus buffer is used to interface the 8259A to the system data bus. The control words and status information are transferred through this data bus buffer.

3. Read/Write Control Logic Section

The function of this Read/Write Control Logic section is to accept the commands from the CPU. It contains the initialization command word (ICW) registers and the operation command word (OCW) registers which store the various control formats for device operation. This section also accepts Read commands from the CPU to allow the CPU to read status words. The four pins are connected to this block whose functions are given below:

$\overline{CS}$ (Chip Select)

This pin is chip select terminal, which is active low. A low signal to this terminal selects this device to work.

$\overline{WR}$ (Write)

This is write input terminal, which is also active low. A low to this input enables the CPU to write Initialization Control Word (ICW) and Operation Command Word (OCW) to the 8259A.

$\overline{RD}$ (Read)

A low to this pin enables the 8259A to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR) or the BCD of the interrupt level on the data bus.

A_0

This is the input signal used in conjunction with the signals $\overline{WR}$ and $\overline{RD}$ to write the commands into the various status registers of the device. This terminal can directly be connected to one of the address lines.

4. Cascade Buffer/Comparator Section

It is well known that more than one 8259A can be used with the system to expand the priority interrupt schemes up to 64 levels. In such cases one 8259A acts as the master and the others act as slaves. The necessary control signals for cascade operations are generated with this block. A high on the Slave Program/Enable Buffer ($\overline{SP}/\overline{EN}$) pin designates the 8259A as the master and a low to this pin designates as slave. The 8259A

can be set as master or a slave by the $\overline{SP}/\overline{EN}$ pin in the non-buffer mode, or by software in buffer mode of operation. In the non-buffer mode $\overline{SP}/\overline{EN}$ pin is used as an output to enable the data bus buffer of the system. In addition to $\overline{SP}/\overline{EN}$ pin, there are three more pins (CAS_0 – CAS_2) associated with the cascade buffer/comparator section, whose functions are described below:

For a master, these pins (CAS_0 – CAS_2) are outputs, and for slave these are input pins. When 8259 is a master, the CALL op code is generated by the master in response to first $\overline{INTA}$ signal. The address for vector locations will be released by the slave 8259. Every 8259 has the identification code of three bits to CAS_0 – CAS_2 lines so that one out of eight possible slave may be selected by the master. Basically, the slave 8259s accept the identification signals as inputs and compare the code put out by the master with code assigned to them during initialization. The slave thus selected then puts out the address of the interrupt service routine during the two additional $\overline{INTA}$ signals sent by CPU.

11.3 INTERFACING OF 8259A WITH 8085A

Figure 11.4 shows the interfacing connection of 8259A with 8085A microprocessor in I/O mapped I/O mode. The output of address decoder is connected to

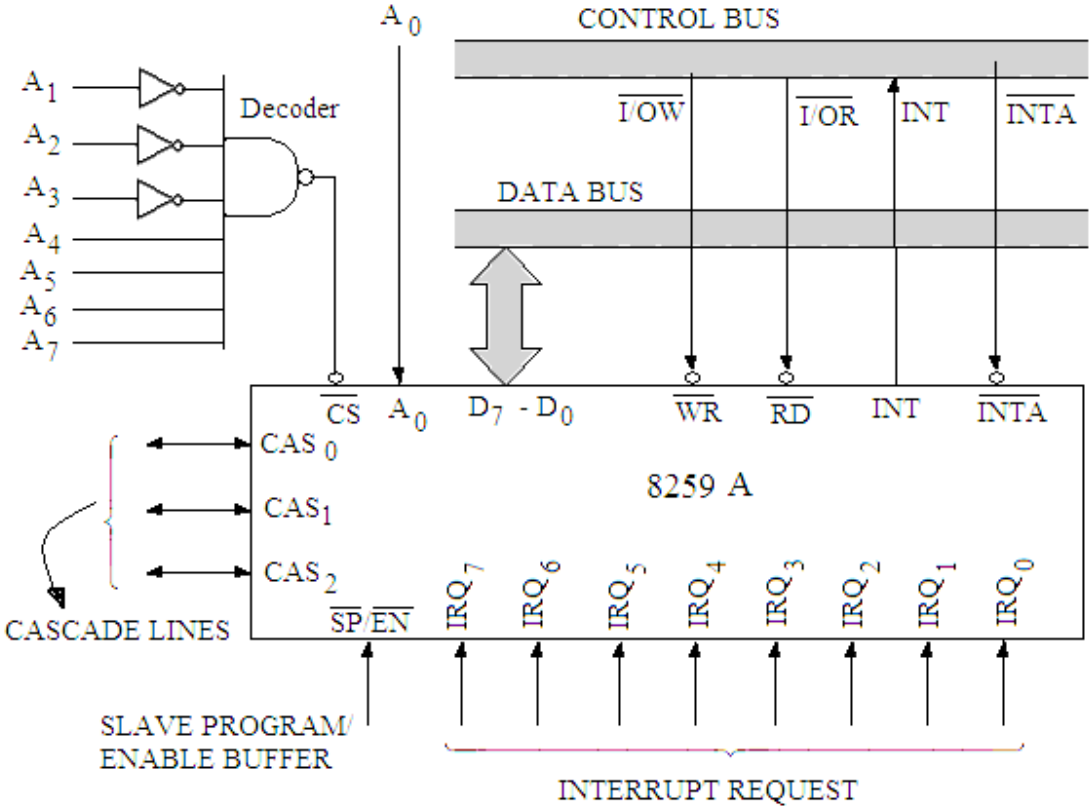


Fig. 11.4

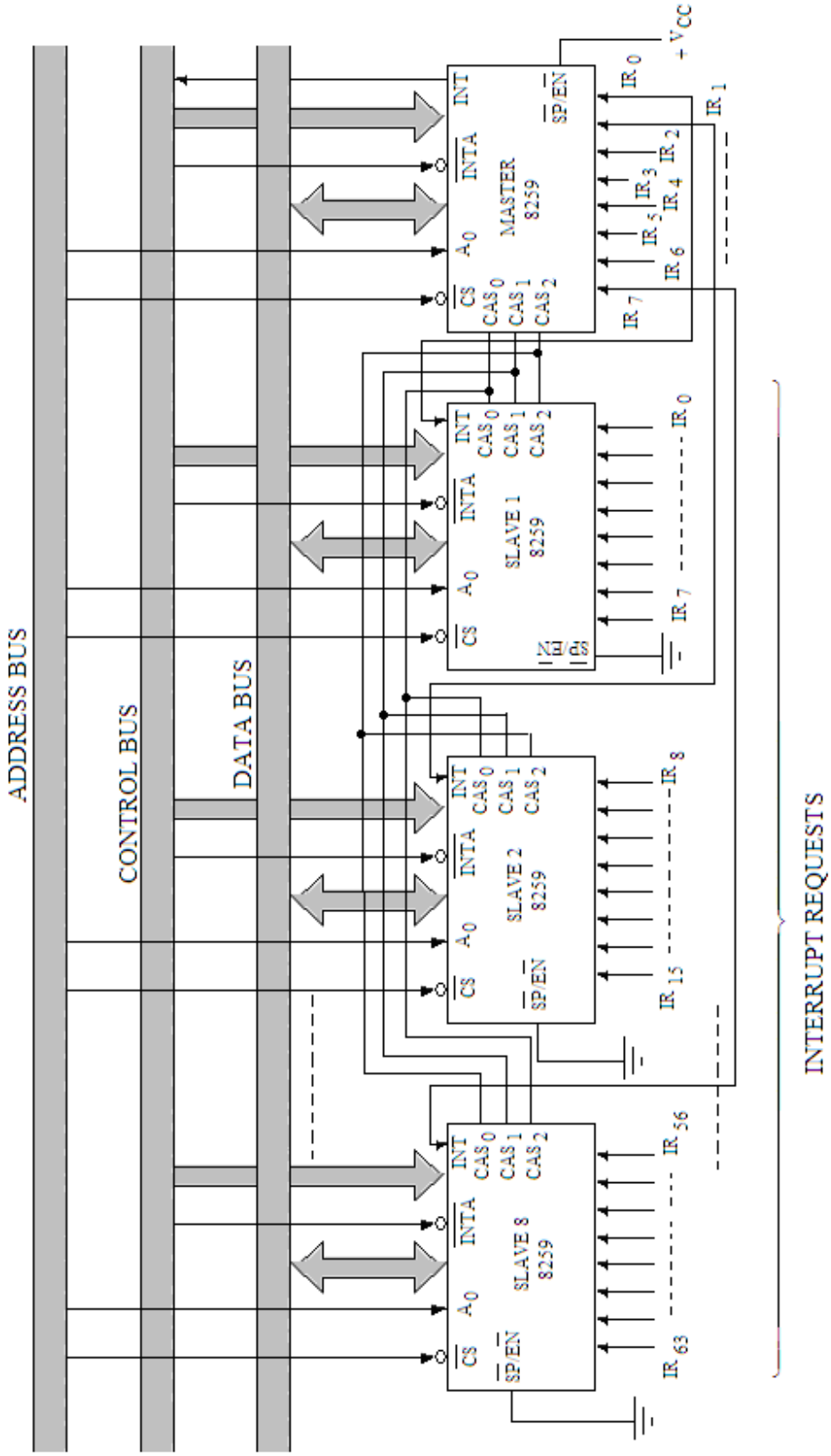

ADDRESS BUS
CONTROL BUS
DATA BUS
INTERRUPT REQUESTS
MASTER 8259
SLAVE 1 8259
SLAVE 2 8259
SLAVE 8 8259
INT
INTA
A0
CS
CAS0
CAS1
CAS2
SP/EN
+VCC
IR0
IR1
IR2
IR3
IR4
IR5
IR6
IR7
IR8
IR15
IR56
IR63

Fig.11.5

the $\overline{CS}$ input of the 8259A. The A_0 line of the IC is directly connected to the address lines of the system. The $\overline{RD}$ and $\overline{WR}$ signals of the IC are connected to $\overline{I/OR}$ and $\overline{I/OW}$ terminals respectively. Further, *INT* terminal of the 8259 is connected to the *INTR* terminal of the processor. The $\overline{INTA}$ output of the processor is connected to the $\overline{INTA}$ terminal of the processor. The terminal $\overline{SP}/\overline{EN}$ is connected to +5 V supply since only one 8259 is used in the system. If many 8259s are cascaded, then this terminal is connected to the ground terminal. The cascaded pins (CAS_0 – CAS_2) are left open. The eight interrupt request pins IR_0-IR_7 are available for the I/O devices to send the interrupt signals. If any of these interrupt request pins are not used, they may be connected to the ground so that the noise pulse can not create any interrupt.

The cascading of many 8259s with the system is shown in figure 11.5. The $\overline{SP}/\overline{EN}$ terminal of the master PIC is connected to + VCC, whereas $\overline{SP}/\overline{EN}$ terminals of all the slave PICs are connected to ground. The *INT* output of each slave PICs are connected to one of the interrupt lines (IR_0-IR_7) of the master 8259 i.e. 8 INT terminals of 8 slave 8259s are connected to 8 interrupt request lines (IR_0-IR_7). When interrupt request line of a slave is activated, the slave PIC in turn activates one of the interrupt request lines of the master 8259, which in turn interrupts the microprocessor. After the $\overline{INTA}$ is received from the microprocessor, the master enables the corresponding CAS_0 – CAS_2 lines to release the vector address on the data bus in the next two $\overline{INTA}$ signals.

The output of the address decoder is connected to the $\overline{CS}$ of the 8259 PIC as shown in figure 11.4. It communicates with the CPU with the bidirectional bus. From this figure 11.4, it is clear that two I/O port addresses F0 H and F1 H are chosen for the 8259 with $A_0 = 0$ and $A_0 = 1$. It may be noted here that each 8259A (interfaced with 8085A) has its own I/O address which is defined by the address decoder.

11.4 VECTORING DATA FORMATS FOR 8259

The eight interrupt levels generate CALLs to eight equally spaced locations in the memory. Through the programming, these locations can be spaced at an interval of 4 or 8 locations. The format of a byte released in response to the first $\overline{INTA}$ for any IR levels is shown in figure 11.6.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	0	0	1	1	0	1	CALL CODE = (CD H)

Fig. 11.6

It is the op code for the CALL instruction (CD H). The format of the byte in response to the second $\overline{INTA}$ for different IR level is shown in figure 11.7, the details of which are given in table 11.1(a) and (b). These tables show for both 4 and 8 interval spacing. For spacing interval of four, the 8259 automatically inserts the bits D_0 – D_4 and D_5 – D_7 as specified while programming the 8259 through ICW 1 (first initialization command word, which will be discussed in the next section). For this interval bits D_4-D_2 specify the interrupt's number and the rest two bits D_1-D_0 have been set to 00. However,

for spacing interval of eight, the 8259 automatically inserts the bits $D_0 - D_5$ and $D_6 - D_7$ as specified while programming the 8259 through ICW 1. For the interval of 8, the bits D_5-D_3 specify the interrupt's number and the rest two bits D_2-D_0 have been set to 000.

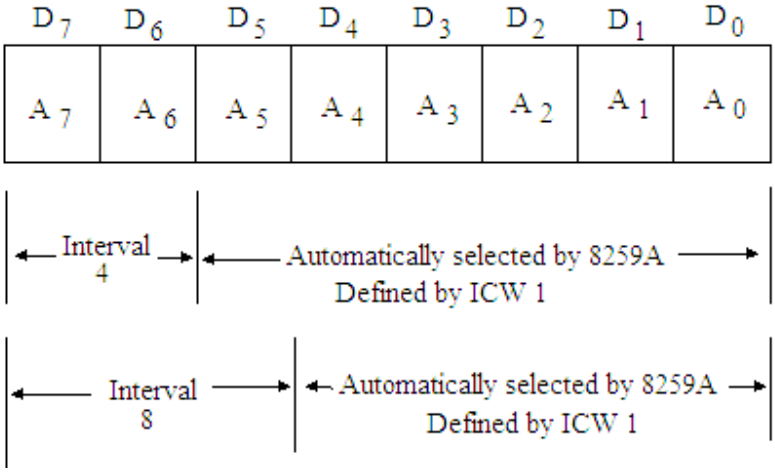


Fig. 11.7

Table 11.1(a)

IR	INTERVAL = 4							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
7	A_7	A_6	A_5	1	1	1	0	0
6	A_7	A_6	A_5	1	1	0	0	0
5	A_7	A_6	A_5	1	0	1	0	0
4	A_7	A_6	A_5	1	0	0	0	0
3	A_7	A_6	A_5	0	1	1	0	0
2	A_7	A_6	A_5	0	1	0	0	0
1	A_7	A_6	A_5	0	0	1	0	0
0	A_7	A_6	A_5	0	0	0	0	0

Table 11.1(b)

IR	INTERVAL = 8							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
7	A_7	A_6	1	1	1	0	0	0
6	A_7	A_6	1	1	0	0	0	0
5	A_7	A_6	1	0	1	0	0	0

4	A_7	A_6	1	0	0	0	0	0
3	A_7	A_6	0	1	1	0	0	0
2	A_7	A_6	0	1	0	0	0	0
1	A_7	A_6	0	0	1	0	0	0
0	A_7	A_6	0	0	0	0	0	0

From this table it is clear that the interrupt requests IR_2 and IR_4 for an interval of eight is given by:

	D_5	D_4	D_3	D_2	D_1	D_0
IR2 for an interval of 8 is	0	1	0	0	0	0
IR 4 for an interval of 8 is	1	0	0	0	0	0

Similarly these interrupt requests for an interval of 4 is:

	D_5	D_4	D_3	D_2	D_1	D_0
IR2 for an interval of 4 is		0	1	0	0	0
IR 4 for an interval of 4 is		1	0	0	0	0

Figure 11.8 shows the format of the byte released by the 8259A in response to the third $\overline{INTA}$. The bits D_7-D_0 represent the high order byte of the interrupt vector address (A_8-A_{15}) of all the interrupts; which are specified in ICW 2.

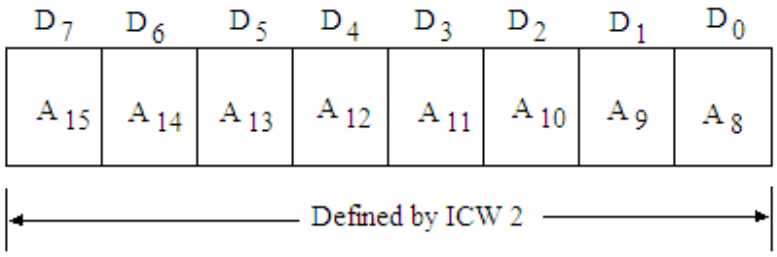


Fig. 11.8

It may be noted here that the 8259A does not allow each interrupt input to have its own independent address. Instead, a base address can be programmed with four or eight interval spacing as discussed above. For example, if the base address is 0100 H and the interval spacing is four, then eight restart locations will exist as shown in table 11.2. Normally, instructions to jump to the appropriate interrupt service routine would be stored in these locations and the 32 bytes of memory from 0100 H to 011F H referred to as jump table.

Table 11.2

Interrupt request	Restart address
IR0	0100 H
IR1	0104 H
IR2	0108 H
IR3	010C H
IR4	0110 H
IR5	0114 H
IR6	0118 H
IR7	011C H

Example 11.1. *For initialization of 8259 the address space available is from E010 H to E040 H. What should be the jump table, if interval spacing between each restart interrupt is four?*

Solution. The address space available for the restart instruction is from E010 H to E040 H, so the restart address for the first interrupt request will start from E020 H, as the five bits of the starting address must be 00000 (ref. Table 11.1a). The jump table is therefore given in table 11.3.

Table 11.3

Interrupt request	Restart address
IR0	E020 H
IR1	E024 H
IR2	E028 H
IR3	E02C H
IR4	E030 H
IR5	E034 H
IR6	E038 H
IR7	E03C H

11.5 INITIALIZATION OF 8259

There are two types of command words in 8259A. These are:

- Initialization Command Words (ICWs)
- Operation Command Words (OCWs)

11.6 INITIALIZATION COMMAND WORDS (ICWs)

There are four command words ICW 1, ICW 2, ICW 3 and ICW 4. The ICW 1 and ICW 2 commands are essential for any 8259A system. However, ICW 3 and ICW 4 commands are optional. The ICW 3 is used if the system has any slave 8259A. The ICW 4 command is used for special operations like special fully nested mode. The microprocessor programs the 8259A by loading the initialization command word (ICWs) and operation command words (OCWs). Each 8259A in the system is first initialized by loading a set of these ICWs in a sequence. The OCWs can be loaded any time after initialization.

Initialization Command Word -1 (ICW 1)

The format for first initialization command word (ICW 1) is shown in figure 11.9. When a byte is sent to the 8259A with $A_0 = 0$, and $D_4 = 1$, it is interpreted as initialization

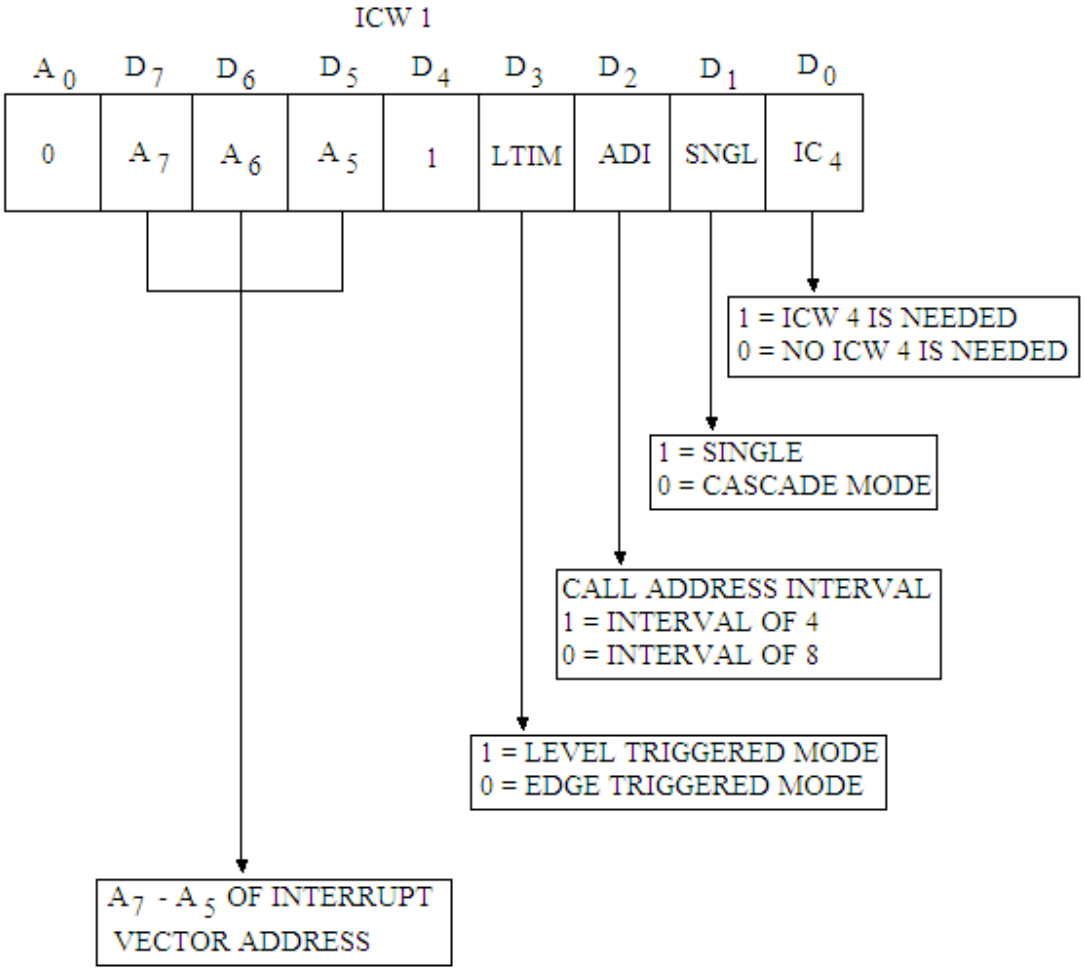


Fig. 11.9

command word-1 (ICW 1). The interpretations of the bits of ICW 1 command word are given below:

Bit D_0 This bit gives the information if the command word ICW 4 is needed.

If $D_0 = 1$, ICW 4 is needed.

If $D_0 = 0$, ICW 4 is not needed i.e. all functions selected by ICW 4 are cleared. This sets the 8259A in the 8085A mode without AEOI (automatic End of Interrupt) and non-buffered operation; and no ICW 4 would be sent with the initialization.

Bit D_1 This bit gives the information if the 8259A is to be used in single or cascaded mode.
If $D_1 = 1$, single 8259A is used and no slave 8259A is used.
If $D_1 = 0$, 8259A is used in cascaded mode.

Bit D_2 This bit gives the information about vector address**.**

If D_2 = 1, Interval for vector address is 4.

If D_2 = 0, Interval for vector address is 8.

Bit D_3 The bit D_3 gives the information about interrupt request inputs whether it is edge triggered or level triggered.
If D_3 = 1, Interrupt request input is level triggered.
If D_3 = 0, Interrupt request input is edge triggered.

Bit D_4 This bit is 1 for ICW 1, as discussed earlier.

Bits D_5 to D_7 These bits give the interrupt vector address (for 8085A only) i.e. these are address bits of the CALL instruction.

Initialization Command Word -2 (ICW 2)

A write command following ICW 1, with A0 = 0, is interpreted as ICW 2. The format for this command word is shown in figure 11.10. This gives information regarding Interrupt Vector Address. The bits D_7-D_0 represent the high order byte of the interrupt vector address (A_8-A_{15}) for 8085 microprocessor. However, T_7-T_3 represent the vector address for 8086/8088 microprocessors.

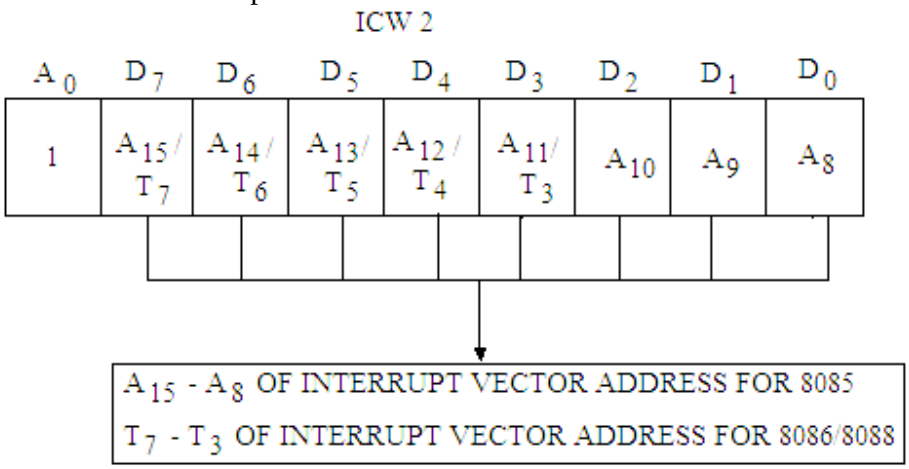


Fig. 11.10

Initialization Sequence of 8259A

Once ICW1 is loaded, the following initialization procedure is carried out internally:

- The edge sense circuit is reset. Since 8259A interrupts are edge sensitive, they are reset.
- IMR is cleared.
- IR7 input is assigned the lowest priority.
- Slave mode address is set to 7.
- Special mask mode is cleared and status read is set to IRR.
- If IC_4 = 0, all functions of ICW4 are set to zero. Master/slave bit is ICW 4 is used in the buffered mode only.

Example 11.2. *Initialize 8259A to interface eight level triggered inputs. The restart address of first interrupt request is E020 H. There is a spacing interval of four byte between each interrupt request. No cascading of 8259A is made. Let the port address is F0 H and F1 H for A_0 = 0 and A_0 = 1 respectively.*

Solution. The single 8259A is to be initialized, so initialization command words ICW 1 and ICW 2 are to be written as given below:

ICW 1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
A_7	A_6	A_5	1	LTM	ADI	SNGL	IC_4	
0	0	1	1	1	1	1	0	= 3E H

In ICW 1, D_0 = 0 as no ICW 4 is required; D_1 = 1 as single 8259 is used, D_2 = 1 as spacing interval of 4 is required, D_3 = 1 as these are level triggered inputs, D_7 D_6 D_5 are 001 as 20 H is the address of the first interrupt request (**001**0 0000).

ICW 2

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
1	1	1	0	0	0	0	0	= E0 H

ICW 2 gives the address E0 H (base address) 1110 0000.

Program:

```
MVI   A, 3E H      ; ICW 1 code is loaded to acc.
OUT   F0 H         ; F0 H is the port address for A0 = 0.
MVI   A, E0 H      ; Base address is given by ICW 2.
OUT   F1 H         ; ICW 2 port address for A0 = 1.
```

Initialization Command Word -3 (ICW 3)

As already discussed, ICW 1 and ICW 2 are compulsory command words in initialized sequence of 8259A where as ICW 3 and ICW 4 are optional. The command word ICW 3 is read only when cascading of 8259s is used, which is denoted by SNGL (Bit D_1) of ICW 1. The ICW 3 is further classified into following two modes:

Master Mode ICW 3

Slave Mode ICW 3.

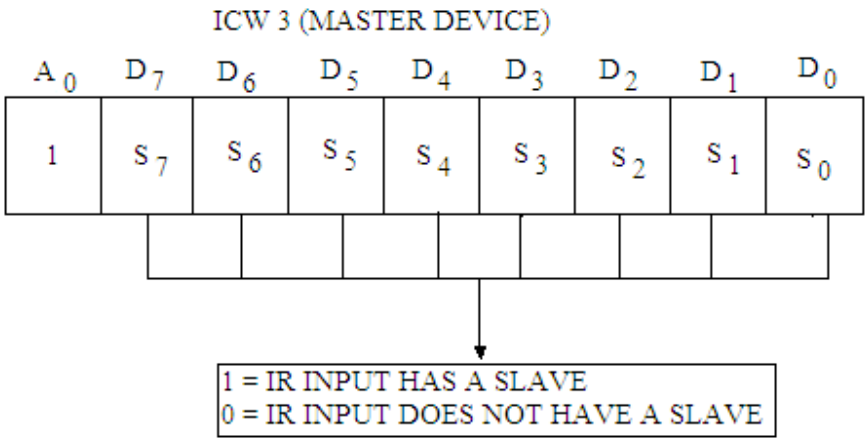
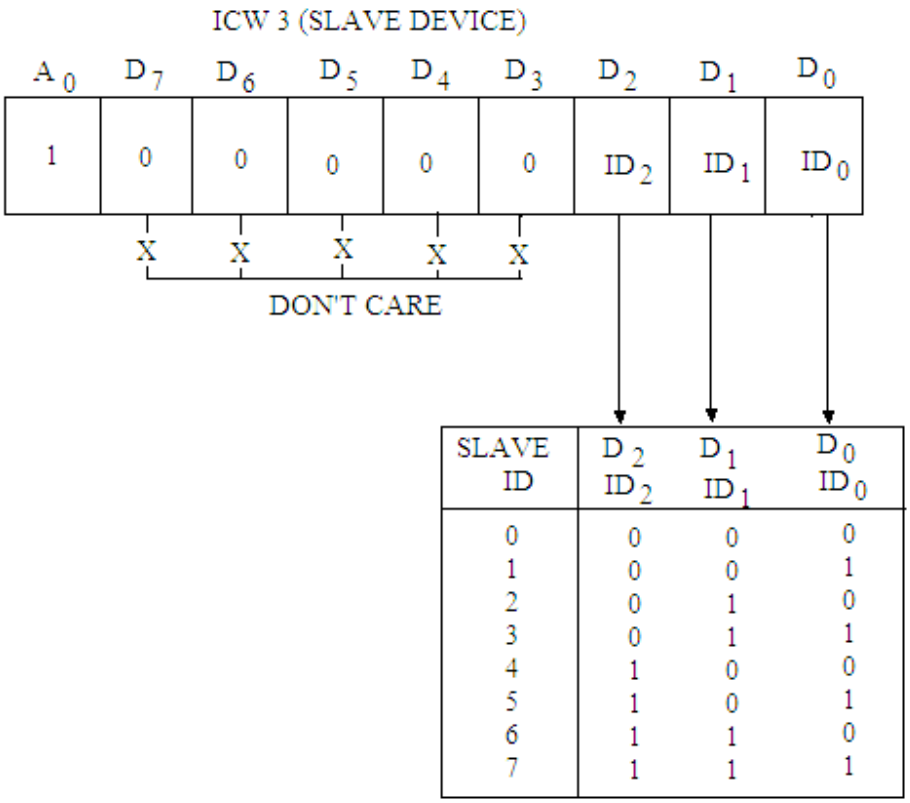


Fig. 11.11

The ICW 3 in Master and Slave modes are shown in figure 11.11 and 11.12 respectively. Each bit in ICW 3 of master mode is used to specify whether it has a slave 8259 attached to its corresponding interrupt request (IRQ) inputs. Suppose S_0 bit is 1. It indicates that 8259 is connected to interrupt request lines IR_0. The command word will be loaded to the 8 bit register and the functions of this register are:

- In the master mode (either when $\overline{SP}/\overline{EN}$ is high, or in buffered mode when M/S = 1 in ICW 4) a 1 set for each slave in the system. The master then will release byte 1 of the CALL sequence and will enable the corresponding slave to release byte 2 and byte 3 through the cascaded lines.
- In the slave mode (either when $\overline{SP}/\overline{EN}$ is low or if BUF = 1 and M/S = 0 in ICW 4) bits D_2-D_0 identify the slave. The slave compares its cascade inputs with these bits and if they are equal, bytes 2 and 3 of the CALL sequence are released by it on the data bus.



SLAVE ID	D_2 ID_2	D_1 ID_1	D_0 ID_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Fig. 11.12

Initialization Command Word -4 (ICW 4)

This command word is loaded only if IC_4 bit of the command word ICW 1 is 1. That is if IC_4 = 1 (of ICW 1) then command word -4 (ICW 4) is used otherwise it is

neglected. The format of this command word is shown in figure 11.13. The bits of this command word identify the following operations:

μPM — This specifies if 8080/8085A or 8086/8088 operation environment is used.
If μPM = 1, 8086/8088 microprocessor operation is selected.
If μPM = 0, then the 8080/8085A operation is selected.

AEOI — If this bit is 1, then the automatic end of interrupt mode is selected, otherwise normal end of interrupt mode.

M/S — If M/S = 1, then 8259A is master.
If M/s = 0, then 8259A is slave.
If BUF = 0, then M/S bit is neglected.

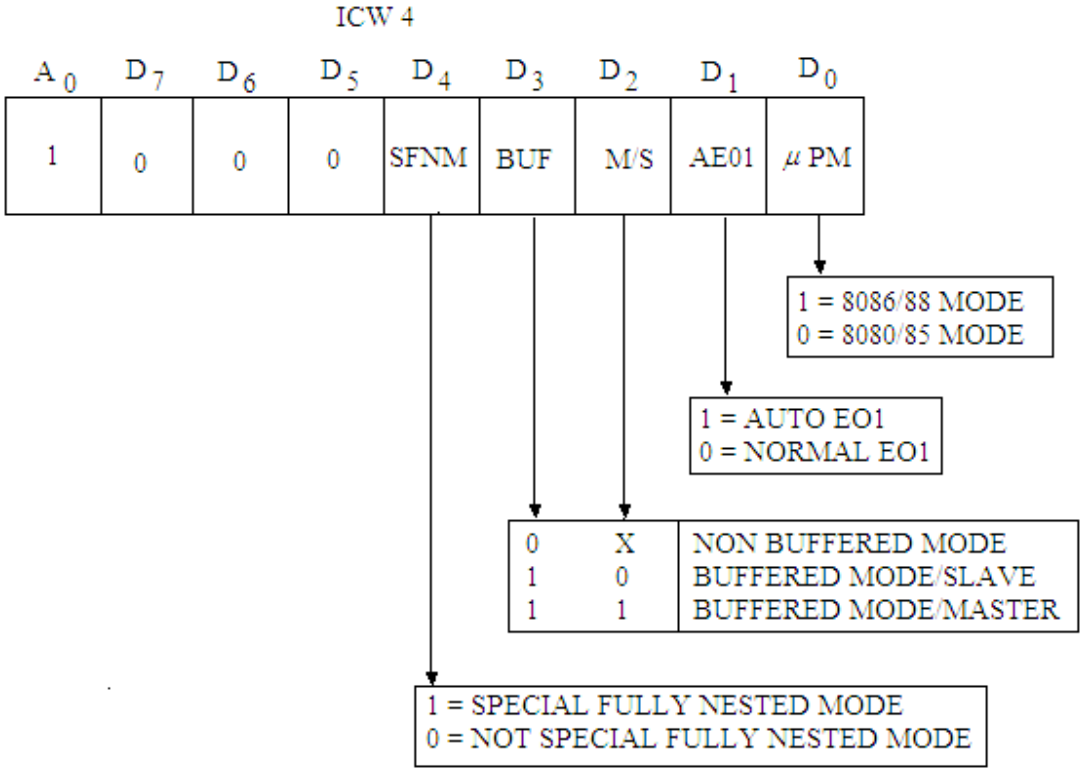


Fig. 11.13

BUF — If BUF = 1, then buffered mode is selected. In buffered mode $\overline{SP}/\overline{EN}$ acts as enable output and Master/Slave is determined by M/S bit of ISW 4.

SFNM — If SFNM = 1, then fully nested mode is selected.
If SFNM = 0, then fully nested mode is not selected.

Figure 11.14 shows the flow chart for the initialization of 8259A.

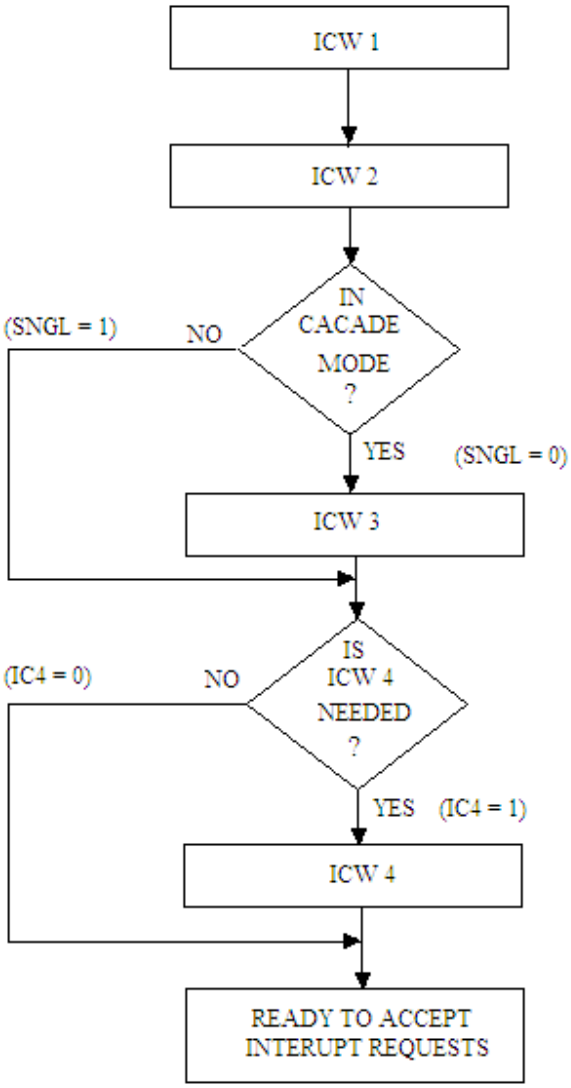


Fig. 11.14

11.7 OPERATION COMMAND WORDS (OCWs)

Once 8259A is initialized using initialization command words (ICWs), the 8259 is ready to accept interrupt requests. For this purpose, different operation command words (OCWs) are loaded in to operation command word registers. There are three operation command words (OCWs) namely: OCW 1, OCW 2 and OCW 3. Now the discussion will made of the operation command words.

Operation Command Word-1 (OCW 1)

This operation command word is written to mask or unmask an interrupt request input. The format for OCW 1 is shown in figure 11.15. For this word A_0 must be high. In this format it is clear if M = 0, then the particular interrupt request is to be unmasked or enabled, however, if M = 1, it is masked or inhibited.

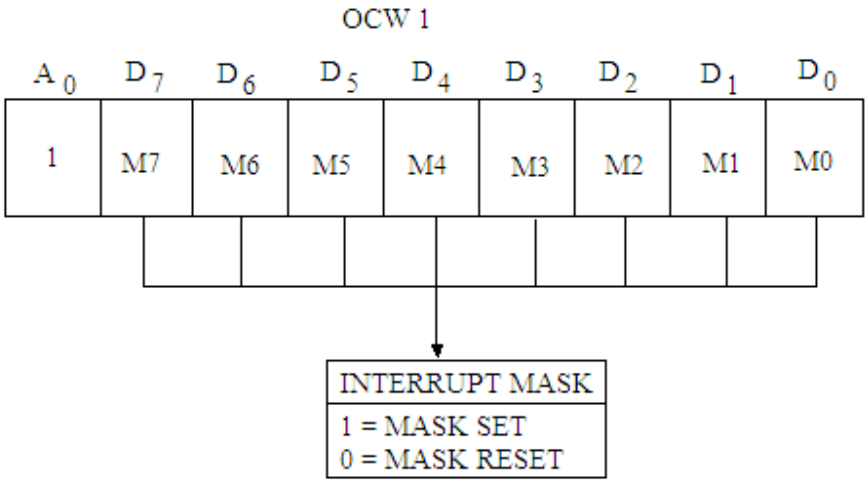


Fig. 11.15

For example, in order to enable the interrupts IR6, IR4 and IR1 then the bit pattern of this operation command word will be:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	0	1	0	0	1	0

Operation Command Word-2 (OCW 2)

A write command with $A_0 = 0$ and $D_4\ D_3 = 0\ 0$ is interpreted as OCW 2, which is used to control the Rotate and End of interrupt mode. This is also called End of Interrupt Command. The OCW 2 is mainly used to reset a bit in the in service register (ISR). The format for this operation command word is shown in figure 11.16.

R, SL and EOI – bits control the Rotate, End of interrupt modes and the combination of two. The bits L_2, L_1 and L_0 determine the interrupt level acted upon when the SL bit is active.

When an interrupt request is acknowledged, the 8259A determines the interrupt of highest priority and sets its corresponding bit in the ISR (in service register). The vector address corresponding to this interrupt is then put out. The bit in ISR remains set until an End of Interrupt (EOI) command through OCW 2 is issued by the CPU. At the end of the service routine, the corresponding bit in ISR is to be reset; otherwise other priority interrupts will not interrupt. The 8259A can respond to the interrupt signal of lower priority once the bit of the current IR bit in ISR is reset. The operation command word-2 (OCW 2) can issue the End of Interrupt (EOI) in the following three forms:

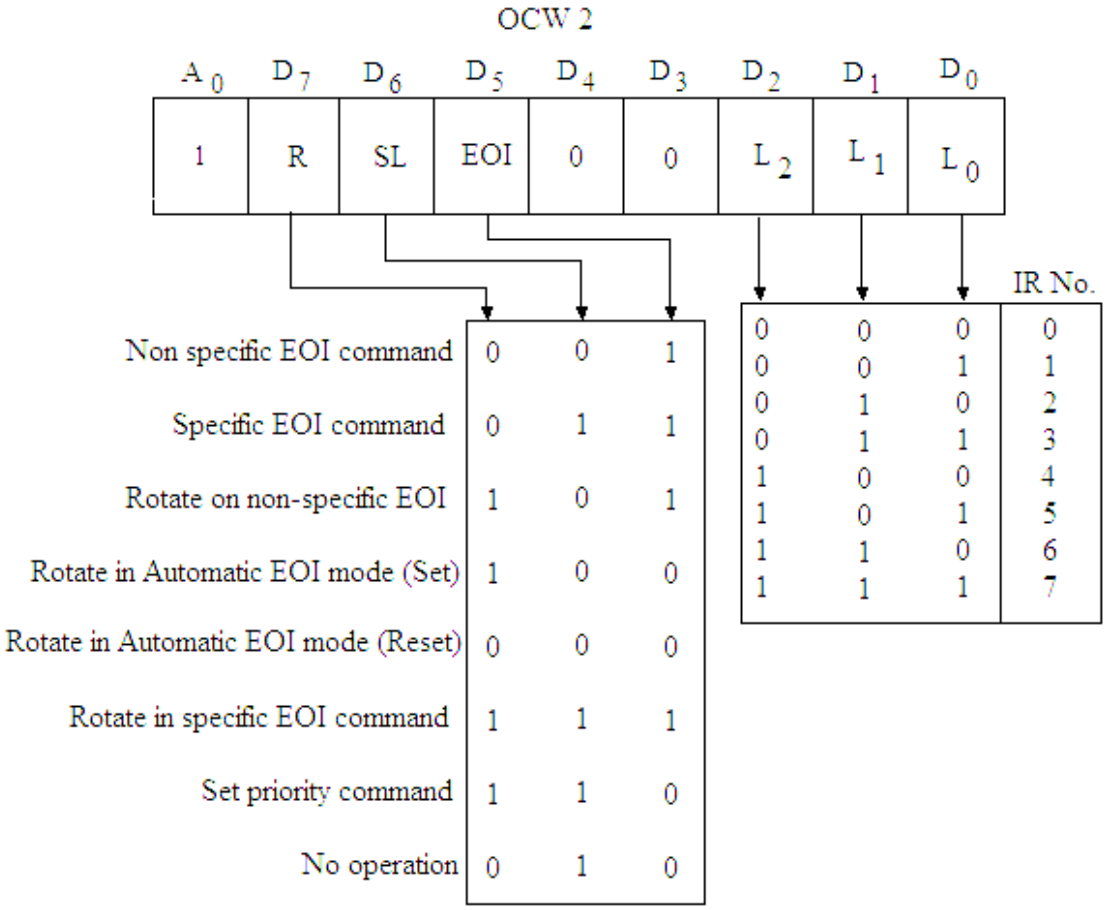


Fig. 11.16

Non Specific EOI Command

In fully nested mode, the highest level in the ISR would necessarily correspond to the last interrupt acknowledged and services. In such a case, a non-specific end of interrupt command (EOI) may be issued by the CPU by the OUT instruction to the 8259A with $A_0 = 0$. For this bit pattern in the format of OCW 2 is given by:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	0	0	0	0	0

The EOI bit (D_5) of the operation command word-2 (OCW 2) is set to 1. This resets the highest priority in service bit of ISR.

Specific EOI Command

If the fully nested mode is not used, the 8259 may not be able to determine the last interrupt acknowledged. In such case, a specific EOI command will have to be issued by the CPU by the OUT instruction to the 8259A with $A_0 = 0$. For this bit pattern in the format of OCW 2 is given by:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	1	0	0	L_2	L_1	L_0

In this bit pattern SL and EOI bits must be set to 1 and the encoded three bit value of the specific bit is written for L_2, L_1, L_0.

For example, in order to reset the bit 3 of the in service register corresponding to IR3, then the bit pattern of the OCW 2 will be given by:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	1	0	1	0	0	0	= 68 H

This operation command OCW 2 must be issued twice in case of cascaded mode; one for master and the other for slave.

Automatic End of Interrupt (AEOI)

If this mode is set, no command is necessary to reset the highest priority in service bit in the ISR. The bit in ISR will be reset automatically at the trailing edge of the third $\overline{INTA}$ signal sent by the CPU. This AEOI mode can only be used for a master 8259A and not for a slave. The AEOI is actuated by writing ICW 4 with D_1 bit to 1. At the time of writing the ICW 4, the A_0 line should be 1.

Operation Command Word-3 (OCW 3)

The format for OCW 3 is shown in figure 11.17. This word is used to read the status of the In Service Register (ISR) and Interrupt Request Register (IRR**); and to set or reset the special mask and polled modes.**

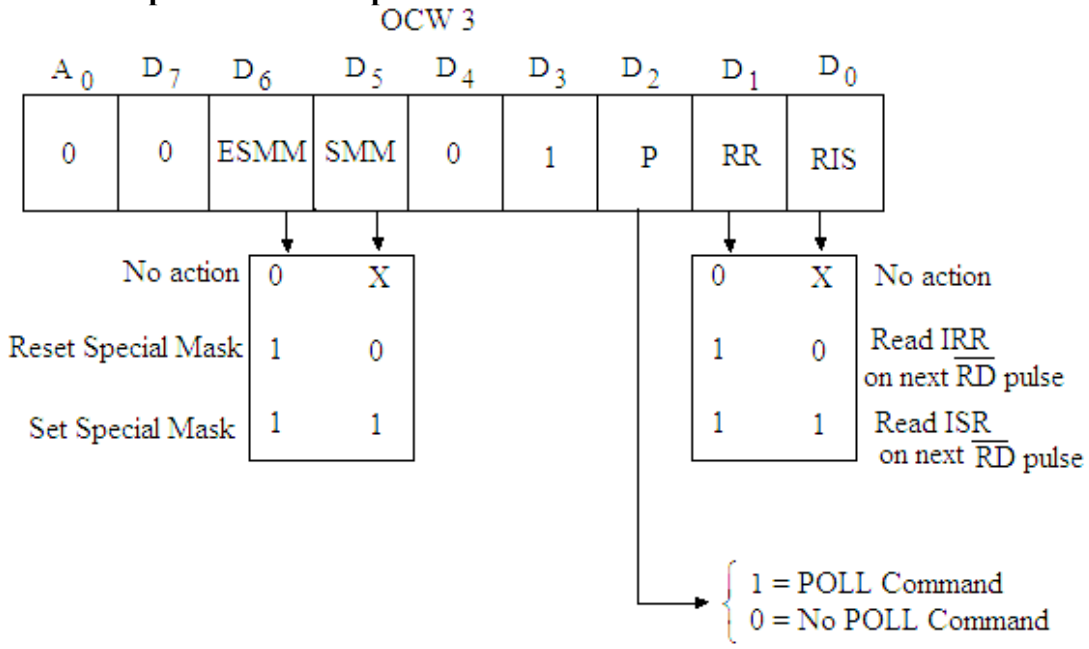


Fig. 11.17

11.8 INTERRUPT MODES OF 8259A

There are following modes of operations of 8259A:

- Fully Nested Mode
- Rotating Priority Mode
- Special Mask Mode
- Polled Mode

11.8.1 Fully Nested Mode (FNM)

The fully nested mode is the general purpose mode in which all interrupt requests are arranged from the highest to lowest priority level. In general, IR_0 has the highest

priority and IR_7 has the lowest priority. This is the default mode setting after initialization. The 8259 continues to operate in FNM until the mode is changed by OCWs. As discussed above, when an interrupt request is acknowledged, the 8259 determines the highest priority interrupt and set its corresponding bit in in-service register (ISR). The vector address corresponding to this interrupt is then put out. The bit in ISR remains set until an End of Interrupt (EOI) command through OCW 2 is issued by the CPU. At the end of the service routine, the corresponding bit in ISR is reset using OCW2.

11.8.2 Rotating Priority Mode

Though by default the priorities of the inputs are assigned as IR0 as the highest priority and IR7 as the lowest priority. But the priorities may be changed using either by automatic rotation or by specific rotation.

Automatic Rotation

This mode is used when all the interrupts are assigned equal priority. In this mode, currently executing interrupt is given the lowest priority after completing its service routine. The next interrupt level is assigned the highest priority. For example, the interrupt request IR_4 and IR_6 are being serviced as it can be read from the status of ISR (figure 11.18). As usual IR_4 is assigned highest priority and will be executed first. When the interrupt service routine for IR_4 is executed, this bit will be reset after EOI command as discussed earlier. Now in this mode of automatic rotation the interrupt IR_4 will have the lowest property and the next interrupt will have the highest priority as shown in figure 11.19.

(IR6, IR4 Being serviced: IR4 is active)

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
ISR Status	0	1	0	1	0	0	0	0
Priority Status	7	6	5	4	3	2	1	0

Lowest Priority — Highest Priority

Before Rotation

Fig. 11.18

(After EOI from IR4 routine)

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
ISR Status	0	1	0	0	0	0	0	0
Priority Status	2	1	0	7	6	5	4	3

After Rotation

Fig. 1.19

Automatic rotation mode on an AEOI is set by loading an OCW 2 with R = 1, SL = 0 and EOI = 0 and is reset by loading an OCW 2 with R = 0, SL = 0, and EOI = 0.

Specific Rotation

In this, the programmer can change the priority by programming the lowest priority. This mode is set by CPU by issuing an OUT instruction with A0 = 0 and other bit pattern as given below:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	0	0	L_2	L_1	L_0

The bits D_2-Do specify the interrupt level that is to be assigned lowest priority. The specific rotation can be accomplished by using the Rotation of specific EOI in OCW2 as R = 1, SL = 1 and EOI = 1.

11.8.3 Special Mask Mode

It may sometimes be desirable to selectively enable lower priority interrupts. Usually, if an EOI command is not given to the 8259A, the in service bit of the last serviced interrupt is not reset. As a result all lower priority interrupts are kept disabled.

This special mask mode can be set by making ESMM and SMM bits as 1 in OCW 3. When a mask bit is set in OCW 1, all further interrupts at that level are inhibited; while interrupts on all other levels that are not masked are enabled. It is thus possible to selectively enable interrupts by programming the mask register. This mode can be cleared by loading an OCW 3 with ESSM = 1 and SMM = 0.

11.8.4 Polled Mode

In this mode the INT output of 8259A is not used. It is either not connected to the INTR input of 8085A or the system interrupts are disabled by software. The 8259A assign the priorities IR_0 through IR_7 inputs and provides a status port that can be polled. The status byte has the form as shown in figure 11.20.

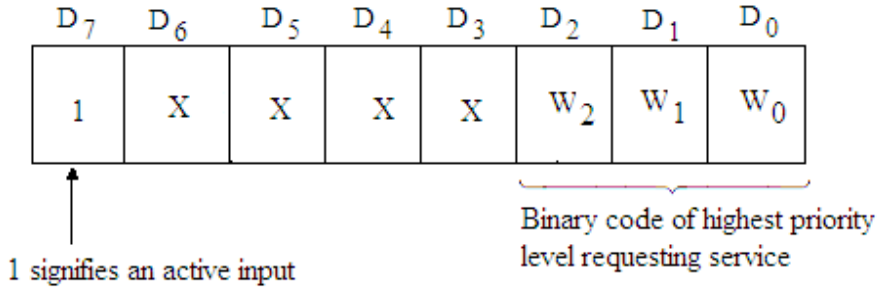


Fig. 11.20

11.9 STATUS READ OPERATION OF 8259

The status of the Interrupt Request Register (IRR), the In-Service Register (ISR) and the Interrupt Mask Register (IMR) of the 8259 may be read by using appropriate read command. These read commands are described below:

Reading IRR Status

Just before reading the IRR (interrupt request register), OCW 3 must be written by making RR (read request) as 1 and RIS (read ISR) as 0. At the time of writing this operation command word, address line A_0 and read signal must be 0. Internally OCW 2 and OCW 3 are recognized by sensing the D_4 and D_3 bits as:

D_4	D_3	
0	0	for OCW 2

1 0 for OCW 3.

After writing the OCW 3, the interrupt request status can be read by using the IN instruction.

IN F0 H if F0 H is the port address with $A_0 = 0$.

Reading ISR Status

For reading ISR status, OCW 3 is initially written with RR = 1 and RIS = 1. The In-service register content can be read using the IN instruction with $A_0 = 0$.

IN F0 H if F0 H is the port address with $A_0 = 0$.

Reading IMR Status

The contents of Interrupt Mask Register (IMR) can be read just by making the A0 = 1. The OCW 3 is not written just before reading.

IN F1 H if F1 H is the port address with $A_0 = 1$.

Example 11.3. *Assume that initialization command words have been written to 8259A (single) having the port address 20 H and 21 H for $A_0 = 0$ and $A_0 = 1$ respectively. Now write proper operation control words for a fully nested priority structure. Mask IR1, IR4 and IR6; enable ISR for a read. What code should be used at the end of interrupt service routine?*

Solution. The initialization commands have already been written. The operation command words are written as:

OCW 1 $A_0 = 1$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
M7	M6	M5	M4	M3	M2	M1	M0	
0	1	0	1	0	0	1	0	= 52 H

In this OCW 1, M1, M4 and M6 are 1 as IR1, IR4 and IR6 are masked.

For reading ISR status, OCW 3 is initially written with RR = 1 and RIS = 1.

OCW 3

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	ESMM	SMM	1	P	RR	RIS	
0	0	0	0	1	0	1	1	= 0B H

In fully nested mode, the highest level in the ISR would necessarily correspond to the last interrupt acknowledged and services. In such a case, a non-specific end of interrupt command (EOI) may be issued by the CPU by the OUT instruction to the 8259A with $A_0 = 0$. For this bit pattern in the format of OCW 2 is given by:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	1	0	0	0	0	0	= 20 H

The EOI bit (D_5) of the operation command word-2 (OCW 2) is set to 1. This resets the highest priority in service bit of ISR.

PROGRAM:

```
MVI   A, 52 H     ; OCW 1 is loaded to accumulator.
OUT   21 H        ; 21 H is the port address for A0 = 1.
MVI   A, 0B H     ; OCW 3 is loaded to accumulator to enable the ISR
                    for read.
OUT   20 H        ; 20 H is the port address for A0 = 0,
IN    20 H        ; Read ISR
```

Each interrupt service routine must end by giving the non-specific EOI command which resets in service bit (ISR bit).

```
MVI   A, 20 H
OUT   20 H
RET
```

Example 11.4. *Write Initializing command word to program 8259A (single) with port addresses C0 H and C1 H for A_0 = 0 and A_0 = 1 respectively. Let the starting address of the interrupt request is 7000 H with interval spacing of 4. The interrupts are edge triggered. Interrupts 1 and 3 are masked.*

Solution. The single 8259A is to be initialized, so initialization command words ICW 1 and ICW 2 are to be written as given below:

ICW 1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
A_7	A_6	A_5	1	LTM	ADI	SNGL	IC_4	
0	0	0	1	0	1	1	0	= 16 H

In ICW 1, D_0 = 0 as no ICW 4 is required; D_1 = 1 as single 8259 is used, D_2 = 1 as spacing interval of 4 is required, D_3 = 0 as these are edge triggered inputs, D_7 D_6 D_5 are 000 as 00 H is the address of the first interrupt request (**000**0 0000).

ICW 2

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
1	1	1	0	0	0	0	0	= E0 H

ICW 2 gives the address 70 H (base address) 0111 0000.

The operation command word is written as:

OCW 1 A_0 =1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
M7	M6	M5	M4	M3	M2	M1	M0	
0	0	0	0	1	0	1	0	= 0A H

In this OCW 1, M1and M3 are 1 as IR1 and IR3 are masked.

Program:

```
MVI   A, 16 H      ; ICW 1 code is loaded to acc.
OUT   C0 H         ; C0 H is the port address for A0 = 0.
MVI   A, 70 H      ; Base address is given by ICW 2.
OUT   C1 H         ; ICW 2 port address for A0 = 1.
MVI   A, 0A H      ; OCW 1 for masking.
OUT   C1 H         ; Port address for A0 = 1.
```

Example 11.5. *After initializing 8259A with port addresses C0 H and C1 H for A_0 = 0 and A_0 = 1 respectively. Give the instruction sequence to read ISR status and IMR status.*

Solution. After initialization of 8259A , OCW 3 is written with RR = 1 and RIS = 1, to enable reading of ISR status (this is necessary for reading the status of ISR). So OCW 3 is given as:

OCW 3

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	ESMM	SMM	1	P	RR	RIS	
0	0	0	0	1	0	1	1	= 0B H

The instruction sequence is then given for the reading the status of ISR and IMR as:

Program:

```
MVI   A, 0B H      ; OCW 3 code is loaded to acc. to enable the
                     reading of ISR status.
OUT   C0 H         ; C0 H is the port address for A0 = 0.
IN    C0 H         ; Read the status of ISR.
IN    C1 H         ; Read the status of IMR.
```

Example 11.6. *An interrupt service routine written for 8259A having port addresses F0 H and F1 H, ends with the instruction sequence given as:*

```
MVI  A, 22 H
OUT  F0 H
RET
```

How does the 8259A interpret these instructions?

Solution. First instruction MVI A, 22 H will show the bit format for OCW 2 is given by:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	0	0	0	1	0

This resets the IR2 in service bit of ISR. This is a non-specific end of interrupt command (EOI) may be issued by the CPU by the OUT instruction to the 8259A with $A_0 = 0$.

PROBLEMS

1. Draw the block diagram of programmable interrupt controller (PIC) 8259A and discuss its various blocks.
2. Describe how the interfacing of 8259A with 8085A microprocessor is done.
3. Discuss how the cascading of 8259A is carried out. How many 8259s may be cascaded? What is the function of master 8259A in the cascading mode?
4. Discuss how initialization of 8259 is carried out.
5. Describe various Initialization command words in 8259A (PIC).
6. What do you understand by operation command words in 8259A? Discuss any one command word.
7. Explain the various interrupt modes of 8259A.
8. Discuss status read operation of 8259A.
9. Explain the vectoring data format of 8259A (PIC).
10. Discuss Rotating Priority mode of operation of 8259A.
11. Initialize 8259A to interface eight edge triggered inputs. The restart address of first interrupt request is 1020 H. There is a spacing interval of four byte between each interrupt request. No cascading of 8259A is made. Let the port address is C0 H and C1 H for $A_0 = 0$ and $A_0 = 1$ respectively.
12. Assume that initialization command words have been written to 8259A(single) having the port address F0 H and F1 H for $A_0 = 0$ and $A_0 = 1$ respectively. Now

write proper operation control words for a fully nested priority structure. Mask IR2, IR3 and IR5; enable ISR for a read. What code should be used at the end of interrupt service routine?

13. Write Initializing command word to program 8259A (single) with port addresses 20 H and 21 H for $A_0 = 0$ and $A_0 = 1$ respectively. Let the starting address of the interrupt request is 8000 H with interval spacing of 4. The interrupts are edge triggered. Interrupts 0 and 4 are masked.
14. Write a non-specific end of interrupt command (EOI) to reset IR3 in service bit of ISR4. Assume the port addresses 20 H and 21 H for 8259.
15. After initializing 8259A with port addresses F0 H and F1 H for $A_0 = 0$ and $A_0 = 1$ respectively. Give the instruction sequence to read ISR status and IMR status.
16. Explain the following initialization instructions:

```
MVI   A, 3E H      ; ICW 1
OUT   80 H         ; Initialize 8259A
MVI   A, E0 H      ; ICW 2
OUT   81 H         ; Initialize 8259A
```

12

Direct Memory Access Controller: 8257

It has already been discussed in an earlier chapter of this book, that the Direct Memory Access (DMA) is the efficient way of transferring the data from the memory to the I/O devices or vice versa without involving the CPU. In DMA operation, the CPU relinquishes the control over the data bus and address bus, so that these can be used for transfer of data between the memory and I/O devices directly. For the data transfer using DMA process, a request to the microprocessor by the I/O device is sent and on receipt of such request, the microprocessor relinquishes the address and data buses and informs the I/O devices of the situation by sending Acknowledge signal. However, in the usual method of data transfer between the I/O devices and the memory, it involves the microprocessor. Each byte is routed through the accumulator, as a result of which it takes lot of time for transferring a large amount of data. For the purpose of DMA operation, a DMA controller is necessary to interface the peripheral to the system. Intel 8257 is a 4-chennel programmable DMA controller used with 8085 and other microprocessors for this mode. In this chapter the details of the DMA controller 8257 will be discussed.

12.1 BLOCK DIAGRAM OF 8257

The 8257 Programmable DMA controller is available in the form of IC dual in line package. It consists of 40 pins and requires +5 volt d.c. supply for its operation. The 8257 enables to interface up to four I/O devices to the microprocessor for the data transfer between the memory and the I/O devices or vice versa using DMA. It has forum DMA channels. Each channel consists of two 16 bit registers. One of these registers holds the address to be used in the next DMA cycle. The other register holds, in the least significant 14 bits, the total number of bytes to be transferred between the memory and peripherals. The two most significant bits of this register are set to indicate the operation to be performed by the device on that channel. A total of 16384 (2^{14} = 16 K) bytes of data may directly be transferred under DMA control without any intervention of the microprocessor.

Figure 12.1 shows the pin diagram of 8257. The pin names of this IC are given below:

D_0-D_7	Data bus	HLDA	HOLD Acknowledge
A0-A7	Address bus	AEN	Address Enable
$\overline{I/OR}$	I/O Read	ADSTB	Address Strobe
$\overline{I/OW}$	I/O Write	TC	Terminal Count
$\overline{MEMR}$	Memory Read	MARK	Modulo 128 MARK
$\overline{MEMW}$	Memory Write	DRQ_0-DRQ_3	DMA Request Input
CLK	Clock Input	$\overline{DACK0}$ - $\overline{DACK3}$	DMA Acknowledge OUT
RESET	RESET Input	$\overline{CS}$	Chip Select
READY	Ready	V_{CC}	+5 V
HRQ	HOLD Request	GND	Ground

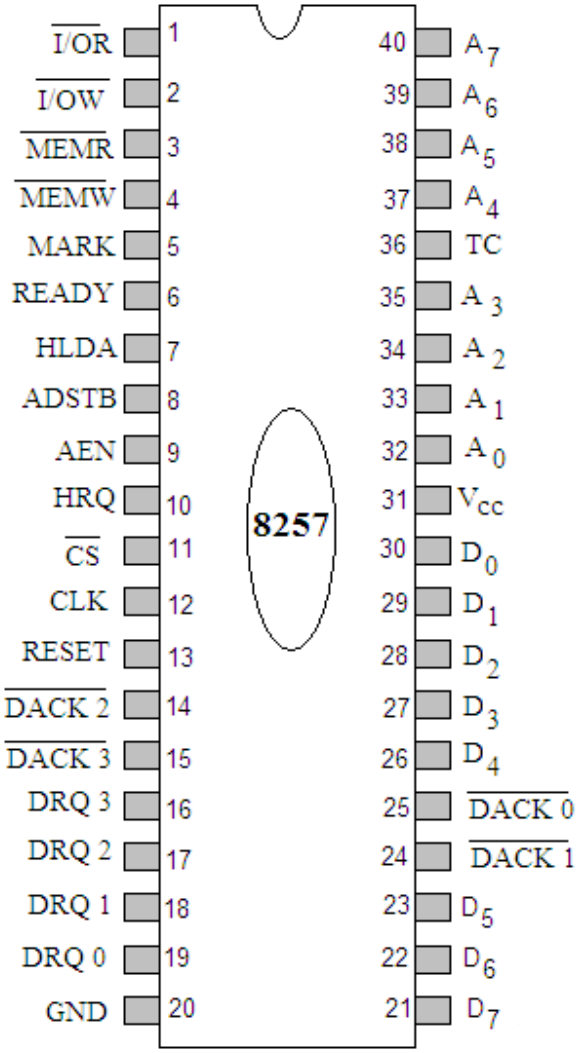


Fig. 12.1

The internal block diagram and schematic diagram of 8257, DMA Controller (DMAC) are shown in figures 12.2 and 12.3 respectively. It contains the following blocks:

- DMA Channels
- Data Bus Buffer
- Read/ Write Logic
- Control Logic
- Mode Set Register
- Status Word Register

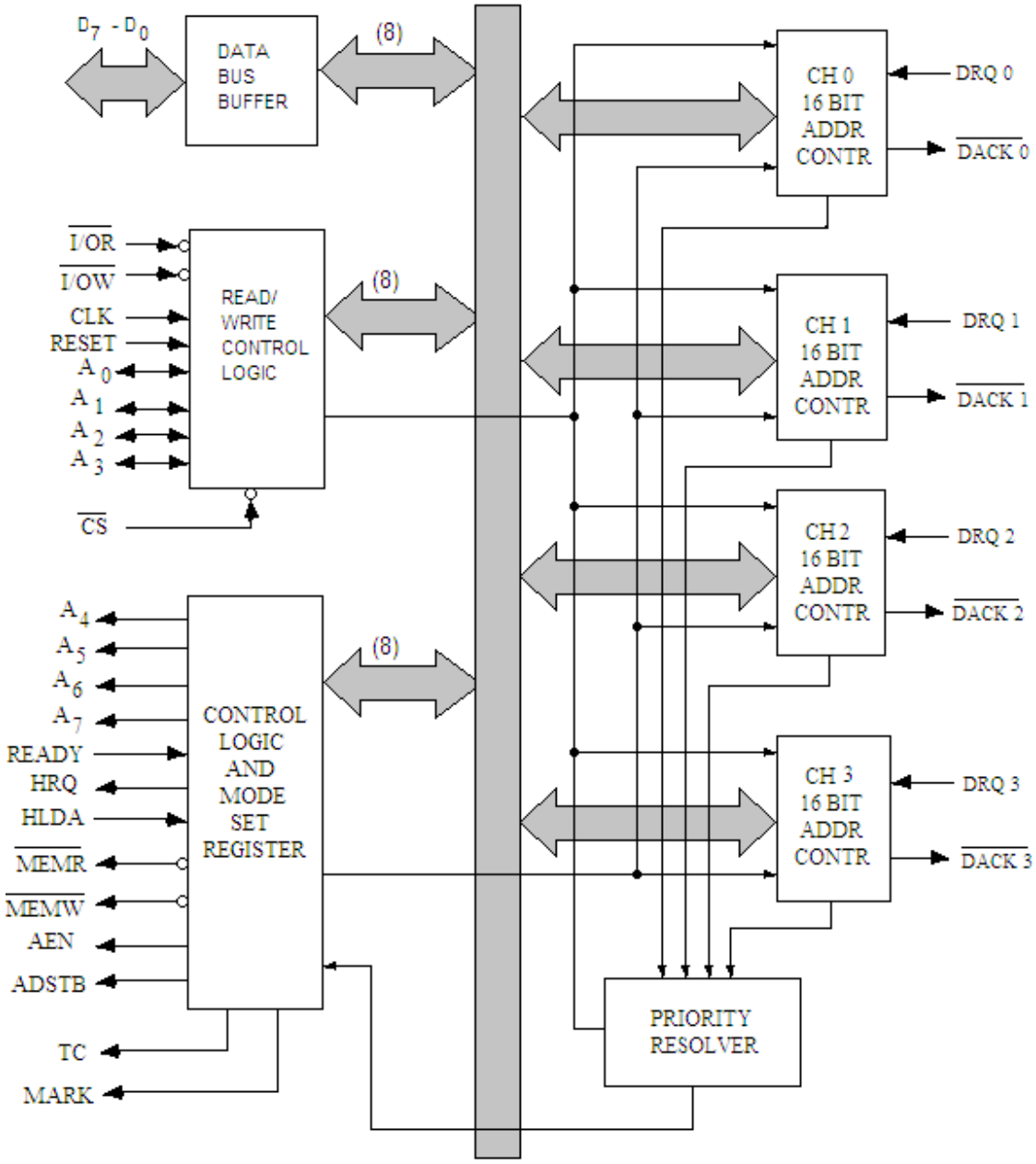


Fig. 12.2

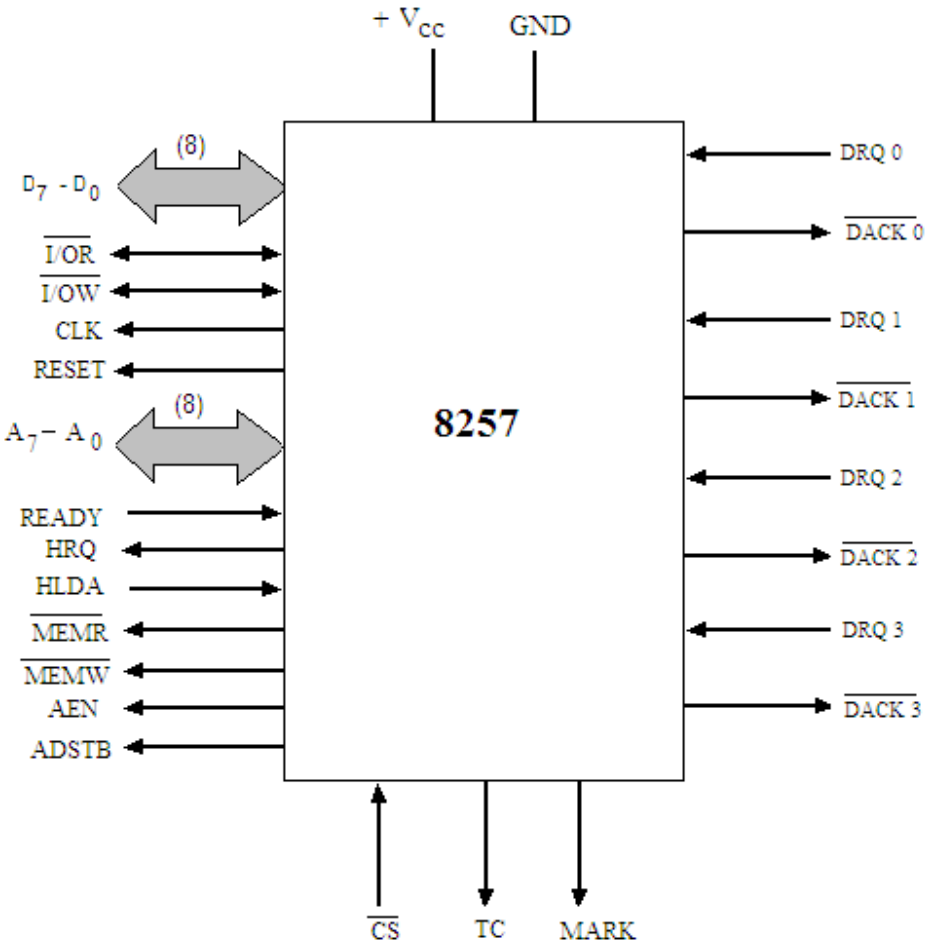


Fig. 12.3

12.1.1 DMA Channels

The 8257 provides 4 channels which can be connected to four separate I/O devices. These channels are designated as Channel 0 (CH-0), Channel 1 (CH-1), Channel 2 (CH-2) and Channel 4 (CH-4). Each of these channels has two 16-bit registers:

DMA Address Register
Terminal Count Register.

DMA Address Register contains the memory address from where the data transfer should begin. This address is loaded by the CPU.

The terminal count register is used to store the number of bytes to be transferred. In this 16 bit register, the least significant 14 bits are used to store the number of bytes to be transferred. In this way one DMA operation can transfer to a maximum of 16 K bytes. The most significant two bits of the terminal register represent the operation to be performed by the device. The operations to be performed are Write, Read and Verify. Figure 12.4 shows the format of the terminal count register of a DMA channel.

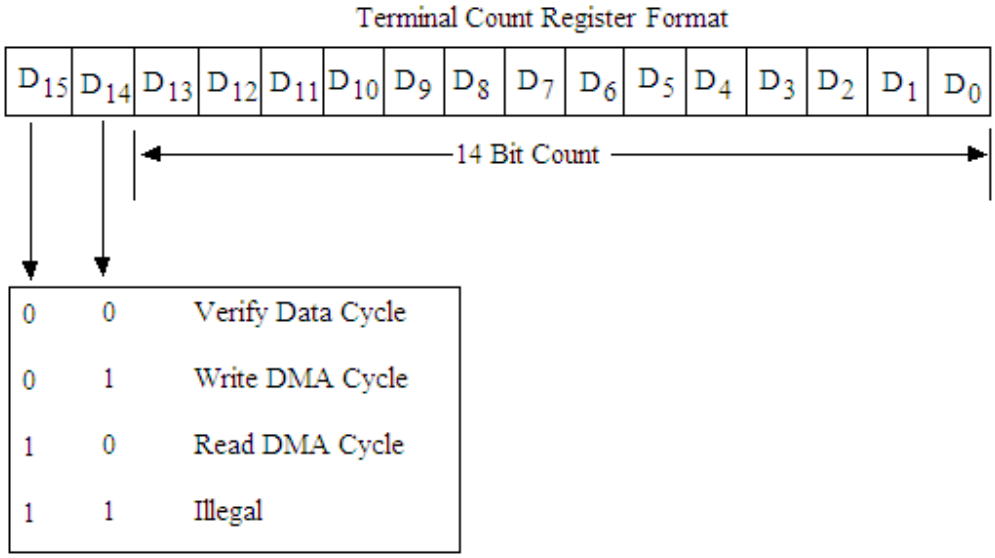


Fig. 12.4

- During DMA write operation, the data is transferred from the peripheral to memory.
- During DMA read operation, the data is transferred from memory to peripheral.
- DMA verify operation does not actually involve the transfer of data.

Each channel has two signals: DMA REQUEST Signal and DMA ACKNOWLEDGE Signal.

DMA REQUEST Signals (DRQ0 – DRQ3):

There are four DMA request signals (DRQ0 – DRQ3) in the 8257 connected one with each channel. Each channel accepts a DMA request through this pin from the peripheral. The DMA controller 8257 has priority facility. The channel 0 (CH 0) has the highest priority and the channel 3 (CH 3) has the lowest priority. As per the priority, a request is generated by the peripheral by making corresponding DRQ line high and holding it high until DMA operation is over.

DMA ACKNOWLEDGE Signals ($\overline{DACK0}$ - $\overline{DACK3}$):

Likewise the DMA request signals, 8257 has the four DMA acknowledge signals ($\overline{DACK0}$ - $\overline{DACK3}$) connected one with each channel. As discussed above, each channel accept a DMA request through it DRQ line from the peripheral. After the receipt of the request channel issues an acknowledge signal through $\overline{DACK}$ output of that channel. It is an active low signal and signifies that the request to be serviced for a DMA cycle is accepted. This signal is made high on completion of the transfer.

12.1.2 Data Bus Buffer

This three state, bidirectional eight bit buffer (D_0-D_7) interfaces the 8257 to the system data bus. When the CPU has the control over the system buses (Slave Mode), the D_0-D_7 pins of the 8257 are used as input pins. The eight bits of data for a DMA address register and terminal count register or the mode set register are received on the data bus. In this slave mode, the CPU can also read eight bits of data at a time from the 16-bit DMA address and terminal count register or from the eight bit status register. However, in the master mode (for DMA cycle, when the CPU relinquishes the control over the control and data buses), the D_0-D_7 pins are used for different purpose. At the beginning of each DMA cycle, most significant bits of DMA address register are put on these pins which are further latched to an external chip 8212 used as latch. These latched values are put on A8-A15 of the system address bus. The D_0-D_7 bus is then released to handle the memory data transfer during the remainder of the DMA cycle.

12.1.3 Read/ Write Logic

When the CPU is programming or reading one of the registers of 8257 (i.e. when the 8257 is in slave mode), the Read/ Write logic accepts the I/ O Read ($\overline{I/OR}$) or I/O Write ($\overline{I/OW}$) signal. It then decodes the least significant four address bits (A_0-A_3), and writes the contents of the data bus into the addressed register (if $\overline{I/OW}$ is active) or places the contents of the addressed register onto the data bus (if $\overline{I/OR}$ is active). During DMA cycle (i.e. when the 8257 is in master mode), the Read/ Write logic generates the I/O Read and Memory Write cycle (DMA Write cycle) and generates the Memory Read and I/O Write Cycle (DMA Read cycle).

$\overline{I/OR}$ (I/O Read)

This is an active low, bidirectional three-state pin. In the slave mode, it is an input signal which allows the 8-bit status register or the upper/lower byte of a 16-bit DMA address register or terminal count register to be read. In the master mode, $\overline{I/OR}$ is a control output, which is used to access data from a peripheral during the DMA write cycle.

$\overline{I/OW}$ (I/O Write)

This is also an active low, bidirectional three-state pin. In the slave mode, it is an input signal which allows the contents of the data bus to be loaded into the 8-bit mode set register or the upper/lower byte of a 16-bit DMA address register or terminal count register. In the master mode, $\overline{I/OW}$ signal is a control output which allows data to be output to a peripheral during a DMA read cycle.

CLK (Clock input)

This input is generally connected to an external clock or to the 8085A CLK output.

RESET (Reset)

This is an active high asynchronous input which disables all DMA channels and clears the mode set register. It also tri states all the control lines.

$\overline{CS}$ (Chip Select)

This is an active low input which enables the I/O Read or I/O write input when the 8257 is being read or programmed in the slave mode. In the master mode, $\overline{CS}$ input is automatically disabled. This is done to prevent the 8257 from selecting itself when it puts out DMA addresses on the address bus.

A_0-A_3 (Address lines)

These least significant four address lines are bidirectional. In the slave mode, they are inputs which select one of the registers to be read or programmed. In the master mode, they are outputs which constitute the least significant four bits of the 16-bit address generated by the 8257.

12.1.4 Control Logic

This block controls the sequence of operations during all DMA cycles by generating the appropriate control signals and the 16-bit address that specifies the memory location to be accessed. The pins associated with the control logic are described below.

A_4-A_7 (Address lines)

These four address lines are the three state output lines. Bits 4-7 of the DMA address register are put on these address lines during the DMA cycles. So, the A_0-A_7 of 8257 carry the least significant byte of DMA address during the execution of a DMA cycle.

READY

This ready input signal can be used by the devices with slow access time in order to insert the wait states during DMA transfer. It has similar function to the Ready input pin of 8085.

HRQ (Hold Request)

This output requests control of the system bus. In systems with only one 8257, this signal will be normally applied the HOLD input of the CPU. HRQ must confirm to specified setup and hold times.

HLDA (Hold Acknowledge)

This is an input from the CPU that informs the 8257 that it has relinquished control of the buses and 8257 may take over the control.

$\overline{MEMR}$ (Memory Read)

This is a three-state active low output which is used to read data from the addressed memory location during DMA Read cycles.

$\overline{MEMW}$ (Memory Write)

This is also three state active low output which is used to write data into the addressed memory location during DMA Write cycles.

ADSTB (Address Strobe)

This is an active high signal which is generated at the beginning of each DMA cycle and has a similar function as that of ALE pin of 8085A.

AEN (Address Enable)

This is an output pin used to float the system data bus and the system control bus. It may also be used to float the system address bus. It may further be used to isolate the 8257 data bus from the system data bus to facilitate the transfer of the 8-bit most significant DMA address bits over the 8257 data I/O pins. When the 8257 is used in an I/O device structure, this AEN output is used to disable the selection of an I/O device when the DMA address is on the address bus. The I/O device selection should be determined by the DMA acknowledge output for the four channels.

TC (Terminal Count)

This is an active high output. When this pin is high it indicates to the currently selected peripheral that the present DMA cycle is the last cycle for this data block. This output goes high, when the contents of the terminal count register of the selected channel are equal to zero. By programming the mode word, a channel can automatically be disabled after its last DMA cycle.

MARK (Modulo 128 Mark)

This active high output notifies the selected peripheral that the current DMA cycle is the 128^{th} cycle since the previous mark output. Mark always occurs at 128 (and all multiples of 128) cycles from the end of the data block. Only if the total umber of DMA cycles (n) is evenly divisible by 128, mark will occur at 128 cycles from the beginning of the data block.

12.1.5 Mode Set Register

The format of Mode Set Register of 8257 is shown in figure 12.5. The control word in the mode set register enables or disables channels and determines other functions. The mode set register can be accessed only for write operation. The various bits in this mode set register enable each of the four DMA channels and four different options for the 8257. The mode set register is cleared by the RESET input, thus disabling all options, inhibiting all channels. This mode set register is also used for programming the 8257 in the following options:

- Rotating Priority
- Extended Write
- TC Stop
- Auto Load

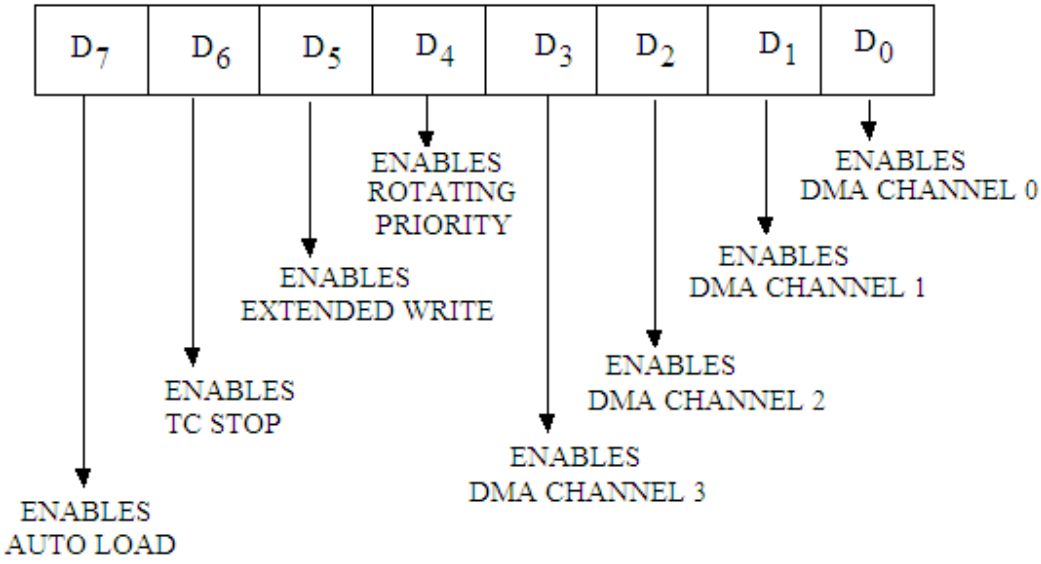


Fig. 12.5

Rotating Priority

The bit 4 (D_4) of the Mode Set Register enables to set the rotating priority mode of the 8257. In the rotating priority mode, the priority of the channels has a circular sequence as shown in figure 12.6. After each DMA cycle the priority of channels has circular sequence. After each DMA cycle, the priority of each channel changes. The channel which had just been serviced will have the lowest priority (Figure 12.7).

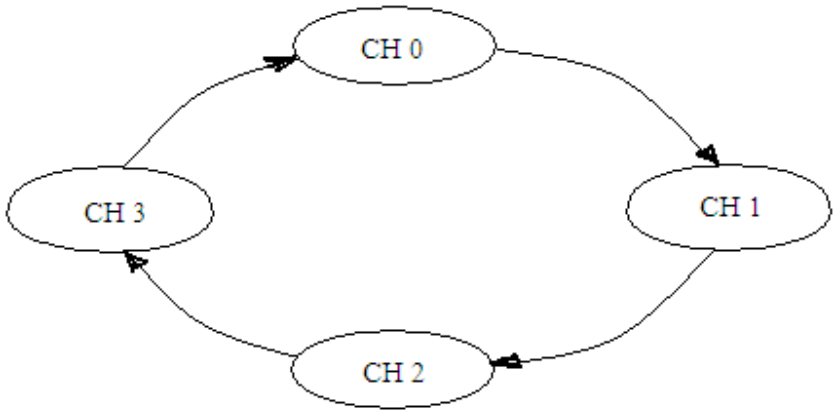


Fig. 12.6

If the rotating priority bit is not set ($D_4 = 0$), each DMA channel has a fixed priority. In the fixed priority mode, channel 0 has the highest priority and the channel 3 has the lowest priority. If the rotating priority bit D_4 is set to 1, the priority of each channel changes after each DMA cycle (not each DMA request). Each channel moves up to the next highest priority assignment, while the channel which has just been serviced moves to the lowest priority assignment.

	CHANNEL JUST SERVED →	CH 0	CH 1	CH 2	CH 3
PRIORITY ASSIGNMENT →	HIGHEST	CH 1	CH 2	CH 3	CH 0
	↕	CH 2	CH 3	CH 0	CH 1
		CH 3	CH 0	CH 1	CH 2
	LOWEST	CH 0	CH 1	CH 2	CH 3

Fig. 12.7

Extended Write

The bit 5 (D_5) of the mode set register is used for extended write option. The data transfer with in microcomputer systems takes place asynchronously which allow the use of various types of memory and I/O devices with different access times. If the device can not be accessed within the specified amount of time, it returns a 'Not Ready' signal on the READY input of the 8257 due to which one or more wait states are inserted in the internal sequencing. Some devices are fast enough to be accessed without the use of wait states. But if they generate their Ready response with the leading edge of the $\overline{I/OW}$ or $\overline{MEMW}$ signal (which generally occurs late in transfer sequence), they would normally cause the 8257 to enter a wait state because it does not receive Ready in time. For systems with these types of devices, the extended write operation provides alternative timing for the I/O and memory write signals which allows the devices to return on early Ready and prevents the unnecessary occurrence of wait states in the 8257.

TC Stop

If the TC stop bit (D_6) of the mode set register is set, a channel is disabled (i.e. its enable bit is reset) after the terminal count (TC) output goes high, thus automatically preventing further DMA operation on that channel. To continue or begin another DMA operation, the enable bit of that channel must be reprogrammed.

If the TC is not set, the occurrence of TC bit output has no effect on the channels enable bits. In that case, it will be responsibility of the peripheral to cease DMA requests in order to terminate a DMA operation.

Auto Load

Bit D_7 of the mode set register enables Auto Load mode. In some cases, it becomes necessary to perform DMA operation repeatedly. In case of a CRT monitor, the data has to repeatedly send to the monitor and this process is called refreshing. With the auto load option, the DMA controller permits such repetitive operation. This bit permits channel 2 to be used for repeat block operation. If the auto load mode is set ($D_7 = 1$), the initial parameters of channel 2 are automatically copied onto channel 3. The channel 2 is programmed with the initial parameters for the first DMA block. When the DMA block of channel 2 is completed, the parameters stored in channel 3 are copied onto channel 2. The channel 2 is then serviced again with the same parameters for repetitive DMA operations. For chaining operation, channel 3 is loaded with a different set of parameters

after channel 2 is loaded; the new parameters are copied onto channel 2 during the update cycle and channel 2 is serviced with a new set of parameters.

12.1.6 Status Word Register

The eight bit status register is shown in figure 12.8. The status register can be read for the status of the terminal counts of the four channels. It also contains the update flag. The count bits (D_0-D_4) of the four channels are set when the terminal count output is high for that channel. The TC bits go low when the status word is read or when the 8257 receives a Reset input. The update flag is reset when:

- 8257 is reset or
- Auto load option in the mode set register is disabled or
- Update cycle gets completed.

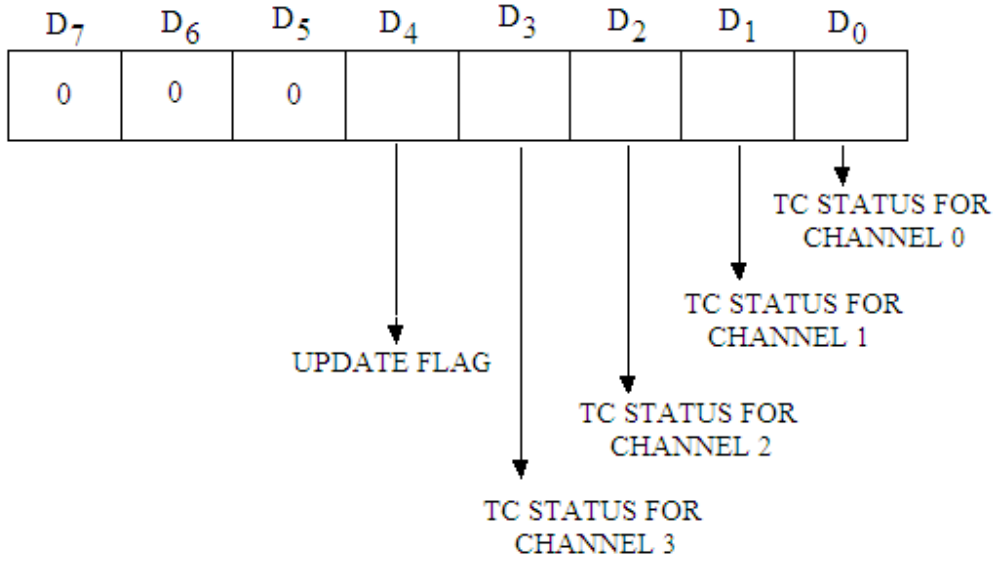


Fig.12.8

12.2 PROGRAMMING OF 8257

As discussed above, there are four pairs of channel registers in 8257; each pair consisting of a 16-bit Address Register and 16 bit Terminal Count Register i.e. one pair for each channel. The 8257 also includes two general registers one 8-bit Mode Set Register and other 8-bit Status Register. These various registers are accessed with the help of four input lines (A_0-A_3) and an internal F/L flip-flop (First/ Last Flip-flop). The address bit 3 (A_3) signifies whether a channel register or the mode set register is being accessed. If A_3 = 0, then channel register is accessed and if A_3 = 1, then the mode set register is accessed. The other three address bits (A_0-A_2) specify which register is to be accessed. When these bits (A_0-A_2) are all zero, the mode set register or status register is being accessed. The 8257 distinguishes between the mode set register (which is write only operation) and the status operation (which is read only operation) by the $\overline{I/OW}$ and $\overline{I/OR}$ inputs respectively. Table 12.1 illustrates the status of A_3 and the control inputs for accessing the various registers. For channel registers, A_2 A_1 specify one of the four channels i.e. for channel 0, it is 00, for channel 1 it is 01, for channel 2 it 01 and for channel 3 it is 11. The bit A_0 specifies whether channel register or terminal count register

is to be accessed. If A_0 = 0, channel register is accessed and if A_0 = 1, terminal count register is accessed. The F/L flip-flop is reset by reset input. This flip-flop toggles for access of the 8257 registers and determines whether upper or lower order byte of a particular register is accessed. It requires that the reading of the address registers or the terminal count registers should be done in pairs; the lower order byte should be accessed first. The addressing of the registers is shown in table 12.2.

Table 12.1

CONTROL INPUT	$\overline{CS}$	$\overline{I/OW}$	$\overline{I/OR}$	A_3
Program Half of a Channel Register	**0**	**0**	**1**	**0**
Read Half of a Channel Register	**0**	**1**	**0**	**0**
Program Mode Set Register	**0**	**0**	**1**	**1**
Read Status Register	**0**	**1**	**0**	**0**

Table 12.2

ADDRESS INPUTS				State of		Register Accessed
A_3	A_2	A_1	A_0	FF F/L	$\overline{CS}$	
0	0	0	0	0	0	LOB of DMA address CH 0
0	0	0	0	1	0	HOB of DMA address CH 0
0	0	0	1	0	0	LOB of Counts of Terminal Count Register CH 0
0	0	0	1	1	0	HOB of Counts of Terminal Count Register CH 0
0	0	1	0	0	0	LOB of DMA address CH 1
0	0	1	0	1	0	HOB of DMA address CH 1
0	0	1	1	0	0	LOB of Counts of Terminal Count Register CH 1
0	0	1	1	1	0	HOB of Counts of Terminal Count Register CH 1
0	1	0	0	0	0	LOB of DMA address CH 2
0	1	0	0	1	0	HOB of DMA address CH 2
0	1	0	1	0	0	LOB of Counts of Terminal Count Register CH 2
0	1	0	1	1	0	HOB of Counts of Terminal Count Register CH 2
0	1	1	0	0	0	LOB of DMA address CH 3

0	1	1	0	1	0	HOB of DMA address CH 3
0	1	1	1	0	0	LOB of Counts of Terminal Count Register CH 3
0	1	1	1	1	0	HOB of Counts of Terminal Count Register CH 3
1	0	0	0	0	0	Mode Set Register (Write Only)
1	0	0	0	0	0	Status Register (Read Only)

HOB – High order byte
LOB – Low order byte

The microprocessor can either read data from or write data into DMA address register and the terminal count register of each channel of 8257. The status register can be read and mode set register may be programmed. All these operations have been discussed above and carried out during the slave mode of 8257.

If the address decoder circuit for the chip select terminal of DMA controller 8257 shown in figure 12.9 is used, it will give the addresses of DMA registers and terminal count registers of all the four channels along with the address of the mode set registers. These addresses are given in table 12.3.

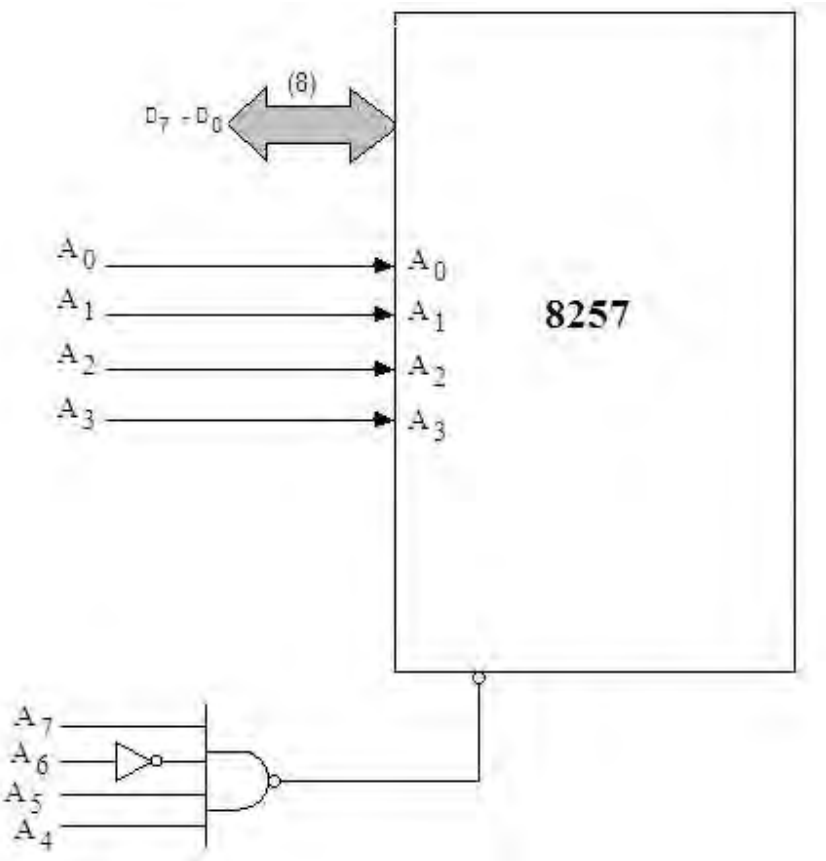


Fig. 12.9

Table 12.3

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address	
0	1	0	0	0	0	0	0	70 H	DMA address CH 0
0	1	0	0	0	0	0	1	71 H	TC address CH 0
0	1	0	0	0	0	1	0	72 H	DMA address CH 1
0	1	0	0	0	0	1	1	73 H	TC address CH 1
0	1	0	0	0	1	0	0	74 H	DMA address CH 2
0	1	0	0	0	1	0	1	75 H	TC address CH 2
0	1	0	0	0	1	1	0	76 H	DMA address CH 3
0	1	0	0	0	1	1	1	77 H	TC address CH 3
0	1	0	0	1	0	0	0	78 H	Mode Set Reg addr.

The various registers have to be first initialized and then DMA operation will be started. The steps, to be followed for the DMA operation of 8257 (Master Mode), after its initialization, are given as:

- The I/O device sends a DMA request signal (DRQ) when it is ready to for the data transfer.
- The 8257 sends Hold Request signal (HRQ) high to the processor. It then enables the particular channel.
- In response to HRQ signal, the processor will relinquishes the system busses in the next cycle and will send a Hold Acknowledge (HLDA) signal to the 8257.
- In response to this HLDA signal from the microprocessor, the DMA controller will generate ($\overline{DACK}$) DMA acknowledge signal (active low) through its Control Logic block as an acknowledgement to the requesting peripheral. At the same time the 8257 enables the Address Enable (AEN) signal which disables systems address lines (A0-A7). These address lines becomes the output lines. The low order byte of the memory location is placed of these lines. The address strobe (ADSTB) signal goes high when AEN is high and places higher byte of the memory location generated by 8212 on the address bus A_{15}-A_8.
- The data transfer continues till the terminal count is reached.

12.3 DMA INTERFACING CIRCUIT

Figure 12.10 shows the interfacing circuit of DMA controller 8257. In this circuit

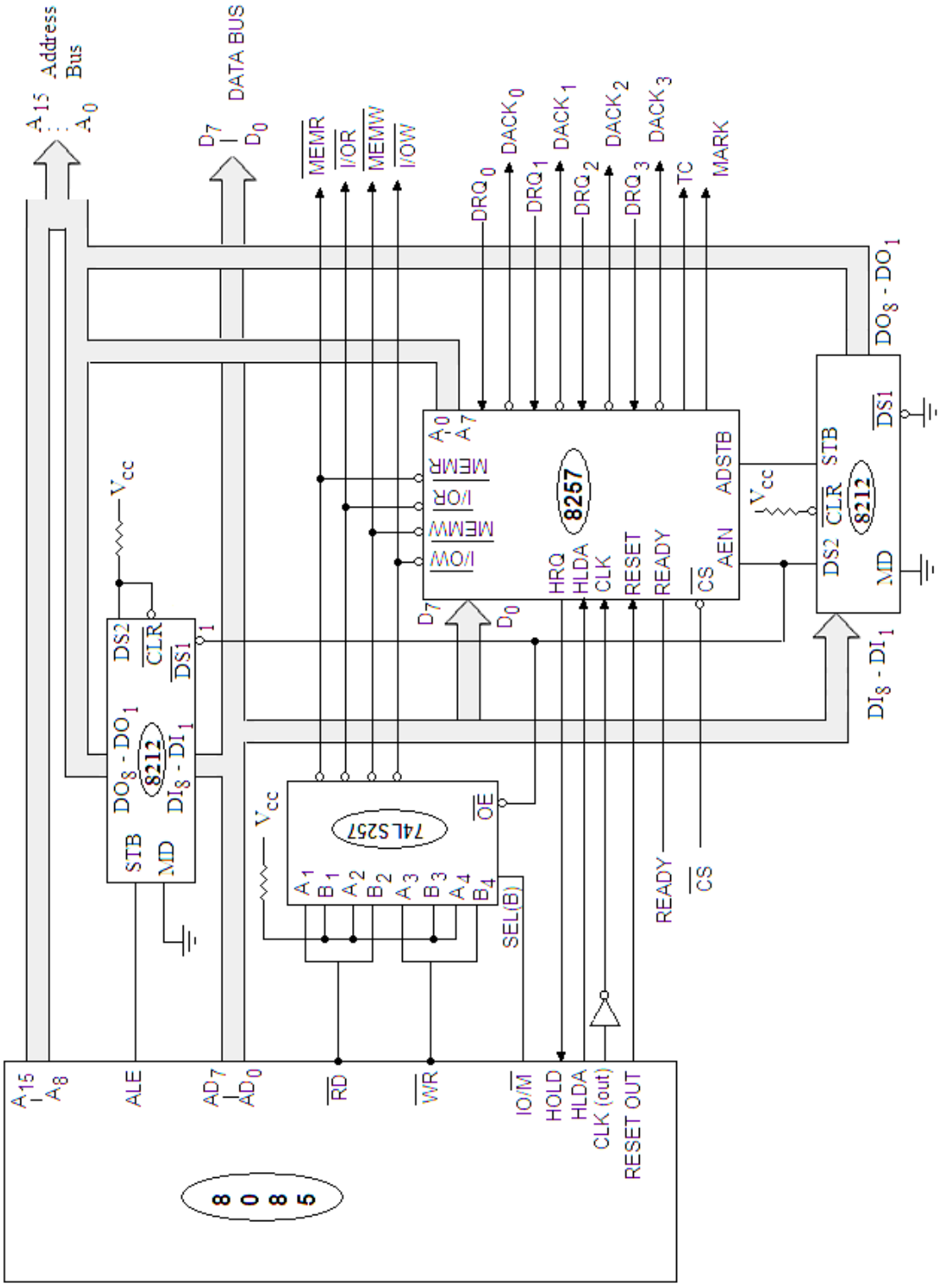

Address Bus
DATA BUS
MEMR
I/OR
MEMW
I/OW
DRQ0
DRQ1
DRQ2
DRQ3
DACK0
DACK1
DACK2
DACK3
TC
MARK
8257
8212
74LS257
8085
ALE
RD
WR
IO/M
HOLD
HLDA
CLK (out)
RESET OUT
READY
CS
Vcc

Fig. 12.10

8212 is used to de-multiplex the 8085 bus to generate the low order address bus (A_7-A_0). The 8257 has eight address lines, but requires sixteen address lines to address a memory location. The additional eight lines are generated by using the signal ADSTB to strobe a high order memory address into the 8212 from the data bus. This IC is not related to 8085 not to the DMA controller 8257, but only $\overline{DS1}$ is controlled by AEN signal of 8257. The IC 74LS257 is a multiplexer which is used to generate control signals. The $\overline{OE}$ terminal of this multiplexer is also controlled by AEN. The AEN output signal is used to disable (float) the system bus and control bus. This signal is also necessary to switch the 8257 from the slave mode to the master mode. The port addresses of the registers of 8257 are used as discussed above and circuit for the same is given in figure 12.9. The port addressed may be chosen as desired. The second 8212 connected to the DMA controller accepts the data during the DMA operation.

Example 12.1 *Initialize 8257 DMA controller to transfer 2K bytes of data stored in memory locations starting at 2101 H to floppy disk connected to the channel 1 of 8257. Assume that the address of the Mode Set Register is 78 H. The addresses of DMA address register for channel 1 and the Terminal Count Register are 72 H and 73 H respectively.*

Solution. The format for Mode Set register is given below:

D_7	D_6	D_5	D_4	D_3	D_1	D_0	
0	1	0	0	0	1	0	= 42 H
	↓						↓
	Enable						Enable CH 1
	TC stop						

Format for Terminal Count Register is given below:

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	= 87FF H

D_{15} and D_{14} as 10 indicate it is memory read operation and D_{10} to D_0 as 11111111111 represents 2 K bytes of data to be transferred to the floppy.

For the initialization of 8257, following steps are to be followed:

- Load mode word in the Mode Set Register.
- Load counts to the terminal count register first LS byte and then MS byte.
- Load starting address of the memory location, from where the data is to be transferred to the floppy disk, to the DMA address register.

The initialization program is therefore given as:

```
MVI A, 42 H      ; Load mode word in the Mode Set Register.
OUT 78 H         ; Address of the Mode Set Register.
MVI A, FF H      ; Load  LS byte of the count in the terminal
                   count register.
OUT 73 H         ; Address of TC register.
```

```
MVI A, 87 H      ; Load MS byte of the count in the terminal
                   count register.
OUT 73 H         ; Address of TC register.
MVI A, 00 H      ; Load the LS byte of the starting address of
                   the memory location from where the data
                   is to be transferred.
OUT 72 H         ; Address of the DMA address register.
MVI A, 21 H      ; Load the MS byte of the starting address of
                   the memory location.
OUT 72 H         ; Address of the DMA address register.
```

Example 12.2 *Initialize 8257 DMA controller to transfer 512 bytes of data from a peripheral to memory locations starting at 3000 H through channel 0. Assume that the address of the Mode Set Register is 28 H. Assume the addresses of DMA address register for channel 0 and the Terminal Count Register are 20 H and 21 H respectively.*

Solution. The format for Mode Set register is given below:

D_7	D_6	D_5	D_4	D_3	D_1	D_0	
0	1	0	0	0	0	1	= 41 H

Enable TC stop

Enable CH 0

Format for Terminal Count Register is given below:

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	= 41FF H

D_{15} and D_{14} as 01 indicate it is memory write operation and D_8 to D_0 as 111111111 represents 512 bytes of data to be transferred.

The initialization program is therefore given as:

```
MVI A, 41 H      ; Load mode word in the Mode Set Register.
OUT 28 H         ; Address of the Mode Set Register.
MVI A, FF H      ; Load  LS byte of the count in the terminal
                   count register.
OUT 21 H         ; Address of TC register.
MVI A, 41 H      ; Load MS byte of the count in the terminal
                   count register.
OUT 21 H         ; Address of TC register.
MVI A, 00 H      ; Load the LS byte of the starting address of
                   the memory location from where the data
                   is to be transferred.
OUT 20 H         ; Address of the DMA address register.
MVI A, 30 H      ; Load the MS byte of the starting address of
                   the memory location.
```

```
OUT 20 H          ; Address of the DMA address register.
```

Example 12.3 *An I/O device which is associated with channel 2 of 8257 DMA controller is to be periodically refreshed with 4K bytes of data starting from 2200 H locations. Write the initialization routine for the DMA controller to be operated with auto-load and extended write operations. The addresses of DMA registers and terminal count registers of all the four channels along with the address of the mode set registers are given in table 12.3.*

Solution. The format for Mode Set register is given below:

D_7	D_6	D_5	D_6	D_5	D_4	D_3	D_1	D_0	
1	0	1	0	0	1	1	0	0	= A6 H

D7 = 1 indicates that the auto load enabled, and

D5 = 1 indicates the extended write enabled.

D4 and D3 = 11, indicate that the channel 2 and channel 3 are enabled, which are needed for the auto load and extended write mode.

Format for Terminal Count Register is given below:

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	= 8FFF H

D_{15} and D_{14} as 10 indicate it is memory read operation and D_{11} to D_0 as 111111111111 represents 4096 (4K) bytes of data to be transferred.

In this case the starting address of the memory location is to be loaded to both DMA address registers of both channel 2 and channel 3 for auto load and extended write mode.

The initialization program is therefore given as:

```
MVI A, A6 H       ; Load mode word in the Mode Set Register.
OUT 78 H          ; Address of the Mode Set Register.
MVI A, FF H       ; Load  LS byte of the count in the terminal
                    count register of channel 2.
OUT 75 H          ; Address of TC register of channel 2.
MVI A, 8F H       ; Load MS byte of the count in the terminal
                    count register of channel 2.
OUT 75 H          ; Address of TC register of channel 2.
MVI A, FF H       ; Load  MS byte of the count in the terminal
                    count register of channel 3.
OUT 77 H          ; Address of TC register of channel 3.
MVI A, 8F H       ; Load MS byte of the count in the terminal
                    count register of channel 3.
OUT 77 H          ; Address of TC register of channel 3.
MVI A, 00 H       ; Load the LS byte of the starting address of
                    the memory location.
```

```
OUT 74 H        ; Address of the DMA address register
                  channel 2.
MVI A, 22 H     ; Load the MS byte of the starting address of
                  the memory location.
OUT 74 H        ; Address of the DMA address register ch.2.
MVI A, 00 H     ; Load the LS byte of the starting address of
                  the memory location.
OUT 76 H        ; Address of the DMA address register
                  channel 3.
MVI A, 22 H     ; Load the MS byte of the starting address of
                  the memory location.
OUT 76 H        ; Address of the DMA address register ch.3.
```

PROBLEMS

1. What do you understand by Direct Memory Access? What is the use of DMA controller for the data transfer from memory to I/O devices or vice-versa?
2. Draw the block diagram of 8257 DMA controller. Discuss the functions of it blocks.
3. How many I/O devices can be connected with the 8257? How many register are there in 8257? Discuss these registers.
4. Give the format of terminal count register. Discuss the functions of the bits of TC register.
5. What is the function of the Read/ Write Logic block of the 8257 DMA controller?
6. Mention the function of each signal of the control logic block of 8257 DMA controller.
7. What is the function of Mode Set Register of the 8257? How 8257 is programmed in Auto Load option?
8. How 8257 is programmed in Rotating Priority option?
9. Give and discuss the format of the Status Word Register of 8257.
10. How the programming of 8257 DMA controller is done?
11. Give the address decoder circuit for addresses of various registers as C0 H to C8 H. The decoder circuit enables chip select terminal ($\overline{CS}$) of the 8257.
12. What steps are carried out for initialization of 8257?
13. Draw and discuss the interfacing circuit of 8257 with 8085A microprocessor.
14. Initialize 8257 DMA controller to transfer 4K bytes of data stored in memory locations starting at 2501 H to an output device connected to the channel 2 of 8257. Assume that the address of the Mode Set Register is C8 H. The addresses of DMA address register for channel 2 and the Terminal Count Register are C4 H and C5 H respectively.

15. Initialize 8257 DMA controller to transfer 200 bytes of data stored in memory locations starting at 2000 H to an output device connected to the channel 0 of 8257. Assume that the address of the Mode Set Register is 28 H. The addresses of DMA address register for channel 0 and the Terminal Count Register are 20 H and 21 H respectively.

16. The DMA controller 8257 should be initialized as follows:

    DMA channel 2 must be enabled and terminal count stop bit is to be enabled.

    2500 H must be written in DMA address register channel 2.

    500 (decimal) must be written in Terminal counter of channel 2 and D14 and D15 are set for memory read operation.

    Assume the address of the DMA address register for channel 2 is 94 H and the terminal count is 95 H; and the address of the mode set register is 98 H.

17. Initialize 8257 DMA controller to transfer 8K bytes of data from a peripheral to memory locations starting at 21FF H through channel 1. Assume that the address of the Mode Set Register is 98 H. Assume the addresses of DMA address register for channel 1 and the Terminal Count Register are 92 H and 93 H respectively.

18. An I/O device which is associated with channel 2 of 8257 DMA controller is to be periodically refreshed with 1K bytes of data starting from 2500 H locations. Write the initialization routine for the DMA controller to be operated with auto-load and extended write operations. The addresses of DMA registers and terminal count registers of all the four channels are C0 H to C7 H; and the address of the mode set register is CH H.

13

Interfacing Data converters: A/D and D/A Converters

Sometimes the information available for processing in microprocessor based system is in digital form while in most of the cases it is available in analog form. For example, the outputs of digital voltmeter, digital frequency meter, digital clock and calculators etc. are available in digital form but most physical quantities such as temperature, pressure, light, voltage and current etc. gives information in analog form. It is often necessary to convert information in one form to another form for the purpose of interfacing with the system. For example, to design the microprocessor base temperature controller, the temperature of the device obtained from the transducer such as thermocouple or thermister is first converted to the digital form using D/A converter then interfaced with the microprocessor. Similarly, for plotting the output of a system on a curve plotter or X-Y recorder, the digital output is first converted to analog output with the help of digital to analog converter, the output of which drives a servomotor. So analog to digital (A/D) converters or digital to analog (D/A) converters are the interfacing devices with the system. In this chapter various types of A/D and D/A converters and their interfacing with the microprocessor will be discussed.

13.1 DIGITAL TO ANALOG CONVERTER

Digital to Analog (D/A) converter converts the digital information into analog form. The input may be of *n*-bit long having different voltage levels. So in the D/A converters, some method is to be used which can convert this voltage level of *n*-bits to its equivalent analog form. This can be accomplished by using different resistive networks. Following two types of resistive networks are basically used for this purpose:

1. Resistive Divider Network or weighted resistor network
2. Binary Ladder Network or R-2R network

The converter which comprises the resistive divider network is known as Resistive Divider D/A converter and the D/A converter which comprises the binary ladder network is known as Binary Ladder D/A converter. These converters will now be discussed.

13.1.1 Resistive Divider D/A converter

As discussed above, the resistive divider D/A converter consists of a resistive divider network, so before discussing the complete circuit diagram of a resistive divider D/A converter, it is better to understand the working of resistive divider network. The resistive divider network changes each of the *n*-bit digital level into its equivalent analog

output. The discussion will now be made for the method of converting the *n*-bit digital input to its equivalent analog signal. A weight is assigned to each bit of *n*-bit digital input in such a way that the sum of weight must be equal to 1. In general, the binary weight assigned to LSB in an *n*-bit digital input is $\frac{1}{2^n - 1}$. The weights assigned to 2nd LSB, 3rd LSB, 4th LSB and so on are obtained by multiplying the weights of LSB to 2^1(=2), 2^2 (=4), 2^3 (=8)…. respectively. For instance, weights assigned to different bits of 4-bit binary input $b_3\ b_2\ b_1\ b_0$ are:

Weight assigned to LSB (b_0 bit) is $\frac{2^0}{2^4 - 1} = \frac{1}{15}$

Weight assigned to 2nd LSB (b_1 bit) is $\frac{2^1}{2^4 - 1} = \frac{2}{15}$

Weight assigned to 3rd LSB (b_2 bit) is $\frac{2^2}{2^4 - 1} = \frac{4}{15}$

Weight assigned to MSB (b_3 bit) is $\frac{2^3}{2^4 - 1} = \frac{8}{15}$

The sum of weights assigned to each bit of 4-bit digital input is 1 as $\frac{1}{15} + \frac{2}{15} + \frac{4}{15} + \frac{8}{15} = 1$.

In a four bit binary system there will be 16 different possible input combinations, corresponding to which the analog signal will be obtained if it is assumed that a certain reference voltage (V_{REF}) is applied whenever there is a 1 in binary bit. In a 4 bit digital system if V_{REF} =15 volts, the analog voltage available for each combination of binary input should be as given in table 13.1.

Table 13.1

b_3	b_2	b_1	b_0	Weight	Analog Voltage
0	0	0	0	0/15	(0/15)V_{REF} = 0 Volt
0	0	0	1	1/15	(1/15) V_{REF} = 1 Volt
0	0	1	0	2/15	(2/15) V_{REF} = 2 Volt
0	0	1	1	3/15	(3/15) V_{REF} = 3 Volt
0	1	0	0	4/15	(4/15) V_{REF} = 4 Volt
0	1	0	1	5/15	(5/15) V_{REF} = 5 Volt
0	1	1	0	6/15	(6/15) V_{REF} = 6 Volt
0	1	1	1	7/15	(7/15) V_{REF} = 7 Volt
1	0	0	0	8/15	(8/15) V_{REF} = 8 Volt
1	0	0	1	9/15	(9/15) V_{REF} = 9 Volt
1	0	1	0	10/15	(10/15) V_{REF} = 10 Volt
1	0	1	1	11/15	(11/15) V_{REF} = 11 Volt
1	1	0	0	12/15	(12/15) V_{REF} = 12 Volt
1	1	0	1	13/15	(13/15) V_{REF} = 13 Volt
1	1	1	0	14/15	(14/15) V_{REF} = 14 Volt
1	1	1	1	15/15	(15/15) V_{REF} = 15 Volt

So the analog voltage for binary word = (weight of the binary word) x V_{REF}

It may be noted from this table 13.1 that the analog voltage corresponding to binary equivalent is discrete step value as given in figure 13.1. The discrete step is of 1 volt if V_{REF} is assumed to be 15 volts in a four bit digital input. The step voltage (analog) will be dependent on the reference voltage. There will, however, be 2^n steps in *n*-bit digital system.

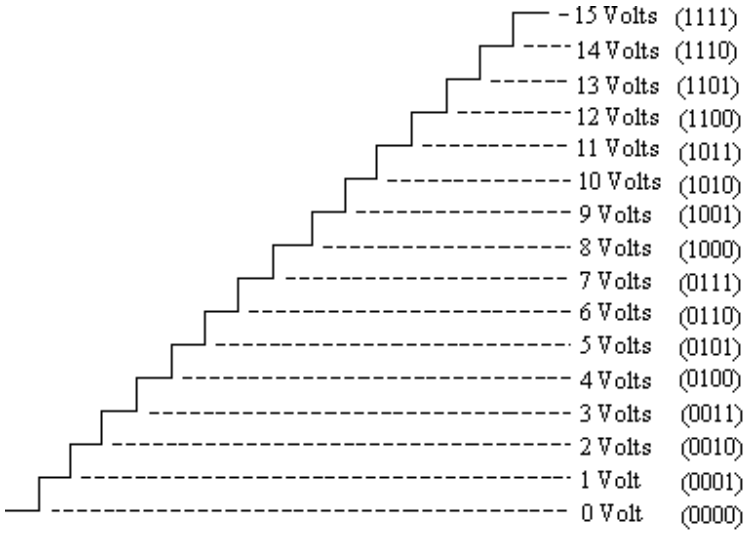


Fig. 13.1

Resistive divider network is used for converting digital inputs to analog outputs. The network for 6 bit binary system shown in figure 13.2 is known as the weighted network, as the resistors are weighted inversely with their current values. The input binary bits are b_5 b_4 b_3 b_2 b_1 b_0 where b_0 is the LSB and b_5 is MSB. These binary bits may be logic 0 or 1. Logic 0 may further be assumed as 0 volt and logic 1 as V_{REF}. So V_0, V_1, V_2, V_3, V_4 and V_5 are the input voltage levels which may be 0 volt or V_{REF} depending on the binary bits. The resistors R_0, R_1, R_2, R_3, R_4 and R_5 are connected to bits b_0, b_1, b_2 , b_3, b_4, b_5 respectively. It may be noted from this network that the resistor connected to the binary bit is half the value of resistor connected to the previous (lower) bit. Hence this network also called as the resistive divider network. Let R_L is the load resistance which is supposed to very high i.e. very much higher than the resistor R_0.

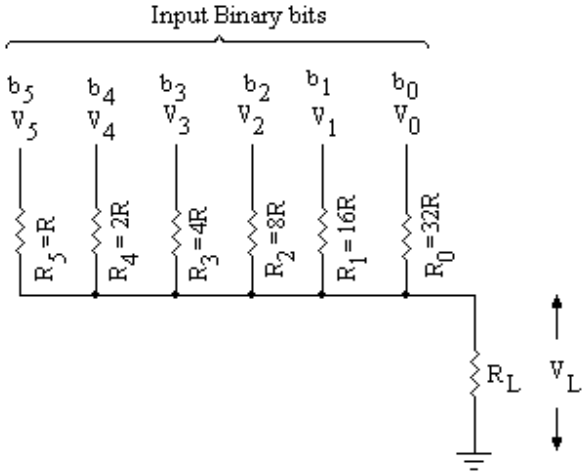


Fig. 13.2

Now the voltage V_L across the load resistance R_L can be obtained by using Millman's theorem. This theorem states that the voltage appearing at any node in a resistive network is equal to the sum of all the currents that would enter to the node divided by the sum of conductances connected to that node.

Thus

$$V_L = \frac{\frac{V_5}{R_5}+\frac{V_4}{R_4}+\frac{V_3}{R_3}+\frac{V_2}{R_2}+\frac{V_1}{R_1}+\frac{V_0}{R_0}}{\frac{1}{R_5}+\frac{1}{R_4}+\frac{1}{R_3}+\frac{1}{R_2}+\frac{1}{R_1}+\frac{1}{R_0}}$$

$$= \frac{\frac{V_5}{R}+\frac{V_4}{2R}+\frac{V_3}{4R}+\frac{V_2}{8R}+\frac{V_1}{16R}+\frac{V_0}{32R}}{\frac{1}{R}+\frac{1}{2R}+\frac{1}{4R}+\frac{1}{8R}+\frac{1}{16R}+\frac{1}{32R}}$$

$$= \frac{\frac{[32V_5+16V_4+8V_3+4V_2+2V_1+V_0]}{32}}{\frac{[32+16+8+4+2+1]}{32}}$$

$$= \frac{V_0+2V_1+4V_2+8V_3+16V_4+32V_5}{63}$$

$$= \frac{1}{(2^6-1)}(2^0V_0+2^1V_1+2^2V_2+2^3V_3+2^4V_4+2^5V_5) \quad \ldots(13.1)$$

In this equation (13.1), the load resistance R_L is not considered as it is assumed to be large enough offering low (almost zero) conductance. From this equation it is clear that if the input binary bits are all 1 (in a six bit system) and reference voltage V_{REF} = 6.4 volts (say), the V_L is given by:

$$V_L = \frac{1}{63} x 63 V_{REF} = 6.4 volts$$

In general, the equation (13.1) for output voltage of *n*-bit binary digits is given as:

$$V_L = \frac{1}{(2^n - 1)}(2^0 V_0 + 2^1 V_1 + 2^2 V_2 + 2^3 V_3 + 2^4 V_4 ..... + 2^{n-1} V_{n-1})$$

…(13.2)

The output of this network is as per our requirement, and is proportional to the input binary data.

Using the network discussed above, a D/A converter (called binary weighted D/A converter or Resistive divider D/A converter) can be designed as given below. The schematic diagram of 6-bit D/A converter is shown in figure 13.3. It consists of the following major parts:

(i) n switches, one for each bit applied to the input,

(ii) A binary weighted resistive network which changes each of the digital level into equivalent binary weighted voltage or current.

(iii) A reference voltage source V_{REF}.

(iv) A summing amplifier that adds the currents flowing in the resistors of the network to develop a signal that is proportional to the digital input.

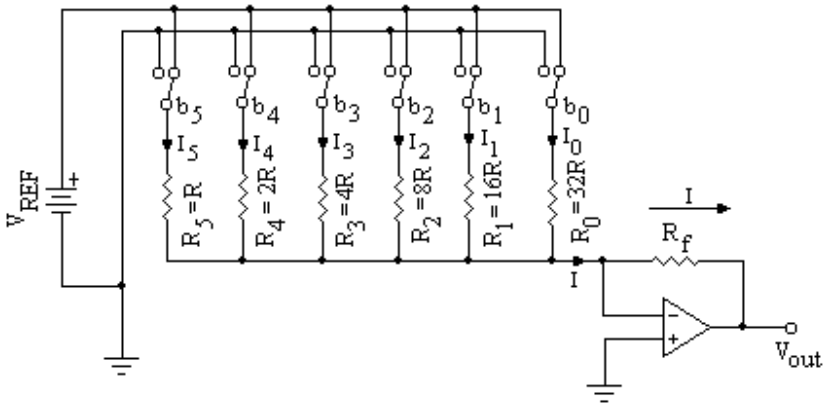


Fig. 13.3

In this circuit, one switch is connected to each binary bit. In fact these switches are such that when the binary bit is 0, the corresponding resistor of the network gets connected to the ground potential and when the binary bit is 1, the corresponding resistor of the network gets connected to the V_{REF} volt. The current flowing through any branch of the network will be the logical voltage (0volt or V_{REF} volts) divided by the corresponding resistor.

So the total current I will be given by (ref. fig. 11.3):

$$I = \frac{V_5}{R_5} + \frac{V_4}{R_4} + \frac{V_3}{R_3} + \frac{V_2}{R_2} + \frac{V_1}{R_1} + \frac{V_0}{R_0}$$

$$= \frac{V_5}{R} + \frac{V_4}{2R} + \frac{V_3}{4R} + \frac{V_2}{8R} + \frac{V_1}{16R} + \frac{V_0}{32R}$$

Since the voltages V_5 through V_0 are either 0 or V_{REF} volts depending upon the bit value, so it is customary to take common voltage V_{REF} and bits are kept in place of voltages. So V_5 is replaced by $V_{REF}.b_5$, V_4 by $V_{REF}.b_4$ and so on; the bits b_5, b4, b3 etc will be 0 or 1. The current I may, therefore, be represented as follows:

$$I = \frac{V_{REF}}{32R}\left[32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0\right]$$

$$= \frac{V_{REF}}{2^5.R}\left[2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3 + 2^4 b_4 + 2^5 b_5\right]$$

This is the equation of current I for 6 input bits. The general equation of current I for n input bits is given by:

$$I = \frac{V_{REF}}{2^{n-1}.R}\left[2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3 + 2^4 b_4 ..... + 2^{n-1} b_{n-1}\right] \quad \ldots(13.3)$$

The voltage at the output of operational amplifier will be given by:

$$V_{out} = -R_f.I$$

The resistor R_f is the feed back resistance in the operational amplifier. The output voltage of the operational amplifier is proportional to input binary data.

The switches connected in figure 13.3 can be replaced by the electronic switches (transistorized) as shown in figure 13.4. When the bit is at logic 1, the corresponding transistor conducts and the current flows through the collector resistor as required; and when the bit is at logic 0 the transistor goes into cutoff and no collector current flows.

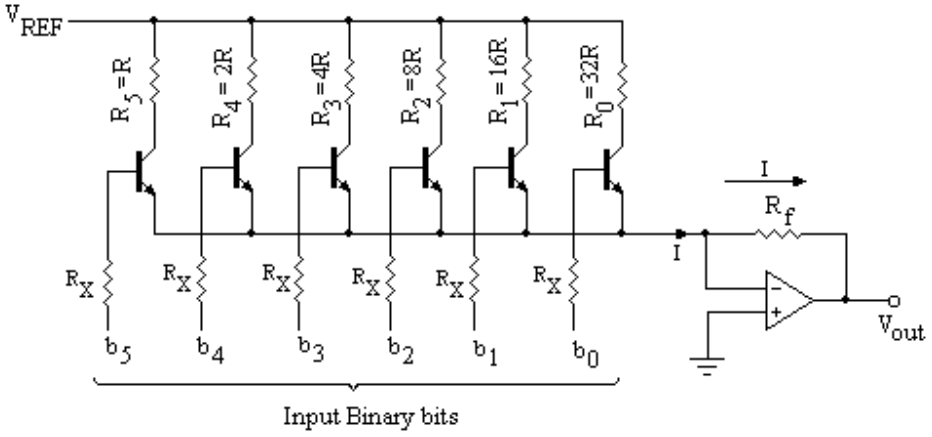


Fig. 13.4

This D/A converter is economical and simple method to design but suffers the following serious drawbacks:

1. The network in this D/A converter is constructed using the precession resistors and resistors have different values. So it is difficult in practice to choose the resistors with accuracy and stability.
2. When the number of bits in the network is large, then the current from the source will be large enough. The current in the MSB branch (resistor) will be much larger than LSB branch. In a 10 bit D/A converter, the current in MSB branch will be 512 times larger than the MSB branch.

13.1.2 Binary Ladder D/A Converter

A more commonly used D/A converter is a binary ladder D/A converter, which removes the drawbacks discussed in resistive divider D/A converter. This type of D/A converter contains an R-2R ladder network. The R-2R resistive ladder network will now be discussed, which gives the output a weighted sum of digital inputs. Such a ladder

network for 4-bit input is shown in figure 13.5. This network is constructed having only two resistor values i.e. R and 2R. In this network b_0, b_1, b_2 and b_3 are the input binary bits and b_0 is the LSB and b_3 is MSB. Any of these bits will be at the ground potential when the corresponding bit is at logic 0 or at the reference potential (V_{REF}) when the input bit is at logic 1.

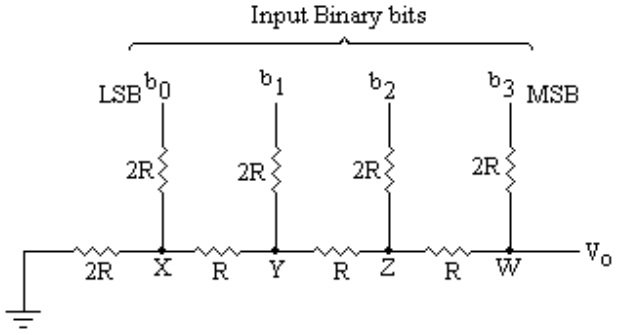
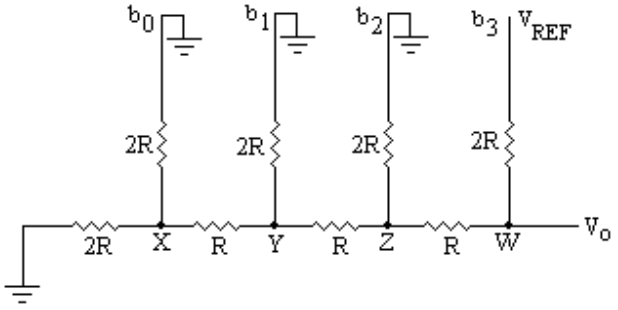
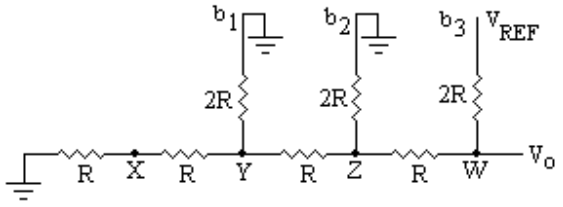


Fig. 13.5

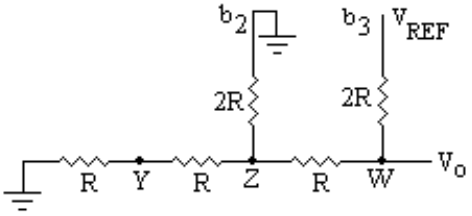
To examine the behaviour of this network, it is assumed that the bit b_3 is at logic 1 (or V_{REF} potential) as shown in figure 13.6(a). The output voltage corresponding to MSB may be calculated as follows. The equivalent resistance at the point X is the parallel combination of two resistances each having the value of 2R. So the equivalent resistance looking at point X and ground is R as shown in figure 13.6(b). At the point Y again there is a parallel combination of two 2R resistances; the equivalent resistance looking at the point Y and ground is R as shown in figure 13.6(c). Similarly, one can find the equivalent resistance looking at the point Z and ground is R as shown in figure 13.6(d).



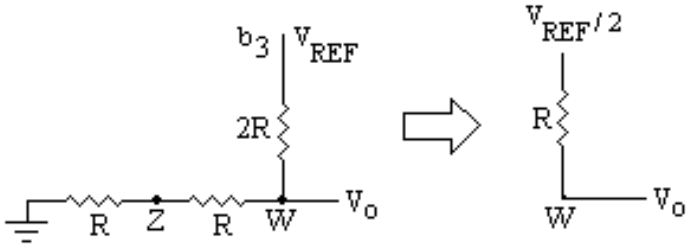
(a)



(b)



(c)



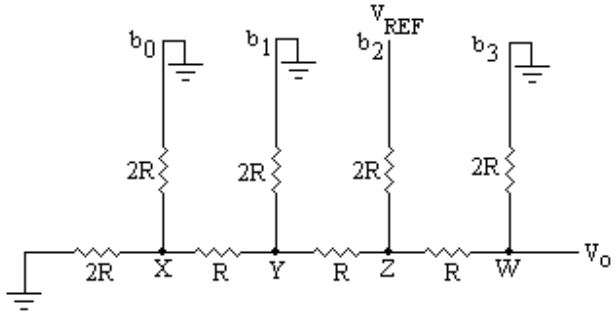
(d)

Fig. 13.6

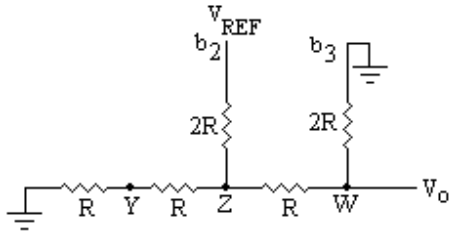
From figure 13.6(d) it is clear that the resistance looking at the point W and ground is 2R, and the resistance looking towards the bit b_3 is also 2R. Thus the output voltage at the point W due to bit b_3 (MSB) assumed at V_{REF} potential is given by:

$$V_0 = \frac{V_{REF} x 2R}{(2R + 2R)} = \frac{V_{REF}}{2}$$

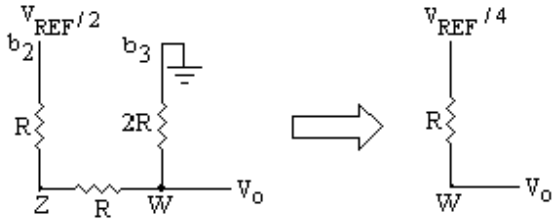
The output voltage V_0 due to the binary input 1000 (only MSB is high) is half of the reference voltage having Thevenis's resistance R in series with it. Similarly one can calculate the output voltage due to the binary input 0100 (i.e. second MSB); the network for this case is shown in figure 13.7(a). The resistance looking at the point Y and ground is R as shown in figure 13.7(b). The resistance between the point Z and ground is 2R. The voltage at point Z and ground is (V_{REF}/ 2) have a Thevenin's resistance R, as shown in figure 13.7(c).



(a)



(b)



(c)

Fig. 13.7

From this figure the output voltage V_0 at the point W is given by:

$$V_0 = \frac{(V_{REF}/2)x2R}{(2R+2R)} = \frac{V_{REF}}{4}$$

So the output voltage due to second MSB (or for binary input 0100) is $\frac{V_{REF}}{4}$ with Thevenin's resistance R in series with it.

It can further be shown that the output due to third MSB (for binary input 0010) is $\frac{V_{REF}}{8}$. And for LSB (0001 binary input) the output is $\frac{V_{REF}}{16}$. Each voltage source will have Thevenin's resistance R in series with the source. The total output voltage in analog form, due to all the inputs as 1 (for 1111) can easily be found by adding the outputs obtained for each bit as given below:

$$V_0 = \frac{V_{REF}}{2} + \frac{V_{REF}}{4} + \frac{V_{REF}}{8} + \frac{V_{REF}}{16}$$

It may be noted that $\frac{V_{REF}}{2}$ is the voltage due to MSB, $\frac{V_{REF}}{4}$ due to second MSB, $\frac{V_{REF}}{8}$ for third MSB and $\frac{V_{REF}}{16}$ for LSB. So to distinguish these voltages it is useful to write the bit positions along with V_{REF} as given below. So if the bit is 0 the voltage corresponding to that bit will be zero otherwise the voltage as discussed above.

$$V_0 = \frac{V_{REF}xb_3}{2} + \frac{V_{REF}xb_2}{4} + \frac{V_{REF}xb_1}{8} + \frac{V_{REF}xb_0}{16}$$

$$= \frac{V_{REF}}{16}\left[8xb_3 + 4xb_2 + 2xb_1 + 1xb_0\right]$$

$$= \frac{V_{REF}}{2^4}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3\right] \quad \ldots(13.4)$$

The equation (13.4) is the equation for voltage at the output of 4 bit binary ladder network. A general equation for the output of *n*-bit binary data can be given as follows:

$$V_0 = \frac{V_{REF}}{2^n}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3 + ... + 2^{n-1} xb_{n-1}\right] \quad \ldots(13.5)$$

The output of this network is proportional to the input binary data. So using this R-2R ladder network, a D/A converter (called binary ladder D/A converter) can be designed as given below. The schematic diagram of 4-bit D/A converter is shown in figure 13.8. It consists of the following major parts.

(i) n switches, one for each bit applied to the input,
(ii) A binary ladder network which changes each of the digital level into equivalent binary weighted voltage or current.
(iii) A reference voltage source V_{REF}.
(iv) A summing amplifier that adds the currents flowing in the resistors of the network to develop a signal that is proportional to the digital input.

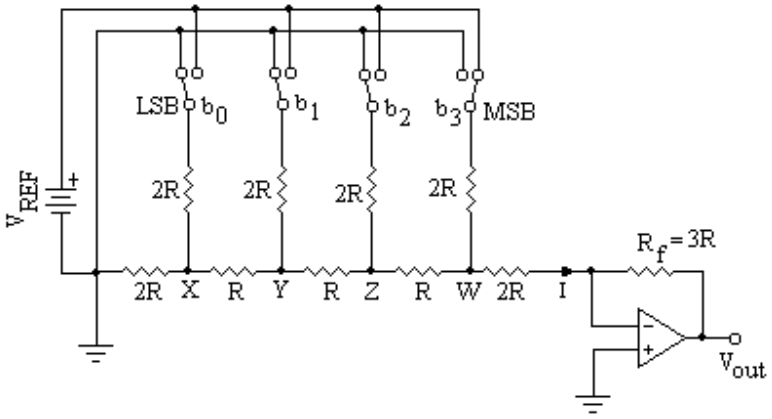


Fig. 13.8

The output voltage V_{out} of this D/A converter due to MSB (1000 binary input) will be calculated as given below:

The voltage at the point W due to MSB is V_{REF} / 2 having a Thevenin's resistance R in series with it as discussed above and is shown in figure 13.9

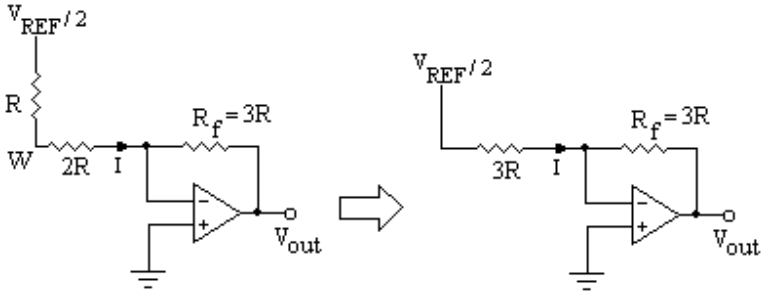


Fig. 13.9

From this figure, the current I is given by:

$$I = \frac{V_{REF}}{2}(\frac{1}{3R})$$

and the output voltage V_{out} is given by:

$$Vout = -I.R_f$$

$$= -\frac{V_{REF}}{2}(\frac{1}{3R}).3R = -\frac{V_{REF}}{2}$$

The output voltage is the same as calculated in equation 13.5, with the difference that it has a negative value because the operational amplifier is used in inverting configuration.

Note that the resistors in the ladder network are either R or 2R. It is the ratio of resistances matters rather than the absolute value of resistances. Further the resistors do not cover a wide range of magnitude; it is, therefore, practically possible to get the precision in the ratio of their magnitudes. The temperature coefficients of these resistances can easily match. Because of these advantages, the ladder network is widely used in D/A converters.

13.2 PERFORMANCE CRITERIA FOR D/A CONVERTER

The D/A converters are available in the form of ICs with different specifications for their performances. So before discussing D/A converter ICs it will be better to discuss first the characteristics of the converters specified by the manufacturers. These specifications include:

1. Resolution
2. Accuracy
3. Monotonicity
4. Settling time

1. **Resolution:** As discussed above, the analog output of D/A converter is proportional to the digital input (binary data), so a perfect staircase is obtained if there is an LSB increment. The resolution is, therefore, a measure of quality of D/A converter, which is defined as the ratio of the LSB increment to the maximum output. For an *n*-bit D/A converter the resolution is given by:

The change in output due to LSB increment for n-bit digital input (Step size)

= Full scale output / No. of steps

$$= \frac{\text{Full scale output}}{2^n - 1}$$

where $(2^n - 1)$ is the number of steps for n-bit D/A converter.

$$\text{Percentage Resolution} = \frac{\text{Full scale output} / (2^n - 1)}{\text{Full scale output}} \times 100\%$$

$$= \frac{1}{2^n - 1} \times 100\%$$

The step size for a 10 bit D/A converter, having full scale output voltage as 10 volts, is given by $= \frac{10}{2^{10} - 1} = \frac{10}{1023} = 9.8mV$

And % Resolution = 0.0978%

2. Accuracy: Accuracy of a D/A converter is the closeness of the output analog voltage to the expected theoretical output. In a linear variation of analog output with digital input, the relative accuracy is the maximum deviation of the D/A output compared with the linear behaviour. It is expressed as a percent of a full-scale or maximum output voltage. For example, if a converter has a full scale output of 10 V and the accuracy is $\pm 0.1\%$, then the maximum error for any output voltage is (10V)(0.001) = 10 mV. Ideally, the accuracy should be at most ± ½ of an LSB.

For an 8 bit D/A converter, one LSB is $\frac{1}{256} = 0.0039 = 0.39\%$ of full scale. The accuracy should be approximately $\pm 0.2\%$.

3. Monotonicity: A D/A converter is said to be monotonic if it gives an analog output voltage which increases regularly and linearly with increase in input digital signal. Such a quality of the converter is called as monotonicity. In order to demonstrate monotonicity of a D/A converter, a counter output is given as digital input to a D/A converter and the analog output is displayed on the CRO. Monotonicity then requires that the output waveform should be a perfect staircase waveform with steps equally spaced and of same magnitude. If the steps are missing or have varying magnitude, the D/A converter is defective.

4. Settling Time: After the application of digital input to a D/A converter, it takes about few nanoseconds to microseconds to produce the correct output. So the settling time is defined as the time the converter takes to give an output to settle within ± ½ LSB of its final value. For example, if a D/A converter has a resolution of 10mV, the settling time is the measure of the time the converter takes to settle with in ± 5mV of its final value. Figure 13.10 illustrates the settling time in a D/A converter. The settling time is important because it places a limit on how fast one can change the digital input. The settling time depends on the stray capacitance, saturation delay time, and other factors.

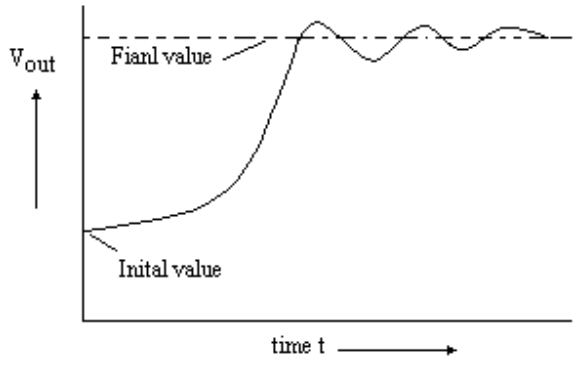


Fig. 13.10

13.3 D/A CONVERTER IC 0808

There are many commercially available D/A converter ICs. The IC 0808 is the most popular, inexpensive and widely used 8 bit D/A converter. It contains a reference current source, an R-2R binary ladder network and 8 transistor switches to steer the binary currents to the network. Figure 13.11 shows the pin configuration of this D/A converter IC 0808.

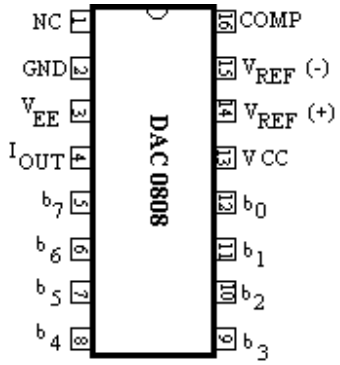


Fig. 13.11

In this IC, pins 5 through 12 are the 8 bit input data, so should be connected to input data bits. Pin 15 is to be connected to ground through a resistance. Pin 13 is to be connected to +5 volt supply. Pin 3 (V_{EE}) is to be connected to – 15 volts. Pin 4 is the output current of the ladder network should be connected to the operational amplifier. Pin 2 is the ground pin. The pin 16 is the frequency compensation pin, a capacitor between pin 16 and 3 is to be connected for this purpose.

A circuit diagram to get the analog output voltage corresponding to 8 bit digital input is shown in figure 13.12. A +5 V supply sets up a reference current of 2mA for the ladder. The output current I_{out} drives the operational amplifier to give final output between 0 and 2 volts (approximately) for the 8 bit digital input.

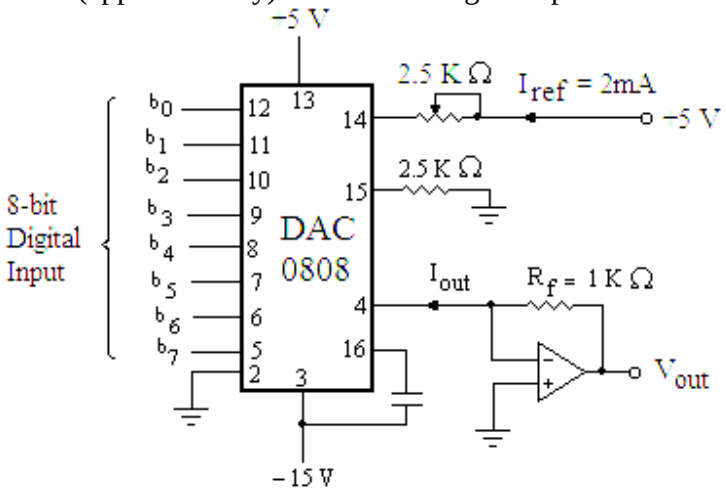


Fig. 13.12

There are many other commercially available D/A converter as given below:

DAC 0800 – A monolithic 8-bit high speed current output DAC.

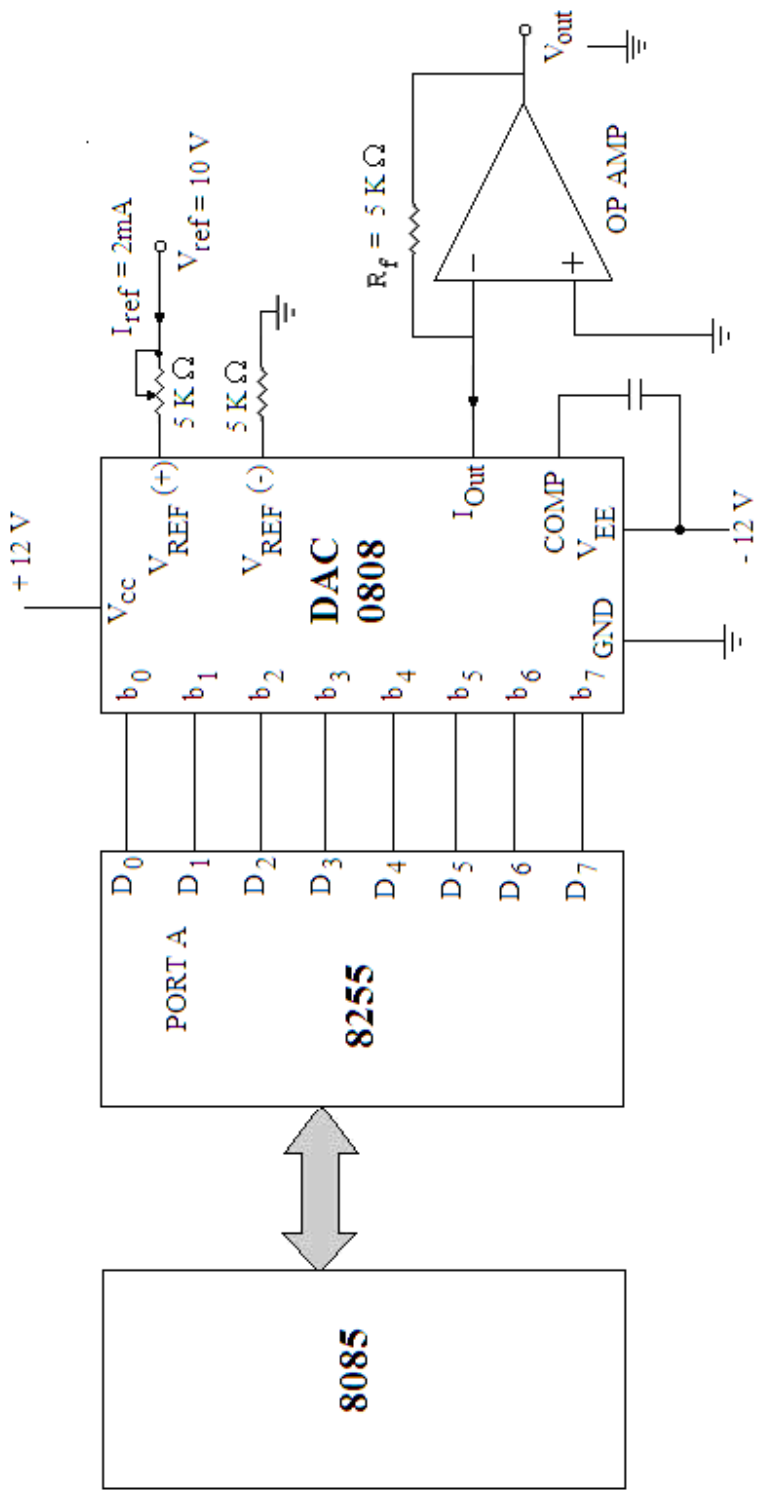
DAC 0806 and DAC 0807 – 8 bit monolithic D/A converters.

DAC 1000 and DAC 1008 – 10 bit microprocessor compatible advanced CMOS D/A converters.

DAC 1202 and DAC 1203 – Three-digit (BCD) D/A converter.

13.4 INTERFACING OF D/A CONVERTER

For the interfacing of 0808 D/A converter with the microprocessor 8085A, the circuit shown in figure 13.12 may be connected to the system through the 8255 PPI (Programmable Peripheral Interface as shown in figure 13.13.


8085
8255
PORT A
D0
D1
D2
D3
D4
D5
D6
D7
DAC
0808
Vcc
VREF (+)
VREF (-)
IOut
COMP
VEE
GND
b0
b1
b2
b3
b4
b5
b6
b7
+ 12 V
- 12 V
Iref = 2mA
Vref = 10 V
5 K Ω
5 K Ω
Rf = 5 K Ω
OP AMP
Vout

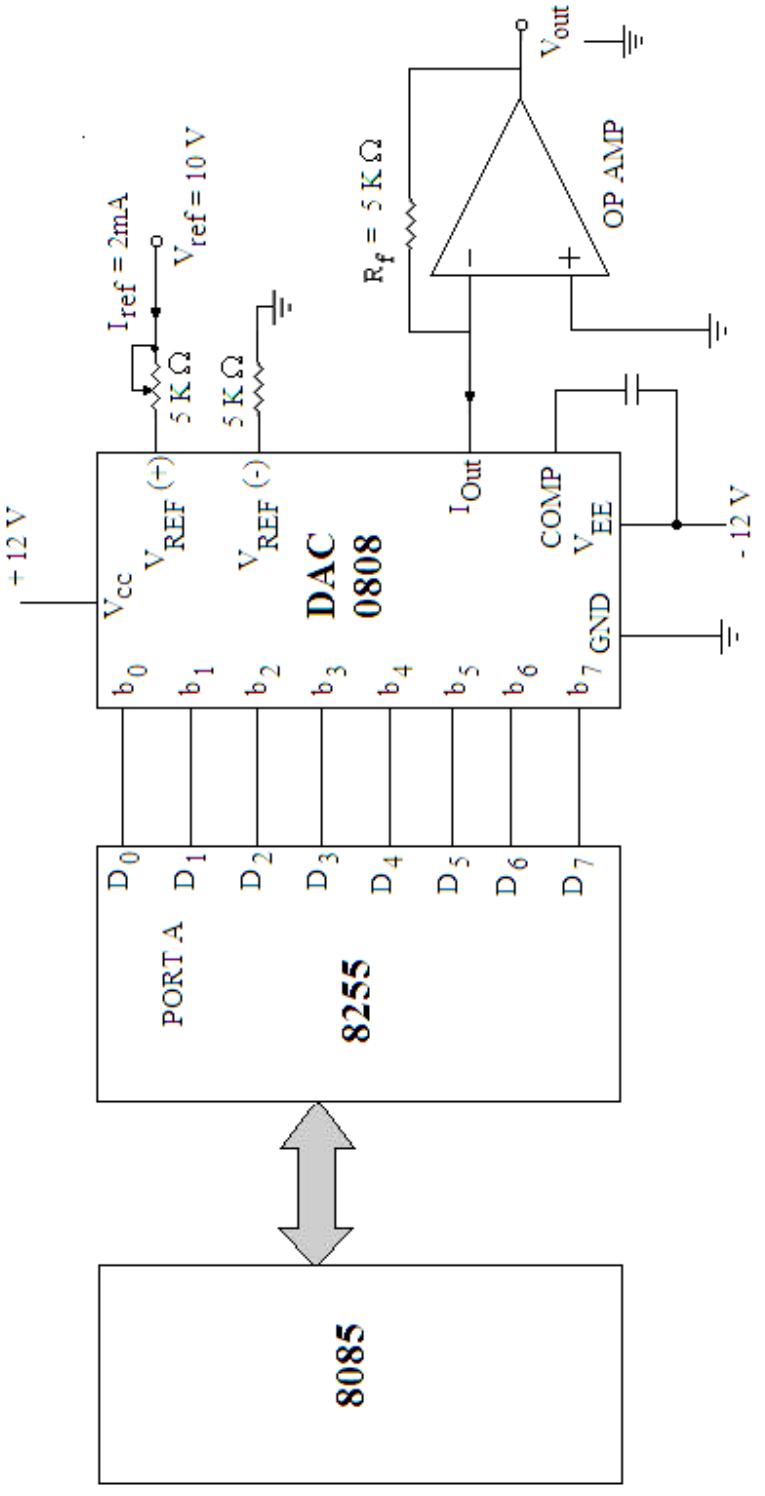

8085
8255
PORT A
D0
D1
D2
D3
D4
D5
D6
D7
DAC
0808
Vcc
VREF (+)
VREF (-)
IOut
COMP
VEE
GND
b0
b1
b2
b3
b4
b5
b6
b7
+ 12 V
- 12 V
Iref = 2mA
Vref = 10 V
5 K Ω
5 K Ω
Rf = 5 K Ω
OP AMP
Vout

Fig. 13.13

In this circuit feedback resistor R_f is considered as 5 KΩ and the reference voltage is taken as V_{REF} = 10 V, so that we get the output current I_{Out} as:

$$I_{Out} = \frac{I_{REF}}{2^n}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3 + ... + 2^{n-1} xb_{n-1}\right]$$

Here the value of n = 8 as it is 8 bit D/A converter.

So $$I_{Out} = \frac{I_{REF}}{256}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3 + ... + 2^{n-1} xb_{n-1}\right]$$

and $$I_{REF} = \frac{V_{REF}}{R_{REF}} = \frac{10V}{5K\Omega} = 2mA$$

and $$V_{OUT} = \frac{2mA}{256}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3 + ... + 2^{n-1} xb_{n-1}\right]xR_f$$

$$= \frac{10}{256}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3 + ... + 2^{n-1} xb_{n-1}\right]$$

If all the input bits (b_0 – b_7) are 1 (FF H), then the output voltage (known as full scale output voltage) is given by:

$$V_{OUT} = \frac{255}{256} x10 \approx 10V$$

The output voltage corresponding to the input 1000 0000 (80 H) is given by:

$$V_{OUT} = \frac{128}{256} x10 = 5V$$

So we have the linear output corresponding to the binary inputs i.e.

Digital Input	Output Voltage
00 H	0 V
80 H	5 V
FF H	10 V

These input output levels may be verified, if the following three programs are executed and the voltages at the outputs in the three cases are measured.

Program 1:

```
MVI A, 80 H     ; 8255 is initialized with all the ports as
                  output port.
OUT  03 H       ; Control word is written in control word
                  register.
MVI A, 00 H     ; Get A = 00 H, so that 00 H is applied to the
                  input of D/A converter.
OUT 00 H        ; 00 H is available at the Port A of 8255 so
                  that it is applied to the input of D/A
                  converter.
HLT             ; Stop processing.
```

After execution of this program, if the voltage at the output of the circuit of D/A converter is measured we get 0 V. It verifies that 00 H is applied at the input of the converter, we get 0 V.

Program 2:

```
MVI A, 80 H     ; 8255 is initialized with all the ports as
                  output port.
```

```
OUT  03 H        ; Control word is written in control word
                   register.
MVI A, 80 H      ; Get A = 80 H, so that 80 H is applied to the
                   input of D/A converter.
OUT 00 H         ; 80 H is available at the Port A of 8255 so
                   that it is applied to the input of D/A
                   converter.
HLT              ; Stop processing.
```

After execution of this program, if the voltage at the output of the circuit of D/A converter is measured we get 5 V. It verifies that 80 H is applied at the input of the converter, we get 5 V.

Program 3:

```
MVI A, 80 H      ; 8255 is initialized with all the ports as
                   output port.
OUT  03 H        ; Control word is written in control word
                   register.
MVI A, FF H      ; Get A = FF H, so that FF H is applied to
                   the input of D/A converter.
OUT 00 H         ; FF H is available at the Port A of 8255 so
                   that it is applied to the input of D/A
                   converter.
HLT              ; Stop processing.
```

After execution of this program, if the voltage at the output of the circuit of D/A converter is measured we get 10 V. It verifies that FF H is applied at the input of the converter, we get 10 V.

Example 13.1. *The input bits of the D/A converter is connected to the output pins of Port A of 8255, which is already connected with the microprocessor (ref. fig.13.13). Write a program to generate stair case voltage with ten steps. It should have the constant pulse duration.*

Solution. There should be 10 steps for the stair case voltage to be generated with the D/A converter. So the height of the stair case should be approximately 1 volt. Since FF H gives 10 volts, so 19 H should be decreased each time in 10 go.

When 19 H is subtracted from FF H in the first go we get E6 H. The output voltage corresponding to E6 H is given by:

$$V_{Out} = \frac{230}{256} x10 = 8.984 \text{ volts}$$

Therefore, the subtraction of 19 H from FF H gives a decrement of 1 volt at the output. The program for the generation of stair voltage is given as:

PROGRAM

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	80 H	; Initialize 8255-I to work all the ports as output ports.
	OUT	03 H	; Write the control word (80 H) in the control word register of 8255-I.
START	MVI A,	FF H	; Load FF H to accumulator.

```
                OUT         00 H        ; Send FF H (10 V) to PA0.
                CALL        DELAY       ; Jump to delay subroutine to
                                          introduce a delay of constant pulse
                                          width.
                MVI B,      00 H        ; Use B-register as counter for 10
                                          steps.
REPEAT          SBI         19 H        ; Subtract 19 H to calculate next
                                          weight for the output of D/A
                                          converter.
                OUT         00 H        ; Send the data to the output.
                INR B                   ; Increment B-register.
                CPI         0A H        ; If 10 steps complete then
                JZ          END         ; Jump to repeat the process.
                PUSH PSW                ; Save PSW
                PUSH B                  ; Save B-C register pair.
                CALL        DELAY       ; Jump to delay subroutine to
                                          introduce a delay of constant pulse
                                          width.
                POP B                   ; Restore the contents of B-C pair.
                POP PSW                 ; Restore the PSW.
                JUMP        REPEAT      ; Jump to repeat to output for the
                                          next weight.
END             MVI A,      00 H        ; Store 00 H to the accumulator.
                OUT         00 H        ; Outputs for 00 H.
                CALL        DELAY       ; Jump to delay subroutine to
                                          introduce a delay of constant pulse
                                          width.
                JMP         START       ; Repeat for next cycle.
```

SUBROUTINE PROGRAM:

```
    Label     Mnemonics     Operand     Comments
    DELAY     LXI D,        0020 H      ; Loads DE register pair with a 16-
                                          bit number.
    LOOP1     DCX D                     ; Decrements DE register pair by 1.
              MOV A,        E           ; Moves the contents of E register to
                                          accumulator.
              ORA D                     ; ORing of the contents of D and E
                                          registers are performed to set the
                                          zero flag.
              JNZ           LOOP1       ; If result is not zero than jump to
                                          LOOP1.
              RET                       ; Go back to main program.
```

In the subroutine program, DE register pair may be loaded with any other 16-bit number to change the pulse width. The exact time of the pulse width may be calculated as discussed in the delay programs. After the execution of this program, the output may be seen on the CRO. The output will be stair case wave.

Example 13.2. *Write a program to generate square wave of 1 KHz frequency with a peak voltage of 2.5 volts. Use D/A converter (ref. fig. 13.13) for this purpose. The square wave should be available at the output of the converter.*

Solution. It is required to generate square wave of 1 KHz frequency, the time period of such wave should be 1msec. During 1 msec the output should be high for 0.5 msec and low for the same time. Time delay can be introduced by using the following subroutine program:

SUBROUTINE PROGRAM:

Label	Mnemonics	Operand	Comments
DELAY	LXI D,	0040 H	;Loads DE register pair with a 16-bit number.
LOOP	DCX D		;Decrements DE register pair by one.
	MOV A,	E	;Moves the contents of E register to accumulator.
	ORA D		;ORing of the contents of D and E registers are performed to set the zero flag.
	JNZ	LOOP	;If result $\neq 0$ jump to loop
	RET		;Go back to main program.

Total T-states used for the above sub routine program are given as:

Mnemonics	T-states
LXI	10
DCX	5
MOV	5
ORA	4
JNZ LOOP	10/7
RET	10

24 T-states are used for the inner loop and 10+7+10 = 27 T-states are used for outer loop.

In this program the execution of loop is for 64 times (as 0040 H = 64_{10}). The condition for the check of zero flag can not be applied just after DCX instruction, since no flag gets affected with this instruction. So to check the zero flag ORA instruction affect the zero flag. The zero flag will be set if the contents of both D and E registers are zero.

The time delay introduced by the inner loop is:

$$T_{LOOP} = 64 \times 24 \times \text{Time of one T-state.}$$

If the system clock frequency is 3 MHz, then

$$T_{LOOP} = 64 \times 24 \times \frac{1}{3}\mu\text{sec}$$

$$= 512\ \mu\text{sec}$$

$$= 0.5\ \text{msec.}$$

Delay introduced for outside loop is:

$$T_{out} = 27x1x\frac{1}{3}\mu sec$$
$$= 9\ \mu sec$$

So the total time delay introduced by the above subroutine program is given by:

$$T_{Delay} = 0.5\ msec + 9\ \mu sec$$
$$\approx 0.5\ msec$$

Further, it is required that the voltage level of the output should be 2.5 volts, for which the digital input should be 40 H as 80 H gives 5 volts. So the main program is given as:

MAIN PROGRAM

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	80 H	; Initialize 8255-I to work all the ports as output ports.
	OUT	03 H	; Write the control word (80 H) in the control word register of 8255-I.
START	MVI A,	00 H	; Load 00 H to accumulator.
	OUT	00 H	; Send 00 H (00 V) to PA_0.
	PUSH PSW		; Save PSW
	CALL	DELAY	; Jump to delay subroutine to introduce a delay of 0.5msec.
	POP PSW		; Restore the PSW.
	MVI A,	40 H	; Store 40 H to the accumulator.
	OUT	00 H	; Outputs for 40 H.
	PUSH PSW		; Save PSW
	CALL	DELAY	; Jump to delay subroutine to introduce a delay of 0.5msec.
	POP PSW		; Restore the PSW.
	JMP	START	; Repeat the process for next cycle.

During the execution of this program the wave shape at output of the D/A converter can be checked. It will be a square wave of 1 KHz frequency.

Example 13.3. *Write a program to generate a triangular wave form using 8255 and DAC 0808 (ref. fig. 13.13).*

Solution. The program for the same is given below:

MAIN PROGRAM

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	80 H	; Initialize 8255-I to work all the ports as output ports.
	OUT	03 H	; Write the control word (80 H) in the control word register of 8255-I.
LOOP	MVI A,	FF H	; Load FF H to accumulator.
	OUT	00 H	; Send FF H (10 V) to PA_0.
	INR A		; Increment accumulator.
	OUT	00 H	; Output corresponding to this weight is available (i.e. increasing

```
                                        voltage is available at the output of
                                        the converter).
          CPI       FF H          ; Compare if output has become 10
                                      Volts.
          JZ        LOOP 1        ; If yes go to LOOP 1.
          JMP       LOOP          ; If no go to LOOP.
LOOP 1    DCR  A                  ; Decrement accumulator.
          OUT       00 H          ; Outputs for decrement data.
          JZ        LOOP          ; If accumulator content becomes 0,
                                      go to LOOP.
          JMP       LOOP          ; Else to LOOP.
```

During the execution of this program the wave shape at output of the D/A converter can be checked. It will be triangular wave.

13.5 MICROPROCESSOR COMPATIBLE D/A CONVERTER

Now a days microprocessor compatible D/A converters are also available, which are designed for the purpose of directly interfacing with the microprocessor without the use of external latch. Among such converters, AD558 is very simple, inexpensive and commonly used D/A converter to be connected directly to the microprocessor.

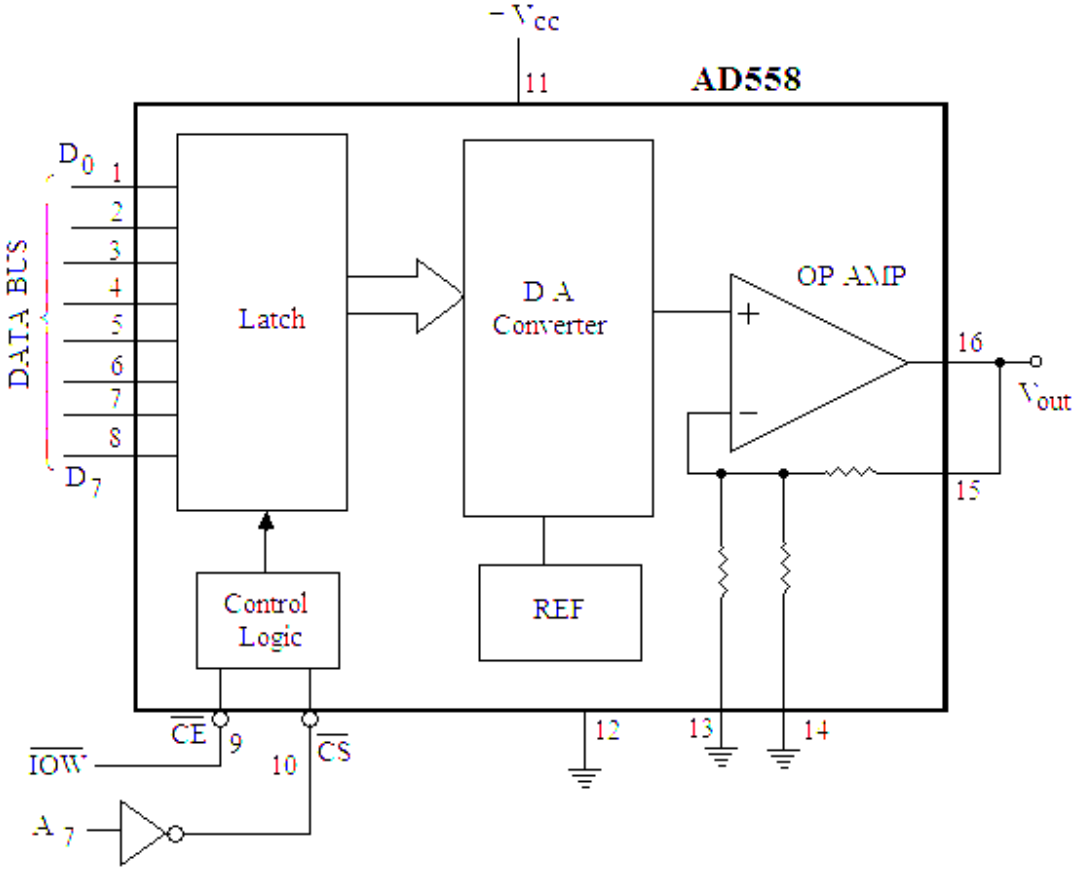


Fig. 13.14

Figure 13.14 shows the block diagram of this D/A converter, which includes internally a latch and output op amp. It can be operated with + 4.5 V to 16.5 V d.c. power

supply. Two pins $\overline{CS}$ (chip select) and $\overline{CE}$ (chip enable) are provided with the chip to interface with the microprocessor 8080 or 8085. For the interfacing with 8085 $\overline{CE}$ may be connected to $\overline{IOW}$ of the processor and $\overline{CS}$ may be connected to the A_7 bit of the address line. The other pins of the address lines may be assumed as 0, so that 80 H will be the port address of this converter. When both the signals $\overline{CS}$ and $\overline{CE}$ are at logic 0, the latch is transparent means the input is transferred to the D/A block of this converter. When either of $\overline{CS}$ or $\overline{CE}$ goes to logic 1, the input is latched in the register and held until both control signal go to logic 0.

The programs given in the above examples (solved) can be executed with this converter by using the proper port address.

13.6 ANALOG TO DIGITAL CONVERTER

Generally the information to be processed by the digital systems is in the analog form. So before applying such signals to the digital systems it is necessary to convert the signal into its equivalent digital form. The method with the help of which the analog signal is converted to digital form is known as analog to digital (A/D) converter. The A/D converter is more complex and difficult than the D/A converter. Followings are the different methods for A/D converter, which will be discussed in the next sections.

(i) Simultaneous A/D converter
(ii) Successive approximation D/A converter
(iii) Counter or Digital Ramp type A/ D converter
(iv) Single slope D/A converter
(v) Dual slope D/A converter

13.7 SIMULTANEOUS A/D CONVERTER

This is the fastest and simplest method of converting an analog signal to digital signal. It utilizes the parallel differential comparators; the input analog voltage is compared by these comparators with known voltages called as reference voltages. The comparators gives the low output (logic 0) when the input is less than the reference voltage and gives the high output (logic 1) when the input analog voltage exceeds the reference voltage. This method of conversion is also called as Flash or parallel type A/D converter.

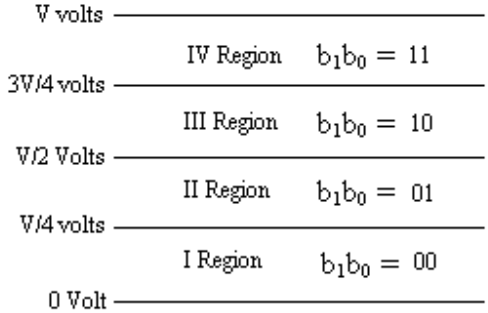


Fig. 13.15

For the conversion of analog voltage ranging between 0 to V volts into two bit digital output, three comparators (in general $2^n - 1$ comparators where n in the number of

bits) are required. The input analog voltage is converted to the 4 (in general 2^n) equal regions as shown in figure 13.15. If the analog voltage is lying in the first region, then the binary bits (b_1, b_0) are 00, similarly to second, third and forth regions the binary bits are 01, 10 and 11 respectively. The reference voltages to the three comparators C_0, C_1, C_2 should be V/4, V/2, 3V/4 respectively as shown in figure 13.16. The output of the three comparators should be connected to the logic gates to produce the desired binary output. The read gates and output registers are used to read the digital output.

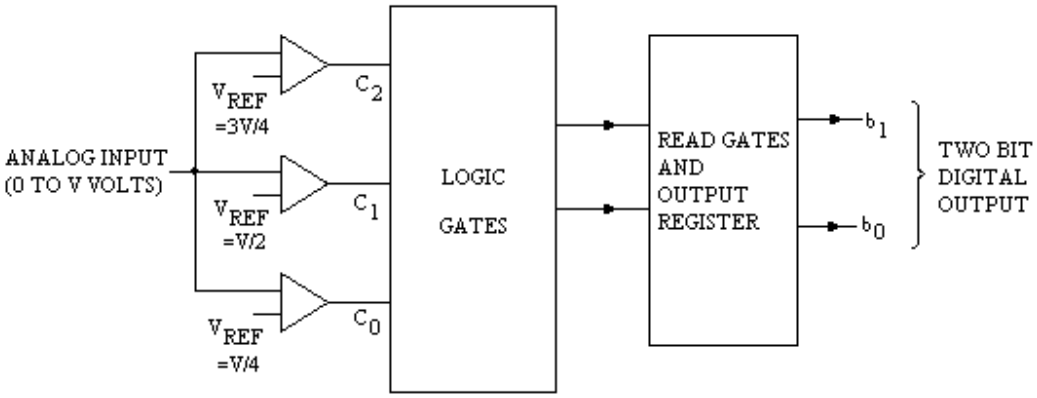


Fig. 13.16

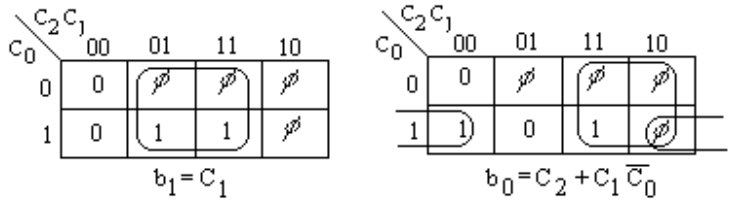
Referring to figures 13.15 and 13.16, if the input analog voltage exceeds the reference voltage to any comparator, the comparator gives high output (logic 1); if on the other hand if the input analog voltage is less than the reference voltage of the comparator, it gives low output (logic 0). In this way if all the comparators give low output, the analog input voltage must be between 0 and V/4 volts (I region) and digital binary output should be 00. If the C_0 is high and C_1 and C_2 are low, the input must be between V/4 and V/2 volts (II region) and digital binary output should be 01. If C_0 and C_1 are high and C_2 is low, the input must be between V/2 and 3V/4 volts (III region) and digital binary output should be 10. Finally if all the comparators give high outputs, the input must lies 3V/4 and V volts (IV region) and digital binary output should be 11. Table 13.2 summarizes outputs of the comparators.

Table 13.2

	Comparator output			Binary output	
Input voltage	C_2	C_1	C_0	b_1	b_0
0 to V/4	0	0	0	0	0
V/4 to V/2	0	0	1	0	1
V/2 to 3V/4	0	1	1	1	0
3V/4 to V	1	1	1	1	1

By drawing the K –maps (figure 13.17), the expressions for b_0 and b_1 are obtained as:

$$b_1 = C_1 \qquad \text{and} \qquad b_0 = C_1 \cdot \overline{C}_0$$



(a) **(b)**

Fig. 13.17

These expressions may be realized using the gates as shown in figure 13.18. The output may be reset by applying high signal to the reset line and to read the data a high signal is applied to the read line.

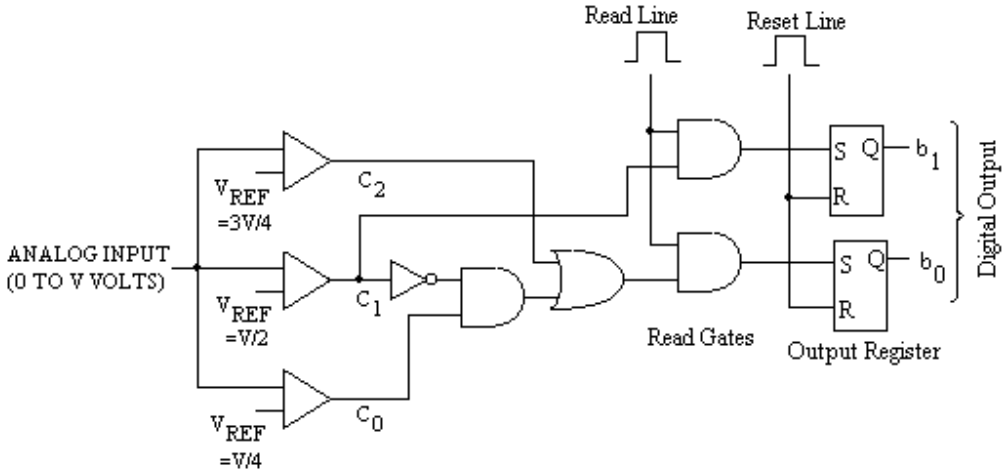


Fig. 13.18

For the conversion of analog input voltage (0 to V volts) into three bit binary output we proceed in the similar method as for the two bits output. For the three bits outputs the input voltages are divided into 8 (as $2^3 = 8$) equal regions and 7 (as $2^3-1 = 7$) comparators are to be used. So the logic circuit to be designed should have seven inputs (output of the seven comparators) and three outputs. The output of comparators and the corresponding binary output are shown in table 13.3.

Table 13.3

Input voltage	Comparator output							Binary output		
	C_6	C_5	C_4	C_3	C_2	C_1	C_0	b_2	b_1	b_0
0 to V/8	0	0	0	0	0	0	0	0	0	0
V/8 to V/4	0	0	0	0	0	0	1	0	0	1
V/4 to 3V/8	0	0	0	0	0	1	1	0	1	0
3V/8 to V/2	0	0	0	0	1	1	1	0	1	1
V/2 to 5V/8	0	0	0	1	1	1	1	1	0	0
5V/8 to 3V/4	0	0	1	1	1	1	1	1	0	1
3V/4 to 7V/8	0	1	1	1	1	1	1	1	1	0
7V/8 to V	1	1	1	1	1	1	1	1	1	1

The expressions of the output bits can easily be obtained by examining the table 13.3.

The bit b_2 gives the high output whenever the output of the comparator C_3 is high. So $b_2 = C_3$.

The bit b_1 is high whenever the output of comparator C_1 is high and output of C_3 is low, or whenever the output of comparator C_5 is high. So $b_1 = C_1 \cdot \overline{C}_3 + C_5$.

Similarly, the expression for bit b_0 can be obtained as:

$$b_0 = C_0 \cdot \overline{C}_1 + C_2 \cdot \overline{C}_3 + C_4 \cdot \overline{C}_5 + C_6$$

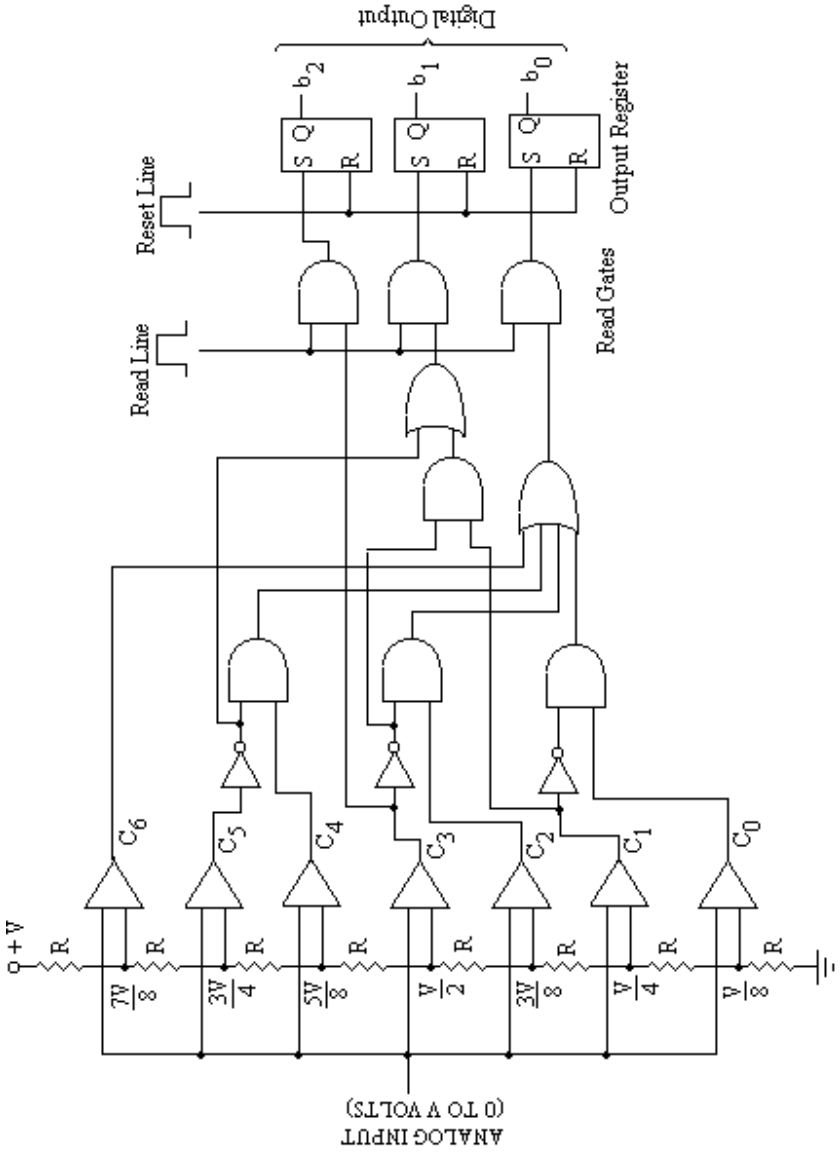


Fig. 13.19

These expressions may be realized using the gates as shown in figure 13.19. The output may be reset by applying high signal to the reset line and to read the data a high signal is applied to the read line.

The design of a simultaneous A/D converter is quite straight forward and relatively easy to understand. However, the design becomes complicated as the number

of bits is increased, since the number of comparators to be used increases drastically. This method has highest speed of conversion.

13.8 SUCCESSIVE APPROXIMATION A/D CONVERTER

Simultaneous A/D converter has the very fast conversion time but becomes unwieldy when the required digital bits are more. The successive approximation method is most useful and commonly used method. The block diagram four bit successive approximation A/D converter is shown in figure 13.20. It consists of a D/A converter, successive approximation register (SAR) and a comparator. The basic principle of this A/D converter is as follows:

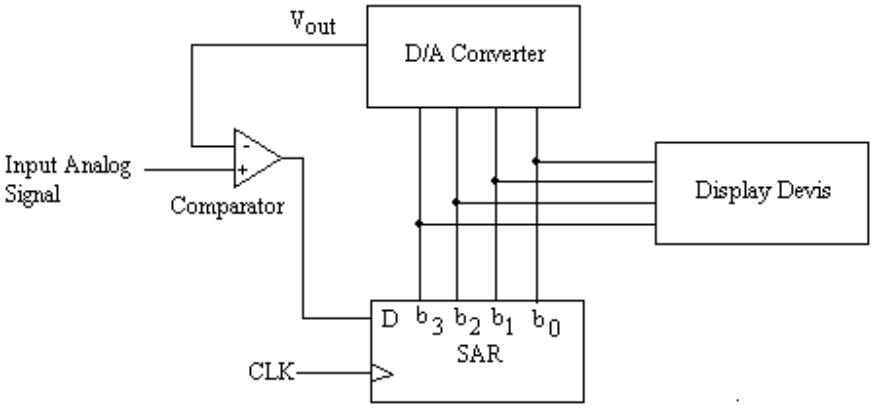


Fig. 13.20

In this type of converter, the bits of D/A converter are enabled one by one, starting with the most significant bit (MSB). The analog output of the D/A converter corresponding to the enabled bit is compared with the input analog voltage. The comparator gives the output low if the input analog voltage is less than the output of the D/A converter and it gives the high out if the input analog voltage is more than the output of the D/A converter. The low output of the comparator resets the corresponding bit of SAR, on the other hand if the comparator's output is high then that bit is retained in SAR. In this way the output of D/A converter are compared with the input voltage for all the bits starting with the most significant bit.

Thus the successive approximation method is the process of approximating the analog voltage bit by bit starting with MSB. This process is shown in figure 13.21.

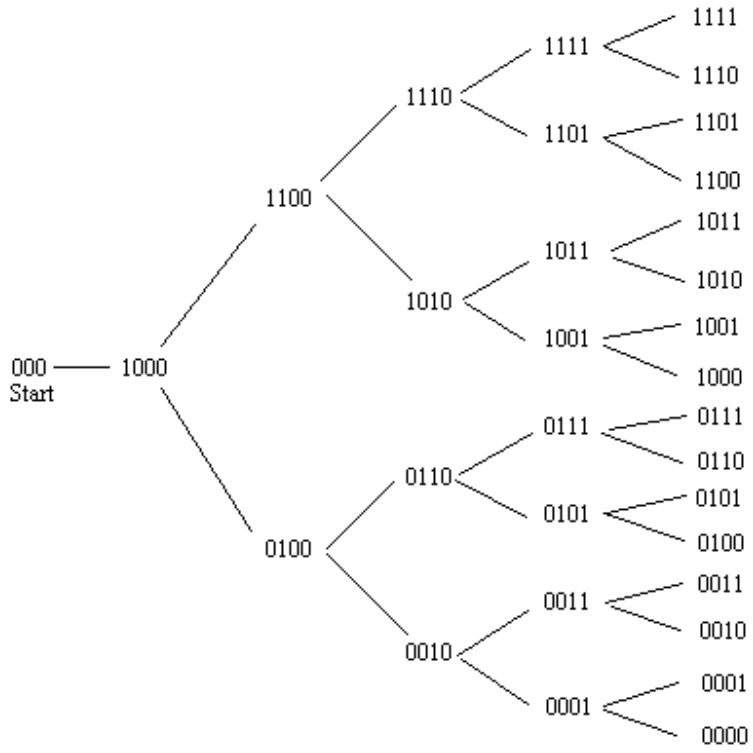


Fig. 13.21

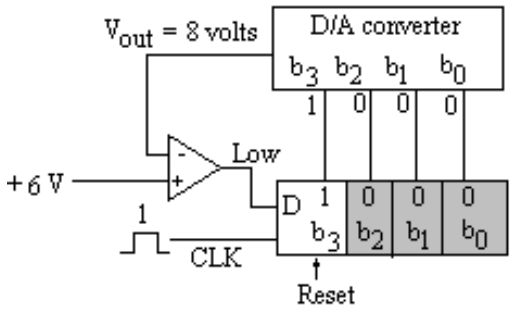
In order to understand the operation of this type of A/D converter, we will take a specific example of a four-bit conversion. Figure 13.22 (a through d) shows the step –by-step conversion of a given analog input voltage (say 6 volts). It is further assumed that D/A converter has the following output characteristics:

V_{out} = 8 volts for bit 3 (MSB or b_3)

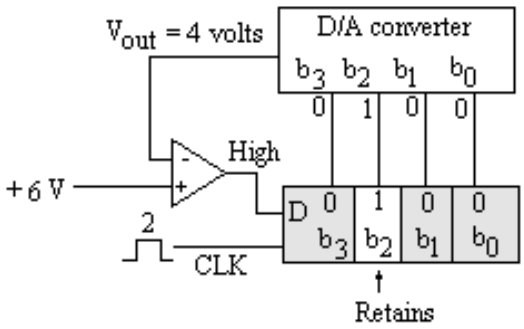
V_{out}= 4 volts for bit 2 (2^{nd} MSB or b_2)

V_{out}= 2 volts for bit 1 (3^{rd} MSB or b_1)

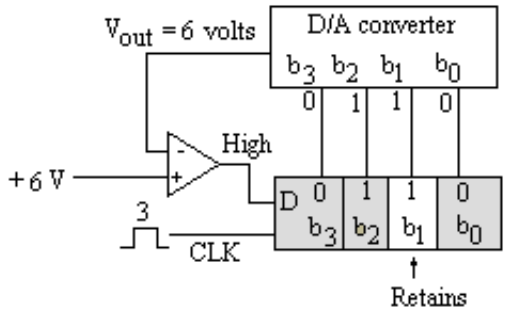
V_{out}=1 volt for bit 0 (LSB or b_0)



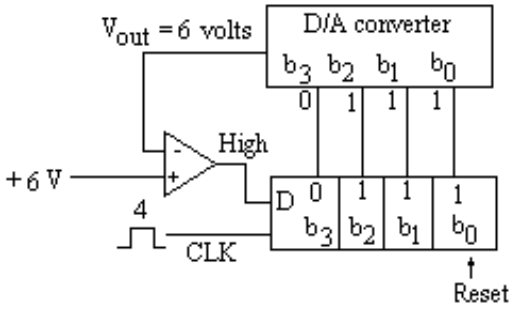
(a)



(b)



(c)



(d)

Fig. 13.22

It is clear from these figures that after completing the conversion cycle. The binary code 0110 is retained in SAR, which is binary value of the input voltage (6Volts). It is finally displayed on the display devices.

13.9 COUNTER OR DIGITAL RAMP TYPE A/D CONVERTER

Another method of converting the analog signal to digital one is the counter or digital ramp type A/D converter which utilizes a binary counter to count a continuous pulse of standard width and height from a clock. The standard clock pulses are passed through a gate which is open for some time to allow these pulses to go to the input of counter. Normally the gate is closed and as soon as the start signal is applied a stair case voltage is initiated. This voltage is increased linearly with the increase of the binary counts in the counter. The gate remains open for the time the linear stair case voltage

becomes equal to the input analog voltage. The counter records the number of clock pulses which is proportional to the input analog voltage.

Figure 13.23 shows the schematic diagram of this type of A/D converter. The analog signal, to be converted to its equivalent digital output, is applied to one input of an operational amplifier being used as a comparator. When a start of conversion pulse is applied to the control unit it resets the binary counter and opens the gate. The counter

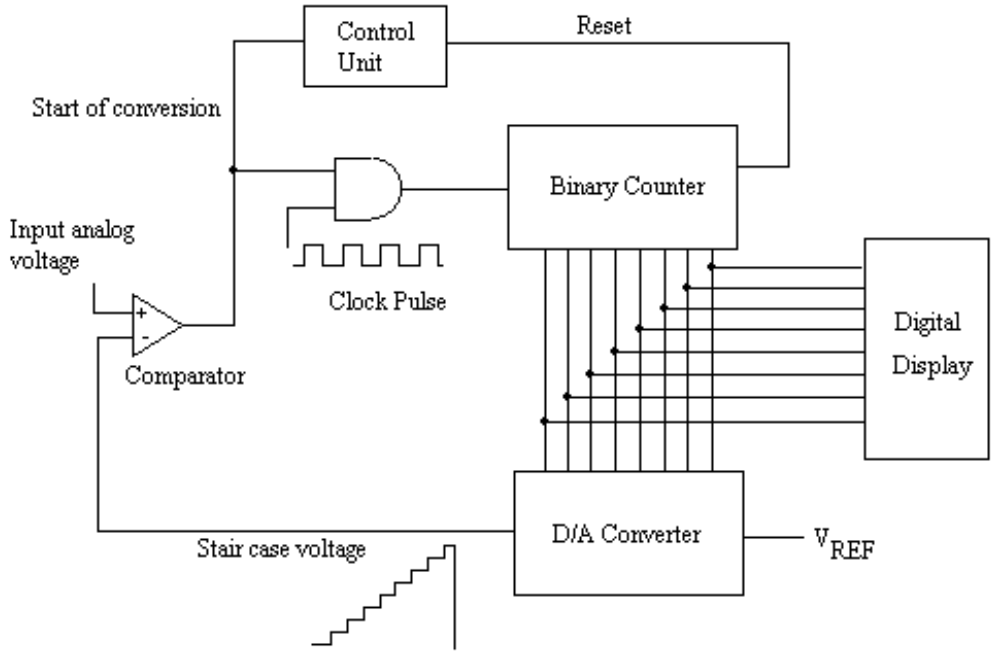


Fig. 13.23

starts counting the clock pulses which are of standard width and height. The output of the counter is fed to a D/A converter which produces an analog output (stair case voltage) in response to the digital signal (output of the counter) as its input. This analog output voltage is fed to the reference input of the comparator. So long as the input analog signal is greater than the stair case voltage the comparator provides the high output to the gate, the gate remains open and the clock pulses are allowed to reach to the input of the counter. These pulses are counted by the counter thus continuously increasing the digital output. The moment the analog output of D/A converter (stair case voltage) exceeds the input analog voltage, the comparator provides a low output disabling the gate and the counter stops counting. The binary number stored in the counter represents the digital output voltage corresponding to the input analog voltage. The digital output is displayed on the display devices.

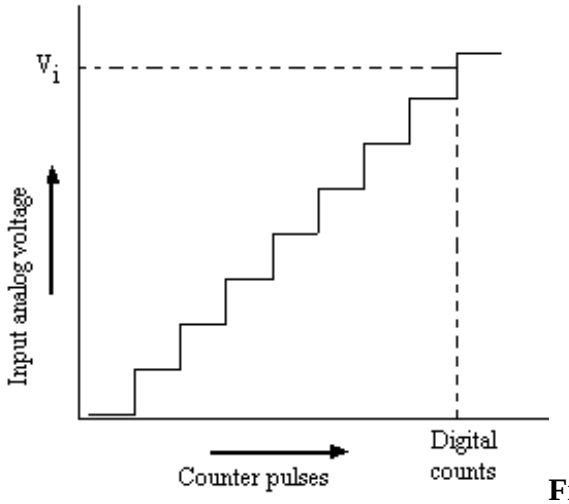


Fig. 13.24

For a steady input the digital output is as shown in figure 13.24. The output is represented by the number of clock pulses counted by the counter till the stair case voltage becomes equal to the input voltage. This method of conversion is slow; as for maximum input, the counter has to count from zero to maximum number of states for the comparison. For each conversion cycle the counter is to be reset and counting starts from beginning. The time of conversion is not important in d.c. or slow varying signals as the output waveform gives a good representation from which the input waveform can be constructed as shown in figure 13.25. But if the conversion time and the signal transient time are comparable the reconstructed digital output will not be correct. In this case it is necessary to reduce the conversion time by using faster D/A converter.

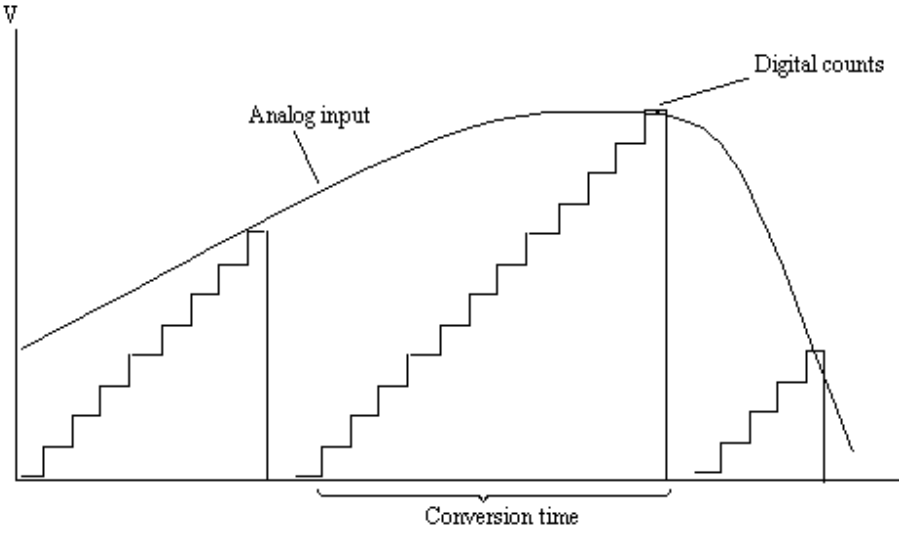


Fig. 13.25

A modification to this converter is possible if the resetting of the counter is avoided each time. For this purpose an up/down counter may be used in place of up

counter. The circuit shown in figure 13.26 illustrates this modification in which an

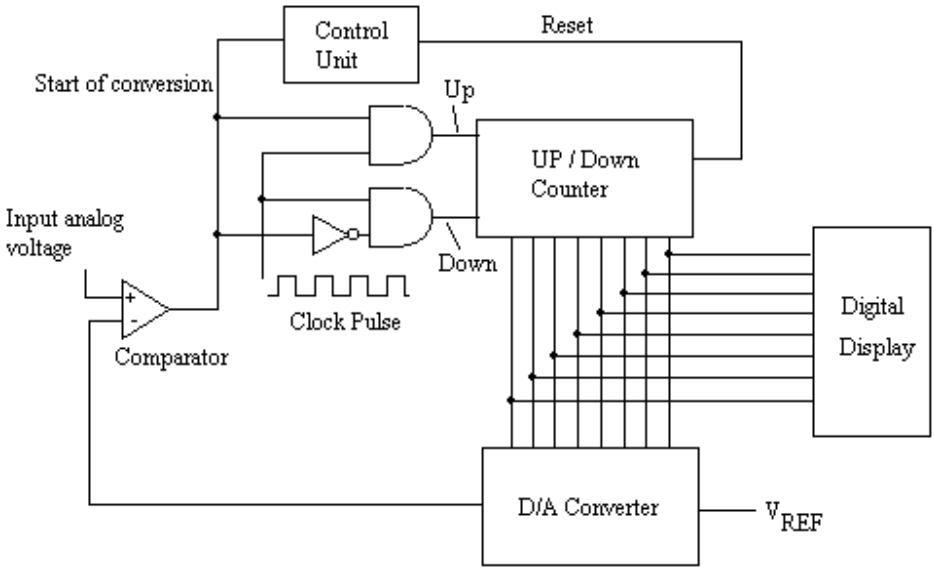


Fig. 13.26

up/down binary counter is used and the converter proceeds without resetting. The circuit is almost the same as the counter or digital ramp type A/D converter. The up/Down counter is operated by up or down signals from the control unit. The digital to analog converter output controls the output of the comparator. Till the D/A converter output is less than the analog input voltage, the up signal is enabled and the counter counts in forward direction. When the analog input falls, the down signal is enabled and the counter starts reverse counting giving an output corresponding to new analog input as shown in figure 13.27.

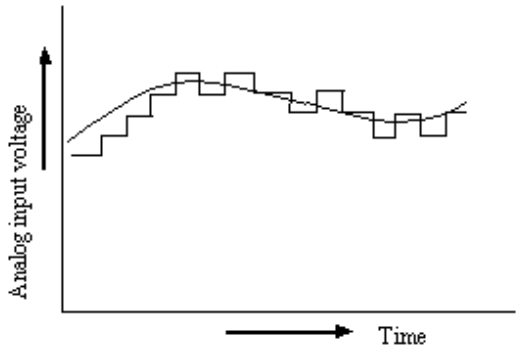


Fig. 13.27

13.10 SINGLE SLOPE A/D CONVERTER

This type of method is similar to counter or digital ramp type A/D converter. In this type of A/D converter also, a gate whose period is proportional to the amplitude of the analog sample is generated. For the generation of gate, the input analog voltage is compared with the output of an integrator. The output of integrator is a ramp voltage of constant slope. The standard clock pulses are passed through the gate and are counted by the counter. The gate remains open for the time proportional to the input analog signal. The recorded number of pulses is, therefore, the required digital output of the analog signal.

The schematic block diagram of such an analog to digital converter is shown in figure 13.28. Initially a reset pulse is applied which clears the counter and resets the integrator. The integrator produces a linearly rising ramp voltage, whose slope will depend on the values of the resistance R and capacitor C. The input analog voltage is compared by a comparator with the ramp voltage. As long as the integrator output is smaller than the input analog voltage, the comparator output is high. This high output enables the AND gate. The standard clock pulses are, therefore, allowed to pass through the gate which will be counted by the counter. When the ramp voltage becomes greater than the input analog voltage, the comparator changes the state thereby disabling the AND gate. The counter stops counting. It can easily be seen that the gate duration is linearly related to the magnitude of the input analog signal. Hence the count accumulated in the counter is a digital representation of the input analog voltage.

It may be mentioned here that the precision in the proportionality between the gate duration and the magnitude of the input analog signal depends upon the linearity of the ramp voltage obtained at the output of the operational amplifier. So the overall accuracy will depend upon the stability of reference source, the off-set of the operational amplifier, the frequency stability of the clock as well as the values of resistance R and capacitance C.

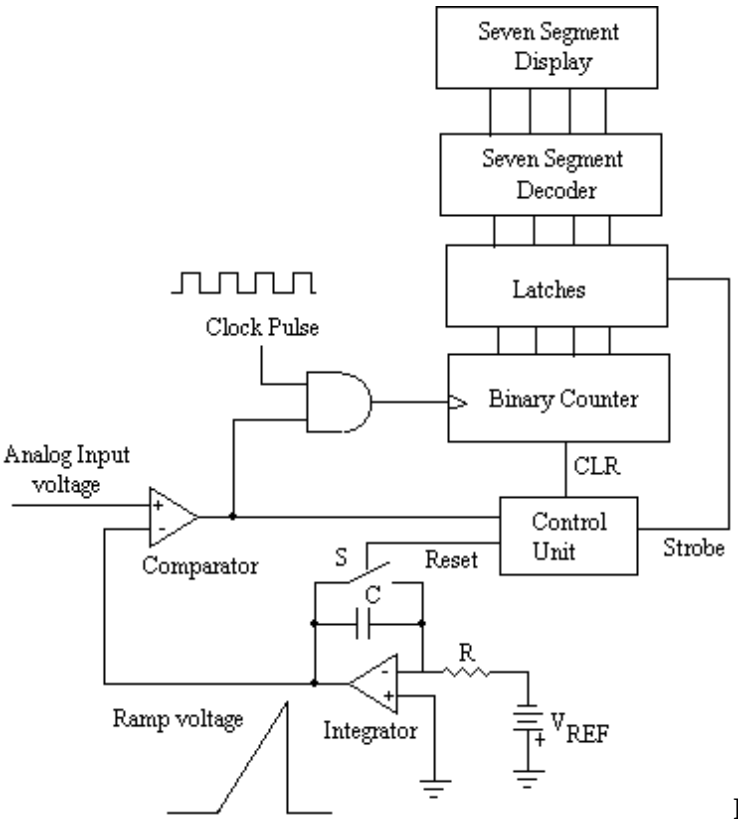


Fig. 13.28

13.11 DUAL SLOPE A/D CONVERTER

In single slope A/D converter, the accuracy of the converter depends on the linearity of the ramp voltage generated by the integrator. The linearity of ramp voltage, however, depends on the accuracy of the values of Resistance R and capacitance C of the integrator, whose values may vary with time and temperature. The dual slope analog to digital converter utilizes two different ramps, one for fixed time and other for fixed slope. It is very popular and widely used D/A converter because it has the slowest conversion time and relatively low cost. This method offers good accuracy, good linearity, and very good noise rejection characteristics.

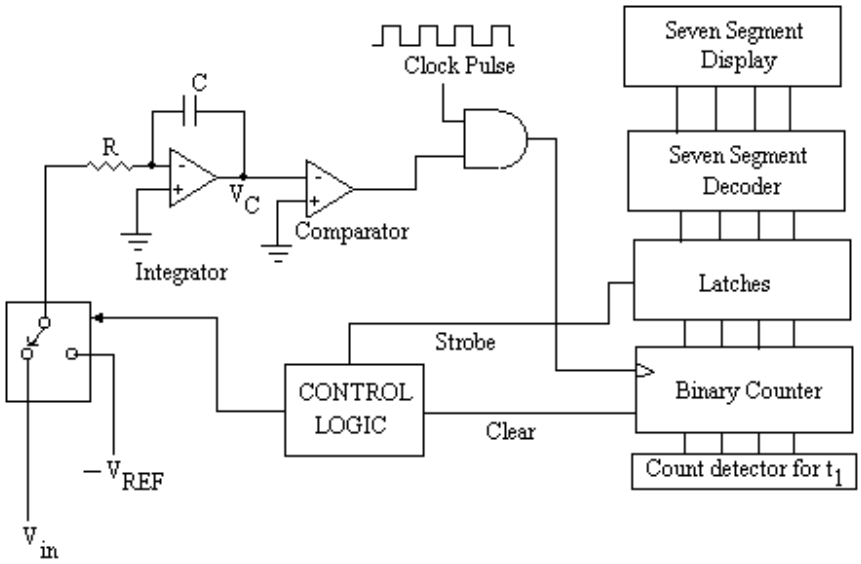


Fig. 13.29

The logic diagram of the dual slope A/D converter is given in figure 13.29. This converter is similar to that of the single slope A/D converter. In this converter, the integrator forms two different ramps, one for fixed time and other for fixed slope. The capacitor of the integrator is first charged with constant current obtained from input analog voltage for fixed time then the capacitor is discharged for fixed slope through other constant current obtained from a reference voltage source. The basic operation of this converter can be understood as follows:

This converter consists of standard clock pulses applied to the gate. The gate allows the pulses to the input of the counter which counts these pulses. Initially all the counters are reset to 0's and ramp too is reset to zero. Now the control logic allows switch S to connect the input analog voltage V_{in} to the integrator circuit. A constant current equal to $\frac{V_{in}}{R}$ flows through the capacitor C as the inverting input of the operational amplifier of the integrator is at virtual ground. The capacitor C will charge linearly with this constant current. This results a negative going ramp at the output of the integrator. The comparator's output will be positive which allows the clock pulse to pass through the AND gate to the input of the counter. This ramp is allowed for fixed time say t_1. The actual time t_1 is determined by the count detector. The voltage V_C at the output of the integrator is given by:

$$V_C = -\frac{1}{R.C}\int_0^{t_1} V_{in}.dt$$

$$= -\frac{1}{R.C}.V_{in}.t_1 \qquad \ldots (13.6)$$

The counter when reaches the fixed count at t_1, the control logic generate a pulse to clear the counter to zero and the switch S connects the integrator input to a negative reference voltage ($-V_{REF}$). The capacitor C of the integrator starts discharging linearly due to the constant current from $-V_{REF.}$ The integrator thus produces a positive going ramp beginning at $-V_C$ and increases steadily till it reaches to 0 volt as shown in figure 11.29. At this time the counter is counting. The conversion cycle ends when $V_C = 0$ volt; the comparator produces the low state, which disables the gate and counter stops counting.

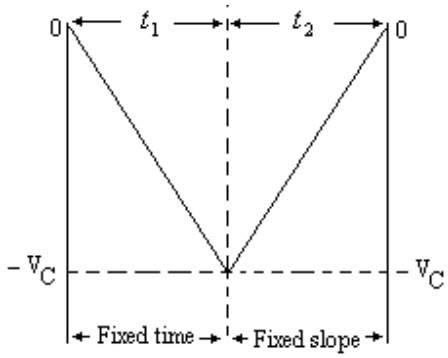


Fig. 13.30

Let t_2 is the time when the output of integrator becomes zero, so the output of the integrator is given by:

$$V_C = \frac{V_{REF}}{R.C}.t_2 \qquad \ldots (13.7)$$

Since the integrator's output beginning at 0 volt and integrates down to $-V_C$ and then integrate back to 0 volt, so the equations (13.6) and (13.7) may be equated as:

$$\frac{V_{in}}{R.C}.t_1 = \frac{V_{REF}}{R.C}.t_2$$

or
$$V_{in} = V_{REF}.\frac{t_2}{t_1} \qquad \ldots (13.8)$$

In this equation V_{REF} and time t_1 are constants, so

$$V_{in} \propto t_2 \qquad \ldots (11.9)$$

This equation is independent of the values of resistance R and capacitance C. Further at the end of conversion cycle, the counts measured by the counter are proportional to the input analog signal are latched to display on the display devices.

13.12 ADC 0808/0809

The ADC 0808/0809 is an 8-bit A/D converter with 8-channel multiplexer which can be interfaced with the microprocessor. It is the most popular, inexpensive and widely

used A/D converter. It is a monolithic CMOS device which uses the method of successive approximation A/D converter. It does not require external zero and full scale adjustment. The device operates with + 5 V d.c. supply. The conversion time is 100 μsec at clock frequency of 640 KHz. The 8-channel multiplexer can directly access any of the 8 single ended analog signals.

The ADC 0808 / 0809 offers following features:

- High speed
- High accuracy
- Minimal temperature dependence
- Excellent long term accuracy and repeatability
- Consume minimal power

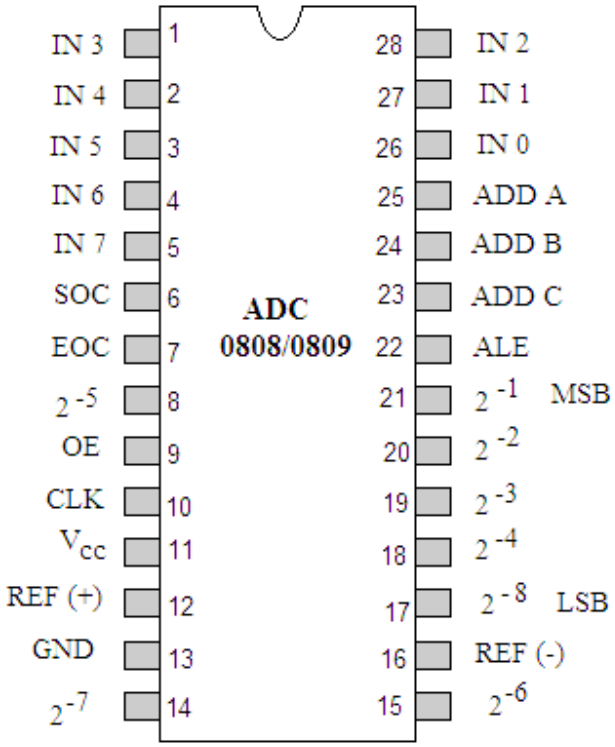


Fig. 13.31

The pin diagram and schematic block diagrams of this IC ADC 0808 / 0809 are shown in figures 13.31 and 13.32 respectively. The pin description of this IC is given as:

Pins 26-28 and 1-5 – are 8 input channels IN0 – IN7 (multiplexed), which are selected by the three (address) lines ADD A, ADD B and ADD C.

Pin 6 SOC – Start of conversion pin. A high to this pin starts the conversion.

Pin 7 EOC – End of conversion. This an output terminal and gives high signal when the conversion ends.

Pins 21-18, 8, 15-14 & 17 8-bit output (latch) pins to be connected to the port of 8255.

Pin 9 OE – Output enable pin which is connected to the positive supply.

Pin 10 CLK – Clock terminal. It is to be connected either to the external clock of frequency between 10 to 1280 KHz or the clock out frequency of the 8085A after dividing it by a factor of four may be connected to this pin.

Pin 11 V_{CC} – Supply pin (+5 V).

Pin 12 Ref (+) – Reference pin which is to be connected to the + supply.

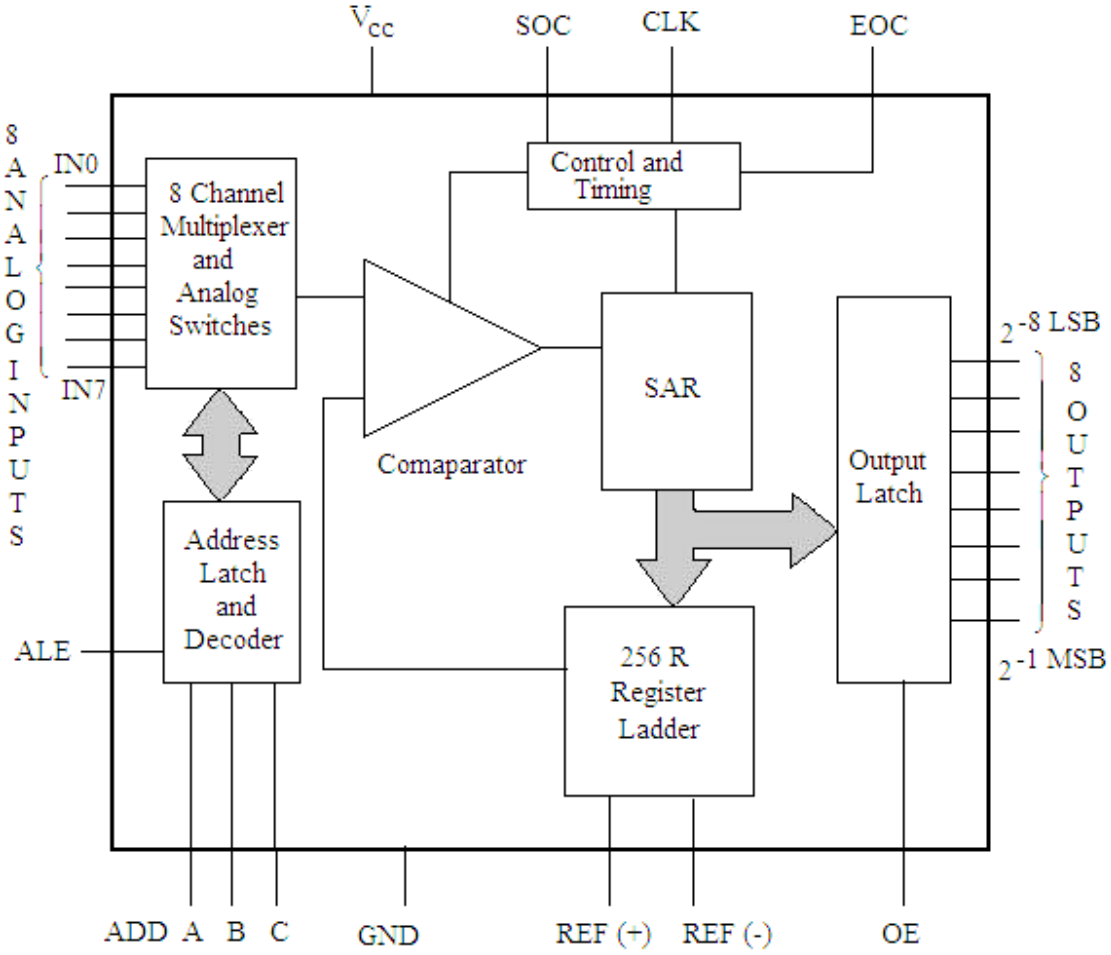


Fig. 13.32

Pin 13 GND – Ground terminal.

Pin 16 Ref (-) – Reference pin to be connected to the ground.

Pins 25-23 – Address pins (ADD A, ADD B and ADD C). These pins are used to select the channels as given below (Table 13.4):

Table 13.4

Channels Selected	Address lines		
	C	B	A
IN0	0	0	0

IN1	0	0	1
IN2	0	1	0
IN3	0	1	1
IN4	1	0	0
IN5	1	0	1
IN6	1	1	0
IN7	1	1	1

The timing diagram of this IC is shown in figure 13.33.

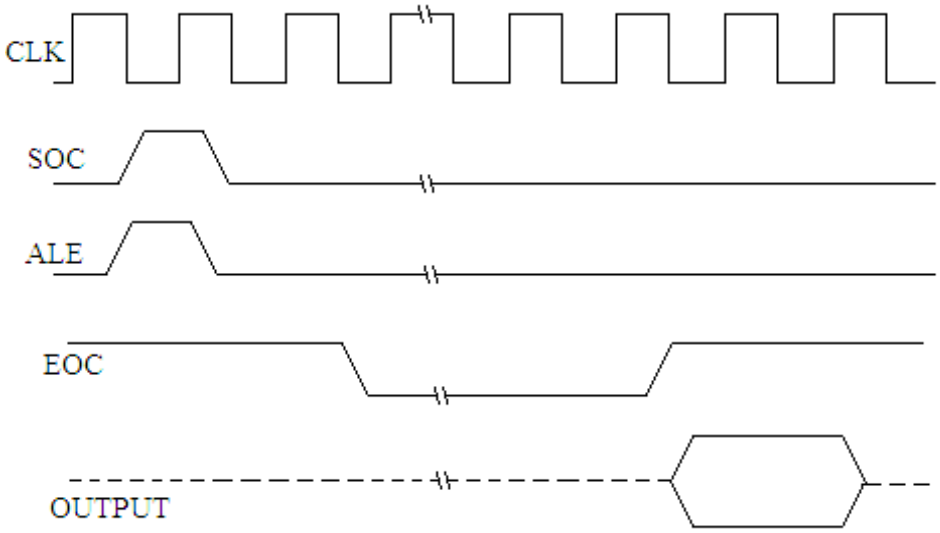


Fig. 13.33

The interfacing connection of ADC 0808 / 0809 with the microprocessor 8085A is shown in figure 13.34. In this circuit an analog voltage, whose equivalent digital value

is to be obtained, is applied to IN0. The output terminals of the ADC are connected to

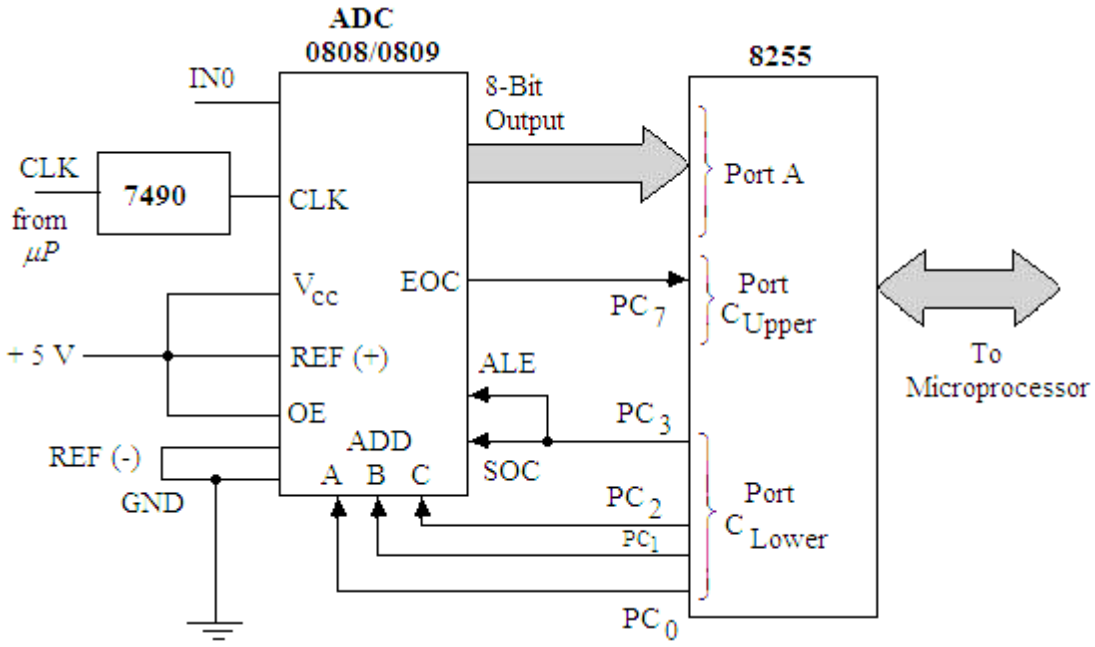


Fig. 13.34

Port A of 8255 PPI. This port is to be used as the input port. The PC_7 terminal of the Port C_{Upper} is connected to the End of Conversion terminal of the ADC. This port is also used as the input port. The port C_{Lower} is used as the output port. The three terminals PC_0 to PC_2 are connected to the three address pins of the ADC 0808 / 0809 for the selection of input channel. The PC3 terminal is connected to the SOC (Start of Conversion) and the ALE terminals of the converter. The OE and REF (+) terminals are connected to the + 5V supply. The assembly language program is given below:

In the above circuit diagram, the channel used is IN0, so for its selection we have:

C	B	A	
PC_2	PC_1	PC_0	
0	0	0	= 00 H

To enable the start of conversion first PC3 is made high and then low without changing the selection of channel, we proceed as follows:

PC_3	PC_2	PC_1	PC_0	
1	0	0	0	=08 H
PC_3	PC_2	PC_1	PC_0	
0	0	0	0	=00 H

PROGRAM

Label	**Mnemonics**	**Operand**	**Comments**
	MVI A,	98 H	; Initialize 8255-I to work Port A and Port C_{Upper} as input ports, and port C_{Lower} as output port.

```
        OUT     03 H    ; Write the control word in the
                          control word register of 8255-I.
        MVI  A, 00 H    ; Load accumulator with 00H.
        OUT     02 H    ; 00H is sent to Port C_Lower to select
                          IN0 channel.
        MVI  A, 08 H    ; Load 08 H to accumulator.
        OUT     02 H    ; ALE and SOC are enabled (high).
        MVI  A, 00 H    ; Load 00 H to accumulator.
        OUT     02 H    ; ALE and SOC will be low. A
                          pulse is applied from high to low
                          for the conversion through PC3
                          without affecting the channel
                          selected.
READ    IN      02 H    ; Read End of conversion (PC7).
        RAL             ; Rotate left to check if PC7 is one.
        JNC     READ    ; If not 1 READ again.
        IN      00 H    ; Read digital output of the A/D
                          converter.
        STA     2100 H  ; Store the result in 2100 H memory
                          location.
        HLT             ; Stop processing.
```

After the execution of this program, the values stored in the memory location 2100 may be checked.

Analog Voltage	Digital output
0 V	00 H
1 V	34 H
1.4 V	47 H
2 V	6D H
2.8 H	90 H
3 V	94 H
4 V	CE H
5 V	FF H

13.13 A/D CONVERTER USING D/A CONVERTER AND SOFTWARE

The A/D converter can also be realized using the D/A converter and the software. For this purpose the software is used to implement successive approximation A/D converter. Figure 13.35 shows the interfacing connection for this. The input analog voltage is applied to a comparator. This voltage is compared by the output of the D/A converter. The output of the comparator is sensed by the microprocessor through the PC_7 of the 8255. A digital input is applied to the D/A converter, through port A of 8255.

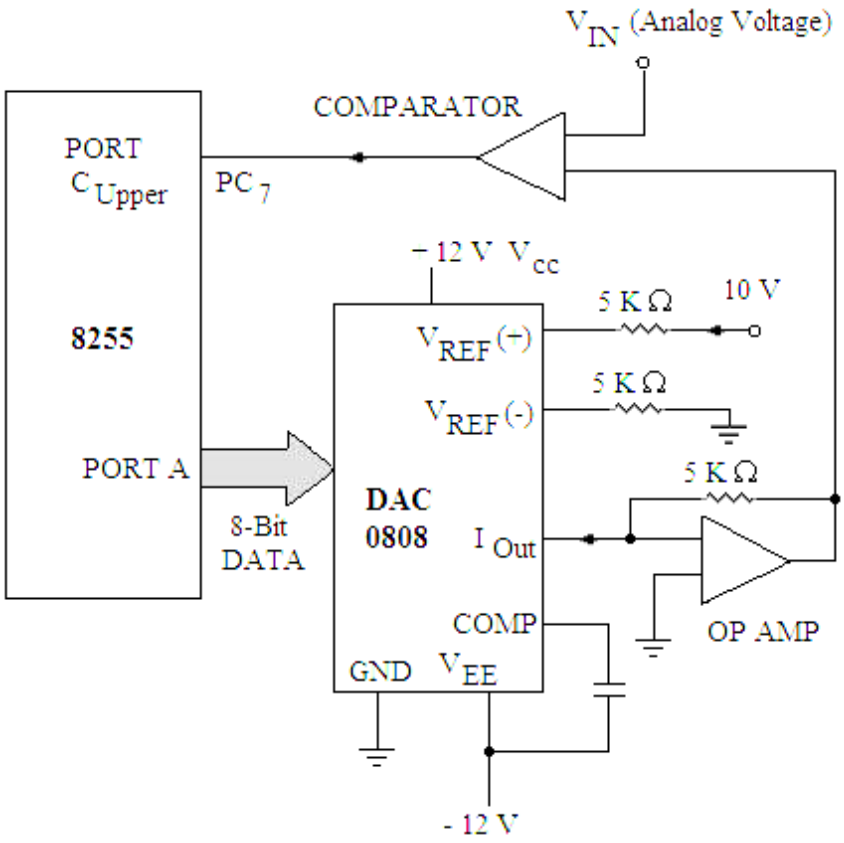


Fig. 13.35

The microprocessor first applies 00 H as the digital input to the DAC 0808 and checks through PC_7 for high signal. If a high pulse does not appear at the output terminal of the comparator, the microprocessor increments the digital input by 1 and the process is repeated till a high signal appears at the output of the comparator. As a high signal appears the microprocessor stops successive approximation and the corresponding digital input value is stored in the memory location. This digital value is equivalent to the analog signal applied at V_{IN} terminal of the comparator. The software for the same is given as:

PROGRAM

Label	**Mnemonics**	**operand**	**Comments**
	LXI SP,	XXXX H	; Initialize stack pointer.
	MVI A,	88 H	; Initialize 8255-I to work Port A, Port B and port C_{Lower} as output ports, and Port C_{Upper} as input ports.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	MVI A,	00 H	; Load accumulator with digital input to send it to D/A converter.
LOOP	OUT	00 H	; The digital input is sent to D/A converter through Port A.
	PUSH PSW		; Save PSW

```
        CALL    DELAY   ; Call delay subroutine program.
        IN      02 H    ; Check the output of the comparator through PC7, Read PC7 bit.
        RAL             ; Rotate left to check PC7, if it is high.
        JC      NXT     ; If is high go to NXT.
        POP  PSW        ; Else get the digital value from the stack.
        INR  A          ; Increment digital value.
        JMP     LOOP    ; Jump to LOOP.
NXT     POP  PSW        ; Get the digital value from the stack.
        STA     2100 H  ; Store the result in 2100 H memory location.
        HLT             ; Stop processing.
```

SUBROUTINE PROGRAM:

```
Label     Mnemonics   Operand   Comments
DELAY     LXI D,      0002 H  ; Loads DE register pair with a 16-bit number.
LOOP1     DCX  D              ; Decrements DE register pair by one.
          MOV  A,     E       ; Moves the contents of E register to accumulator.
          ORA  D              ; ORing of the contents of D and E registers are performed to set the zero flag.
          JNZ       LOOP1     ; If result ≠ 0 jump to loop1
          RET                 ; Go back to main program.
```

After the execution of this program, the values stored in the memory location may be checked for the input applied voltage.

Analog Input	Digital output
0 V	00 H
2.5 V	40 H
5 V	80 H
10 V	FF H

PROBLEMS

1. Discuss the resistive divider D/A converter. Find the general expression for the output voltage of a resistive divider network.
2. Using the resistive divider network draw the circuit of a 6 bit D/A converter and explain its operation. What are the drawbacks of this D/A converter?
3. Show that the outputs of a binary weighted resistor network are directly proportional to the binary inputs.
4. Draw the schematic diagram of a resistive divider D/A converter. Explain its operation. Mention the drawbacks of this converter.

5. A 5 bit resistive divider network has 0 volts full scale output, find the output voltage for a binary input 10101. (Ans. 6.774 V)
6. A 6 bit resistive divider D/A converter has resistance of 100 KΩ in MSB branch. The reference voltage is 15 V. The resistance in the feed back path of the operational amplifier is 39 KΩ. What is the output voltage for the binary input 101101? (Ans. – 8.22V)
7. For a 6 bit resistive divider network, the reference voltage is 10 V, find the following:
   (i) Full Scale output voltage.
   (ii) The analog output voltage for a digital input of 010011.
   (iii) The output voltage change due to least significant bit.
   (Ans.:10V, 3.02 V, 0.16 V)
8. Draw the schematic diagram of a binary ladder D/A converter. Explain its operation. Mention its merits and demerits.
9. Find the expressions for the output due to MSB and second MSB of a 4-bit binary ladder network.
10. Discuss the binary ladder D/A converter. Find the general expression for the output voltage of a binary ladder network.
11. What are the performance criteria for the D/A converter? Discuss their importance while selecting a D/A converter.
12. For a 6-bit binary ladder D/A converter the input levels are 0 = 0 V and 1 = 10 V, find
    (i) The output voltages caused by each bit.
    (ii) The output voltage corresponding to an input of 101101.
    (iii) The full scale output voltage of the ladder.
    (Ans. (i) -5V, -2.5 V, -1.25 V, -0.625 V, -0.3125 V, -0.15625 V (ii) – 7.03125 V (iii) – 9.84 V)
13. For a 5-bit binary ladder D/A converter the input levels are 0 = 0 V and 1 = 10 V, find the output voltage corresponding to binary input of (i) 10111 (ii) 01101.
    (Ans. – 7.1875 V, – 4.0625 V)
14. What is the step size (or resolution in volts) of a 10 bit D/A converter, if the full scale output is +10 volts? Find the percentage resolution also.
    (Ans. 9.78 mV, 0.0978%)
15. How many bits are required at the input of a D/A converter to achieve a resolution of 15mV, if the full scale output is 15 volts?
    (Ans. 10 bits)
16. Give the details of D/A converter IC 0808. Using this IC draw a circuit diagram to get the analog output voltage corresponding to 8 bit digital input.
17. Describe how the interfacing of the D/A converter is done with the microprocessor 8085A. Give its circuit diagram.
18. Discuss the simultaneous A/D converter to convert 0 to V volts analog voltage to 3 bit digital output. Draw the logic diagram also. What are the disadvantages of this type of A/D converter?
19. Give the details of the microprocessor compatible D/A converter IC AD558.
20. Draw a logic diagram to convert 0 to V volt analog voltage to its equivalent 2 bit digital output using simultaneous A/D converter.

21. Describe the successive approximation method for A/D conversion.
22. Draw a schematic diagram of counter or digital ramp type A/D converter. Explain its operation.
23. Describe the modified counter or digital ramp type A/D converter with neat diagram.
24. Draw a schematic diagram of a D/A converter. Explain its operation.
25. Describe single slope A/D converter with it logic diagram. Mention its merits and demerits.
26. Describe Dual slope A/D converter with it logic diagram.
27. Give the details of A/D converter IC 0801.
28. The input bits of the D/A converter is connected to the output pins of Port A of 8255, which is already connected with the microprocessor (ref. fig.13.13). Write a program to generate stair case voltage with ten steps. It should have the constant pulse duration.
29. Write a program to generate square wave of 2 KHz frequency with a peak voltage of 2.5 volts. Use D/A converter (ref. fig. 13.13) for this purpose. The square wave should be available at the output of the converter.
30. Give the details of A/D converter IC ADC0808/0809. Using this IC draw a circuit diagram to get the digital value of the input analog voltage.
31. Describe how the interfacing of the A/D converter is done with the microprocessor 8085A. Give its circuit diagram.
32. Write a program to get the digital output of an analog signal applied to ADC 0808/0809 interfaced with the 8085A. The input analog signal is applied to the channel 3 of this IC.
33. Explain how the A/D converter can be designed using D/A converter and the software. For this purpose the software is used to implement successive approximation A/D converter.

14

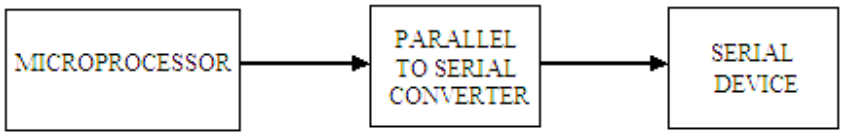
Serial Communication and Programmable Communication Interface

In continuation to the series of interfacing devices, the discussion will now be made on serial data communication and programmable communication interface, since the serial data transmission is always preferred for long distance transmission. Though the serial data transmission can be made through the SID (serial input data) and SOD (serial out data) lines of 8085A microprocessor, but it requires suitable software for the data transfer. Moreover, most of the microprocessors do not have the provision of SID and SOD lines. Apart from this, it can not offer high baud rate. Hence this chapter will deal the detailed discussion on serial communication and most powerful programmable communication interface IC 8251.

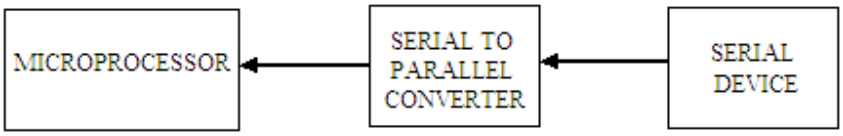
14.1 SERIAL DATA COMMUNICATION

There are two types of data communication, Parallel and Serial data communications. In parallel data communication each bit of a word is to be transmitted on a separate line along with a common ground line. In the 8085A microprocessor the data to be transmitted is of 8 bits. This parallel data transmission is not recommended for long distance communication as large number of data lines is needed. In serial data transmission, data is transmitted bit by bit either from the microprocessor to I/O devices or vice-versa. There are certain input/output devices in the microprocessor based systems, which only accept the serial data. The common I/O devices for such a data communication are CRT terminals, printers, cassette recorders etc. The microprocessors thus have the provision for transferring the data in the serial fashion. In such I/O devices, a parallel to serial conversion is required for the data transfer from the microprocessor to the peripheral device (figure 14.1a). Similarly, a serial to parallel conversion is required in transmitting the data from the peripheral device to the microprocessor (figure 14.1b).

As already discussed, the 8085A microprocessor has the provision of SID and SOD lines for the serial data transfer between the microprocessor and I/O devices. But it requires the software. Moreover, most of the microprocessors do not have the provision of these SID and SOD lines. Apart from this, it can not offer high baud rate. So, serial data communication is generally used for the long distance communication with high baud rate. Further, it requires less number of lines for the data transmission.



(a)



(b)

Fig. 14.1

The serial communication occurs either in the following two formats:

- Synchronous
- Asynchronous

In synchronous format, a receiver and a transmitter are synchronized. In this format the data is divided into fairly large blocks, typically 1000s of bytes. A block of characters is transmitted along with the synchronization information. The synchronizing signals are added once before each block. The synchronization characters mark the beginning of the transmission. This format is generally used for high baud rate (more than 20 K bits/second). The baud rate is defined as the number of bits transmitted or received per second. Figure 14.2 illustrates this format.

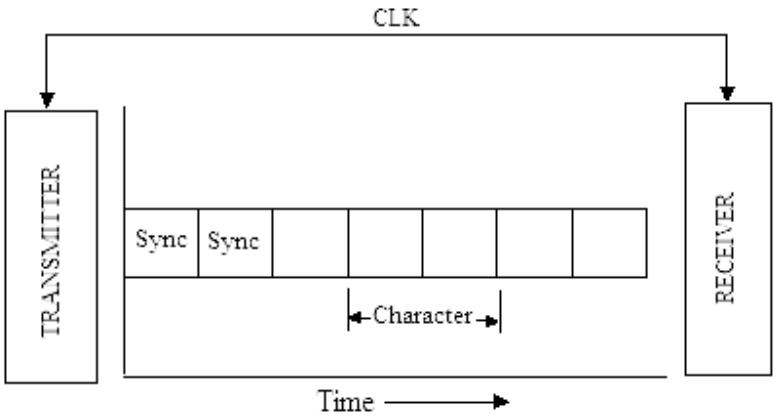


Fig. 14.2

In asynchronous format, timing signals are added to each character of data. A START bit is added at the beginning of each character. A STOP bit is added at the end of the character data. A low signal is used for the START bit and a high signal is used for STOP bit. When no data is being transmitted, a receiver stays high, called **MARK**. The process of adding start and stop bits to a character is known as **FRAMING**. The asynchronous format is generally used for low speed transmission. This format is illustrated in figure 14.3.

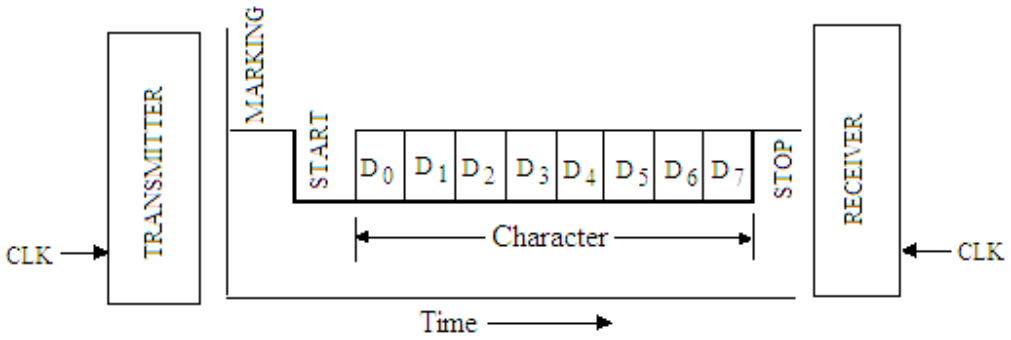


Fig. 14.3

The serial mode of data transfer can be divided into three groups namely:

Simplex Method

Duplex Method

Half Duplex Method

In simplex method data transfer takes place in unidirectional i.e. either from the system to the I/O devices or from I/O devices to the system, of course through the serial link. A typical example is the transmission from a system to a printer.

In the duplex method data transfer takes place in both the directions. However, if the transmission goes one way at a time, it is called half duplex; if it goes both ways simultaneously, it is called full duplex. The walkie talkie is an example of half duplex and the communication between the computers is an example of full duplex.

14.2 MODEM

One of the most common applications of the serial transmission would be data transmission between large computers. For such serial digital transmission the existing telephone lines are used. The telephone lines are designed to carry analog signals in the range of 40 Hz to 4 KHz. These lines will not readily support digital signals. Therefore, the digital data is to be converted into audio frequency format. A device that can convert the digital value in the form of the audio signal has to be used because the basic binary bits can not be placed directly on these lines. Such a device is known as MODEM (**MOD**ULATOR- **DEM**ODULATOR). The modem is be used at both the ends; at the

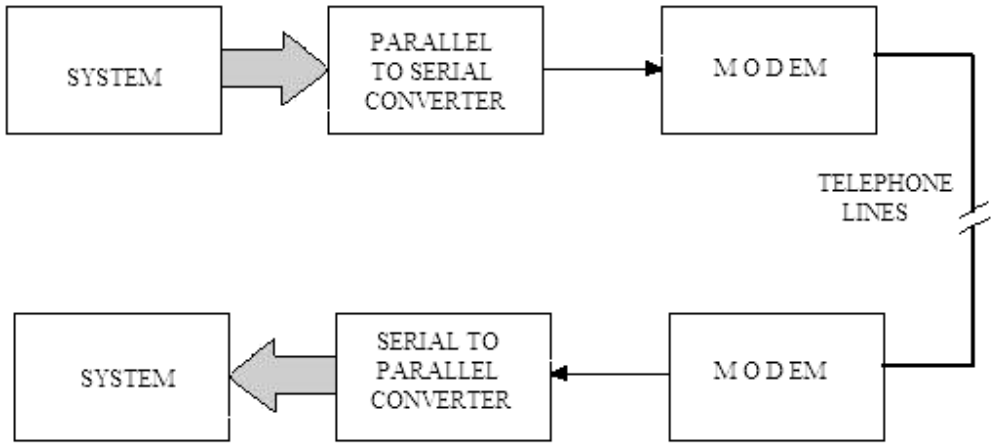


Fig. 14.4

receiving end as well as at the transmitting end. The modem at the transmitting end converts digital signal into analog signal format (audio frequency range) and at the receiving end the modem converts audio signal back to the digital data.

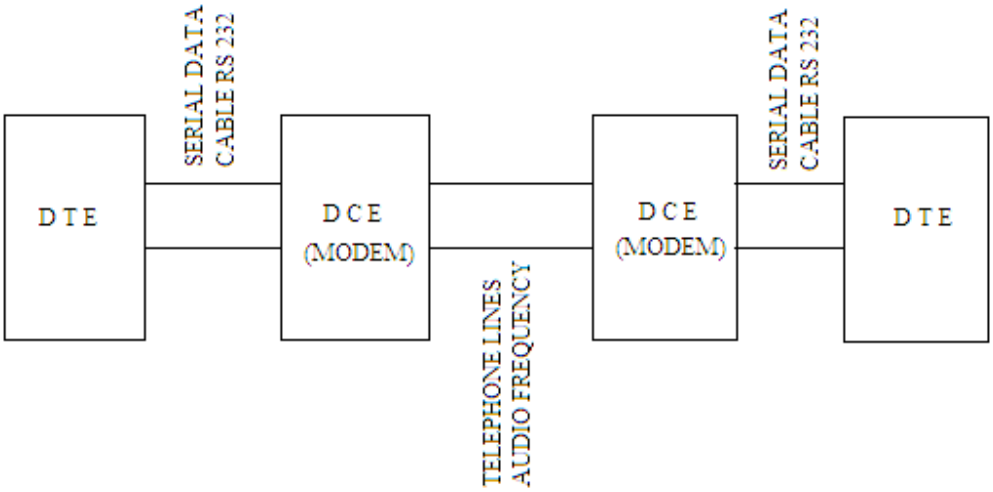


Fig. 14.5

Modems often use frequency shift keying technique for the conversion. It converts a 1 level signal into an audio signal of 1200 HZ and 0 level into audio signal of 2200 Hz. These converted audio signals are modulated on the carrier and then transmitted to the telephone lines. However at the receiving end another modem is used to convert the audio signal back to digital form. Figure 14.4 shows the block diagram of such transmission system. The originators and receptors of the digital data are called Data Terminal Equipments (DTE) and the modem units are called Data Communication Equipments (DCE). Figure 14.5 shows the modem connections using telephone lines.

14.3 SERIAL COMMUNICATION STANDARD

There are primarily two standards for transmitting the information in serial form. All the serial I/O devices work on either of these standards.

- Current Loop
- Voltage standard

In a current loop method, the current flows through the lines for the logic 1 and no current flow for the logic 0.

The two standards for the current loop are:

20 milliampere loop
60 milliampere loop.

A current loop reduces noise pickup and is suitable for long distance transmission.

The other standard is the voltage signal. When the data is transmitted as voltage, the commonly used standard is known as RS 232C. It is defined in reference to DTE (Data Terminal Equipment) and DCE (Data Communication Equipment) – terminal and Modem as shown in figure 14.6, which illustrates the connections of DTE to DCE. This standard was developed before the existence of TTL logics; its voltage levels are not compatible with TTL logic levels.

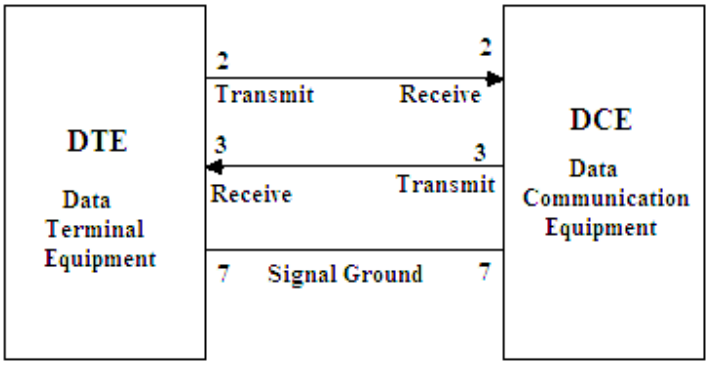


Fig. 14.6

RS 232C

Figure 14.7 shows the RS 232C 25-pin connector. The signals are divided into four groups namely:

- Data Signals
- Control Signals
- Timing Signals
- Ground

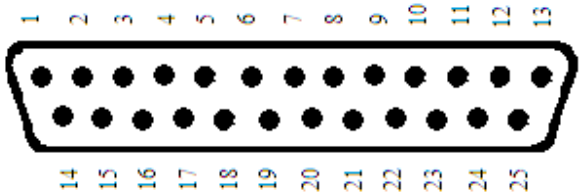


Fig. 14.7

The pin assignments are given below:

Pin 1: Protective Ground
Pin 2: Transmitted Data (T x D) → DCE
Pin 3: Received Data (R x D) → DTE
Pin 4: Request to Send (RTS) → DCE
Pin 5: Clear to Send (CTS) → DTE
Pin 6: Data Set Ready → DTE
Pin 7: Signal Ground
Pin 8: Received Line Signal Detector
Pin 9: Reserved for Data Set Testing
Pin 10: Reserved for Data Set Testing
Pin 11: Unused
Pin 12: Secondary Received Line Signal Detector
Pin 13: Secondary Clear to send
Pin 14: Secondary Transmitted Data
Pin15: Transmission Signal Element Timing (DCE Source)

Pin 16: Secondary Received Data
Pin 17: Receiver Signal Element Timing (DCE Source)
Pin 18: Unused
Pin 19: Secondary Request to Send
Pin 20: DCE ⟵ Data Terminal Ready (DTR)
Pin 21: Signal Quality Detector
Pin 22: Ring Indicator
Pin 23: Data Signal Rate Selector (DTE/DCE Source)
Pin 24: Transmit Signal Element Timing (DTE Source)
Pin 25: Unused

Technically the RS232C has -3V to -12V for logic '1' and +3V to +12V for logic '0'. This is negative logic. Because of incompatibility with TTL logic, voltage translator called line drivers and line receivers are required to interface TTL logic with RS232 signals. As shown in figure 14.6, the minimum interface requires three lines: pins 2, 3 and 7. These lines are defined in relation to the DTE; the terminal transmits on pin 2 and receives on pin 3. On the other hand, the DCE transmits on pin 3 and receives on pin 2.

The comparative study of synchronous and Asynchronous modes of data transfer to and from the serial I/O devices is given in table 14.1.

Table 14.1

Sr. No.	Format	Serial Transmission	
		Synchronous	**Asynchronous**
1.	Data format	Groups of characters	One character at a time.
2.	Framing	Synchronous characters are sent with each group.	START and STOP bit/bits are sent with each character.
3.	Speed	High speed, 20 K bits/sec or more.	Low, Less than 20 K bits/sec.
4.	Distance for communication.	Recommended for long distance communication.	Recommended for small distance communication.
5.	Implementation	Hardware.	Hardware or Software.
6.	Data direction	Simplex, half and full duplex.	Simplex, half and full duplex.

The computer communicates with the serial I/O devices either in Asynchronous or Synchronous modes. Here the discussion will be made on the Asynchronous mode of serial data transfer, as it is both hardware and software implemented.

14.4 ASYNCHRONOUS SOFTWARE APPROACH

In this case, the framing of each character is done by introducing START and STOP bits using the software. The data in parallel form is available with the microprocessor to be transmitted to serial I/O devices. The data is then converted to serial form using the software and sent to the serial I/O devices through an interfacing circuit. The interfacing circuit is to be designed for the microprocessor 8085A microprocessor is shown in figure 14.8. This circuit consists of D-type flip-flop (IC 7475) whose device address is chosen to be FF H. The address bus is decoded using an 8-inpout NAND gate (IC 7430), and combined with the control signal $\overline{IOW}$ to clock the latch. When the instruction OUT FF H is executed, the clock goes high and bit at the D_{IN} terminal is transferred to output Q of the flip-flop. The bit is latched as the clock goes low. If the Q is high 20 mA current flows through the TTY (Tele Type) circuit.

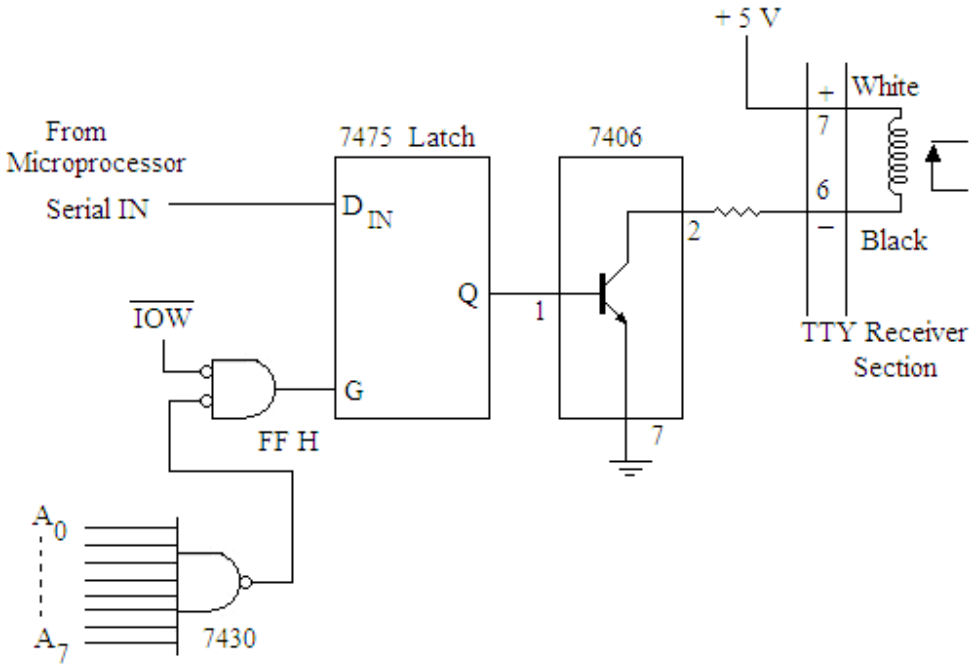


Fig. 14.8

Let us consider, a message is to be transmitted to teletype (TTY) using the 8085A microprocessor. The interfacing circuit, shown in figure 14.8, is used for this transmission. The characters of the message are stored in the memory locations starting at 2101 H. Each character includes framing information (START and STOP bits). Each character has 11 bits for the transmission; 8 bits for ASCII character, one START bit and two STOP bits. .Let the first character of the message is **W** (ASCII code for W is 57 H). Figure 14.9 shows a stream of eleven bits for this first ASCII character of the message. The character bits are transmitted beginning with the least significant bit (LSB) D_0. The

bit time, the delay between two successive bits, is determined by the transmission rate (baud rate).

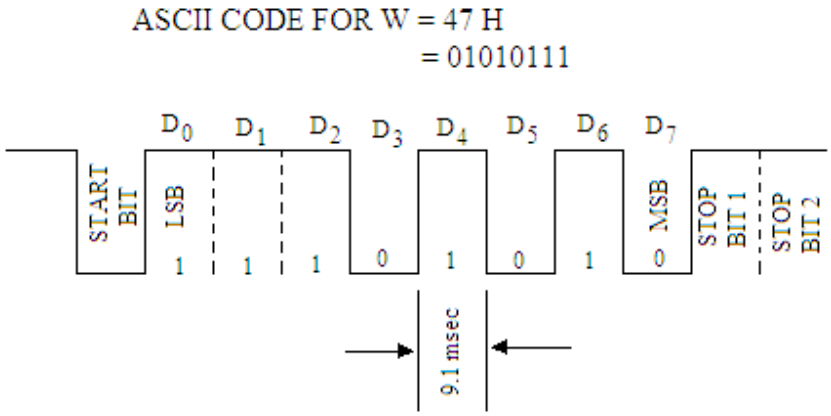


Fig. 14.9

A typical (TTY) sends (or receives) 10 ASCII characters per second. If each character has 11 bits, then TTY transmits 110 bits/second.

TTY transmission rate = 110 bits/second

Time for each bit $= \frac{1\text{sec}}{110}$

= 9.1 msec.

Therefore, microprocessor should send bit by bit information (including framing bits) to TTY to transmit the bit a time interval of 9.1 msec.

For the transmission the necessary software for 8085A is given as:

PROGRAM

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	; Initialize Stack pointer.
	LXI H,	2101 H	; Initialize H-L register pair for index pointer.
	MVI A,	01 H	; Set up MARK bit as 1.
	OUT	FF H	; Goes to the output of Flip-flop, whose address is FF H.
NXT	MOV A,	M	; Accumulator is loaded with the character of the message.
	CPI	0D H	; Compare if the character is Carriage return (0D H is the ASCII Code of carriage return).
	MOV B,	A	; Save the character in register B.
	JZ	END	; If the character is Carriage return, jump to END.

```
        CALL    SERIAL      ; Else Call subroutine for converting
                            into the serial form of the
                            character.
        INX H               ; Point to next character of the
                            message.
        JMP     NXT         ; Jump for next character.
END     CALL    SERIAL      ; Call subroutine for converting into
                            the serial form of the character
                            'Carriage return'.
        MVI B,  0A H        ; Load the character of Line feed
                            (0A H is the ASCII code for line
                            feed).
        CALL    SERIAL      ; Transmit line feed serially.
        HLT                 ; Stop Processing.
```

SUBROUTINE PROGRAM:

This is the subroutine for the conversion of bit of a character into serial form and then transmits to a TTY with a baud rate of 110.

```
Label   Mnemonics   Operand     Comments
SERIAL  MVI C,      08 H        ; Load register C with 08 H, which
                                is used as counter.
        MVI A,      00 H        ; Load START bit.
        OUT         FF H        ; Send the start bit at the output of
                                the flip-flop.
        CALL        DELAY       ; Call delay subroutine which
                                introduces a time delay of 9.1 msec
                                after START bit.
        MOV A,      B           ; Load the character into the
                                accumulator from register B.
NXT1    OUT         FF H        ; Send the bit.
        CALL        DELAY       ; Call delay subroutine which
                                introduces a time delay of 9.1 msec
                                between each bit.
        RRC                     ; Rotate right for the serial
                                conversion.
        DCR C                   ; Decrement C.
        JNZ         NXT1        ; If rotated 8 times then jump to
                                NXT1.
        MVI A,      01 H        ; Introduce STOP bit.
        OUT         FF H        ; STOP is sent to the output of the
                                flip-flop.
        CALL        DELAY       ; Call delay subroutine which
                                introduces a time delay of 9.1
                                msec.
```

```
        CALL    DELAY       ; Call delay subroutine which
                            introduces a time delay of 9.1
                            msec.
        RET                 ; Go back to main program.
```

Delay Program may be written for the time delay of 9.1 msec.

Similar, circuit and software may be used for the reception of the messages in the serial fashion.

The RIM and SIM instructions can also be used for the transmission and reception of data in serial fashion.

14.5 PROGRAMMABLE COMMUNICATION INTERFACE

Many types of UART (Universal Asynchronous Receiver Transmitter) and USART (Universal Synchronous/Asynchronous Receiver Transmitter) have been developed for data transfer between serial I/O device and the microcomputer in the form of the chips. The USART is most powerful and commonly used LSI chip. It provides a programmable serial communication interface between the microprocessor and serial I/O devices. The data transfer between the microprocessor and USART takes place in the parallel form. On the other hand the data transfer between the I/O devices and the USART is in the serial form. The USART has to be initialized before the data transfer takes place. The desired data format and synchronization method for the data transfer are specified during initialization by command byte written by the microprocessor for the USART. Thus an USART works as parallel to serial converter from microprocessor to I/O devices and vice-versa. Here we shall discuss the details of the popular USART chip 8251.

14.6 BLOCK DIAGRAM OF 8251A

The 8251A is a programmable communication interface available in the form of IC dual in line package. It consists of 28 pins and requires + 5 V d.c. supply for its operation. Figure 14.10 shows the pin diagram of 8251A. The pin details are as given below:

D_0-D_7	Data bus (8 bits)	$\overline{RxC}$	Receiver Clock
C/$\overline{D}$	Control or data to be written or read	RxD RxRDY	Receiver Data Receiver Ready
$\overline{RD}$	Read Data	TxRDY	Transmitter Ready
	Command	$\overline{DSR}$	Data Set Ready
$\overline{WR}$	Write Data or Control Command	$\overline{DTR}$ SYNDET/BD	Data Terminal Ready Sync/Break Detect
CLK	Clock Pulse (TTL)	$\overline{RTS}$	Request to Send Data
$\overline{CS}$	Chip Enable	$\overline{CTS}$	Clear to Send Data
RESET	RESET Input	TxE	Transmitter Empty
$\overline{TxC}$	Transmitter Clock	V_{CC}	+ 5 V
TxD	Transmitter Data	GND	Ground

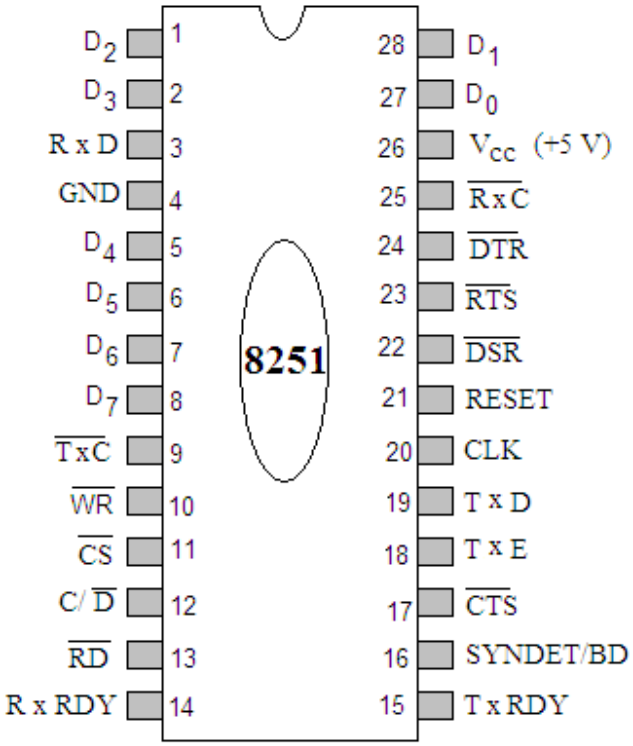


Fig. 14.10

The functional block diagram of 8251A is shown in figure 14.11. It includes the following four sections:

- **Read/Write Control Logic**
- **Transmitter Section**
- **Receiver Section**
- **Modem Control**

14.6.1 Read/Write Control Logic

This section includes 6-input signals: $\overline{CS}$, $C/\overline{D}$, $\overline{WR}$, *RESET* and *CLK*; and three buffer registers:

- **Control Register**
- **Status Register**
- **Data Bus Buffer Register**

$\overline{CS}$: It is a Chip Select terminal. A low to this terminal selects the 8251A for communication with the microprocessor. This is connected to a decoded address bus.

$C/\overline{D}$: It is a Control/Data pin. A high on this terminal addresses the control register or status register; whereas a low addresses the data bus buffer.

$\overline{WR}$: It is an active low Write signal. When a low signal is applied to this terminal, the microprocessor either writes in the control word register or

sends output to the data buffer. This pin is connected to either $\overline{IOW}$ or $\overline{MRMW}$.

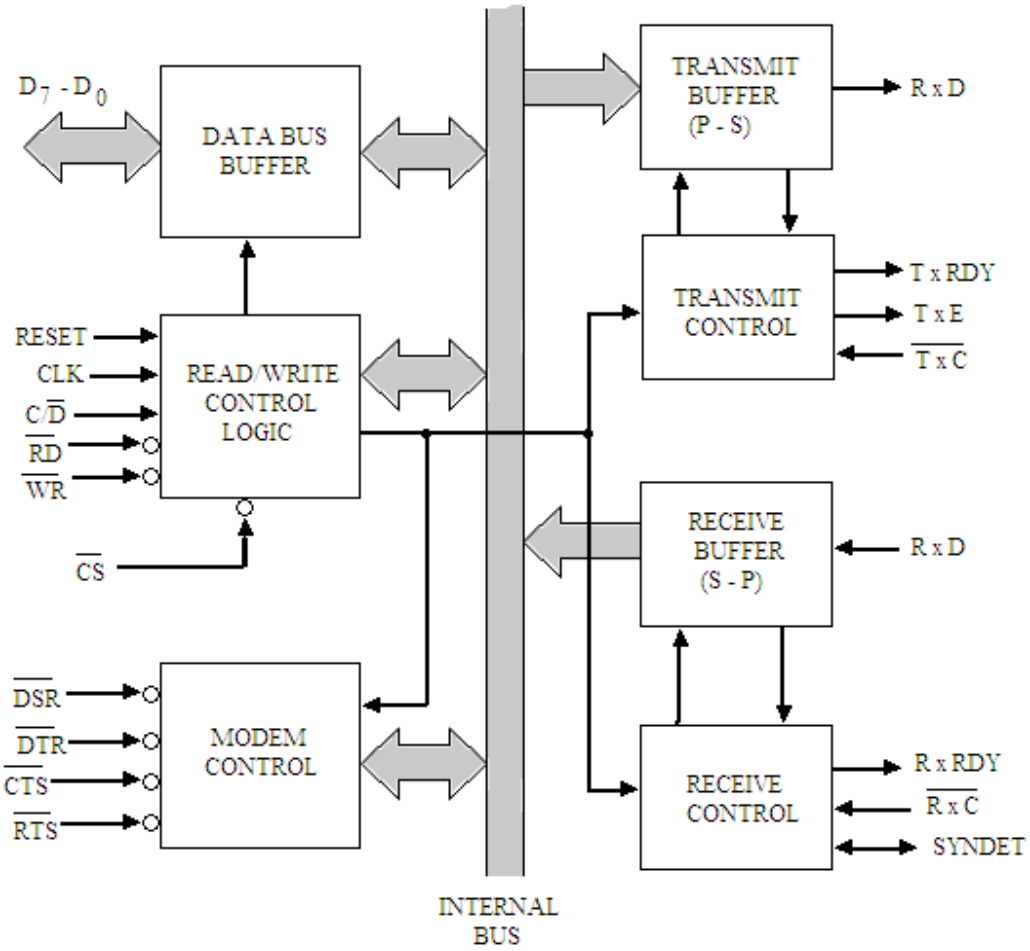


Fig. 14.11

$\overline{RD}$: It is an active low Read signal. When this terminal is low, the microprocessor either reads the status from the status register or accepts data from the data buffer. This is connected to either $\overline{IOR}$ or $\overline{MEMR}$.

RESET : It is a Reset input pin. When this terminal is high, it resets the 8251 and forces it in the idle mode.

CLK : This is the Clock input, usually connected with the system clock.

The details of the three registers are as:

Control Register:

This is a 16-bit register and contains the control word with two independent bytes.

The first byte is called the Mode Instruction Word, and

the second byte is called the Command Instruction Word.

This register can be accessed as an output port when $C / \overline{D}$ pin is high.

Status Register:

This input register checks the Ready status of a peripheral. This register is addressed as an input port when $C/\overline{D}$ terminal is high. It has the same port address as the control register.

Data Bus Buffer Register

This is a bidirectional register and can be addressed as an input port and an output port when $C/\overline{D}$ pin is low. Table 14.2 gives the summary of interfacing and control signals.

Table 14.2

$\overline{CS}$	$C/\overline{D}$	$\overline{RD}$	$\overline{WR}$	Functions
0	1	1	0	Microprocessor writes instruction in USART control register.
0	1	0	1	Microprocessor reads from USART status register.
0	0	0	1	Microprocessor outputs data to USART data buffer.
0	0	0	1	Microprocessor accepts data from USART data buffer.
1	X	X	X	USART not selected.

Figure 14.12 shows the expanded version of this section.

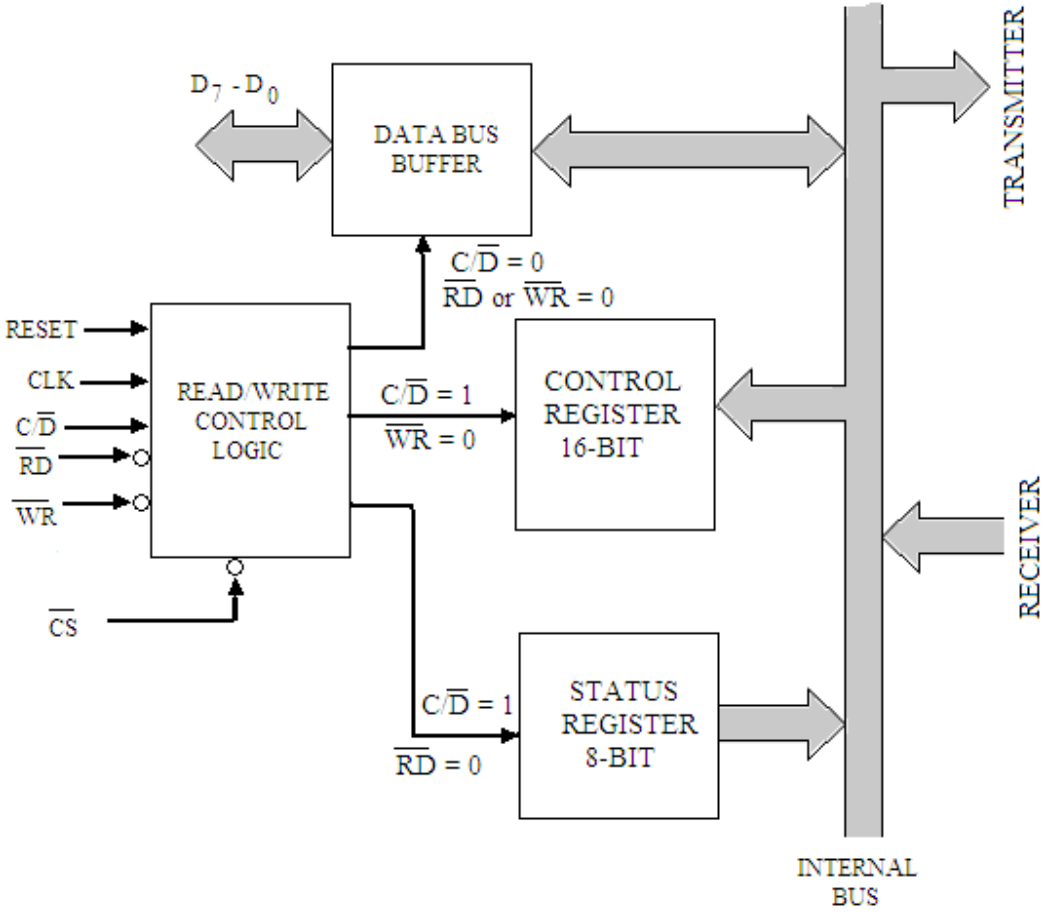


Fig. 14.12

14.6.2 Transmitter Section

The expanded block diagram of the transmitter section is shown in figure 14.13. It consists of the following registers:

- Transmitter Buffer Register – It holds 8-bit data.
- Serial Output Register – Converts 8-bits into a stream of serial bits.
- Transmitter Control Logic – It directs the output register to send the serial data at the output register through TxD terminal.

Three output signals and one input signal are also associated with the transmitter section. These signals are describes as:

TxD : It is **Transmit Data** terminal. The serial bits are transmitted on this line.

$\overline{TxC}$: This pin is **Transmitter Clock**. This controls the rate at which bits are to be transmitted by the 8251A. The clock frequency can be 1, 16 or 64 times of the baud.

TxRDY : This is **Transmitter Ready** pin. When this output terminal is high, it indicates that the buffer is empty and the 8251 is ready to accept a byte. It can be used either to interrupt the microprocessor or to indicate the status. This signal becomes reset when the data byte is loaded into the transmitter buffer register.

TxE : This is **Transmitter Empty** signal used as output terminal. A high on this terminal indicates that the output register is empty and becomes reset when a byte is transferred from the buffer register to the output register.

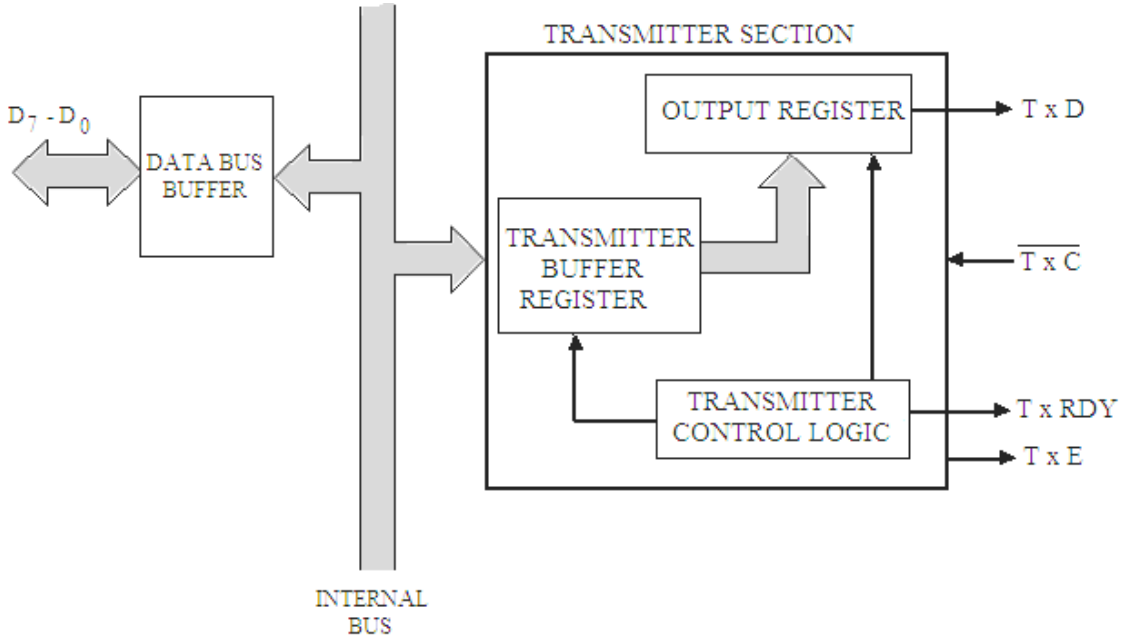


Fig. 14.13

14.6.3 Receiver Section

The expanded block diagram of the receiver section is shown in figure 14.14. It consists of the following registers:

- Receiver Buffer Register
- Serial Input Register
- Receiver Control Logic

The receiver section accepts serial data on the RxD terminal and converts it to parallel data. When the RxD line goes low, the control logic assumes that it is a START bit and waits for half a bit time, and samples the line again. It the line is still low, the input register accepts the next coming bits and forms a character. The character is then loaded to the buffer register. Subsequently, the parallel byte is transferred to the microprocessor when a request is made. Following signals are associated with the receiver section which are described below:

RxD : It is **Receive Data** terminal. The serial bits are received on this line and converted to a parallel byte in the receiver input register.

$\overline{RxC}$: This pin is **Receiver Clock**. This controls the rate at which bits are received by the 8251A. In the asynchronous mode, the clock can be set to 1, 16 or 64 times of the baud.

RxRDY : This is **Receiver Ready** pin. When this output terminal is high, the 8251A has a character in the buffer register and is ready to transfer it to the microprocessor. It can be used either to indicate the status or to interrupt the microprocessor.

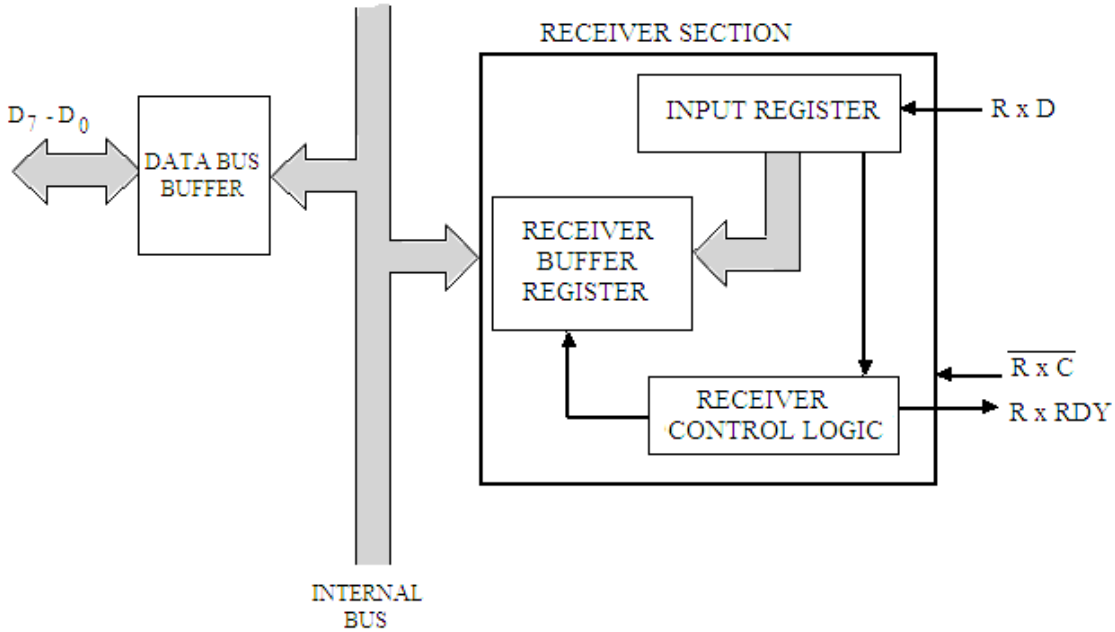


Fig. 14.14

14.6.4 Modem Control

The modem control section of the 8251A provides two input signals $\overline{DSR}$ (Data Set Ready) and $\overline{CTS}$ (Clear to Send); and two output control signals *DTR* (Data Terminal Ready) and *RTS* (Request to Send) to handle DTE and DCE. These signals are described as:

$\overline{\text{DSR}}$ (Data Set Ready): This is an active low input terminal used by the modem to indicate that it is ready for communication.

$\overline{\text{CTS}}$ (Clear to Send): This active low input terminal is used by the modem to signal the DTE that the communication channel is clear and it can send out the serial data.

DTR (Data Terminal Ready): This output signal is used by the 8251 to signal the modem to indicate that the terminal is ready to communicate.

RTS (Request to Send): This output signal is used by the 8251A to signal to modem that it has data to be transmitted.

14.7 INTERFACING OF 8251A

Figure 14.15 shows the interfacing connection of 8251A with the microprocessor. The eight data lines of 8251A are to be connected to the data bus of the microprocessor.

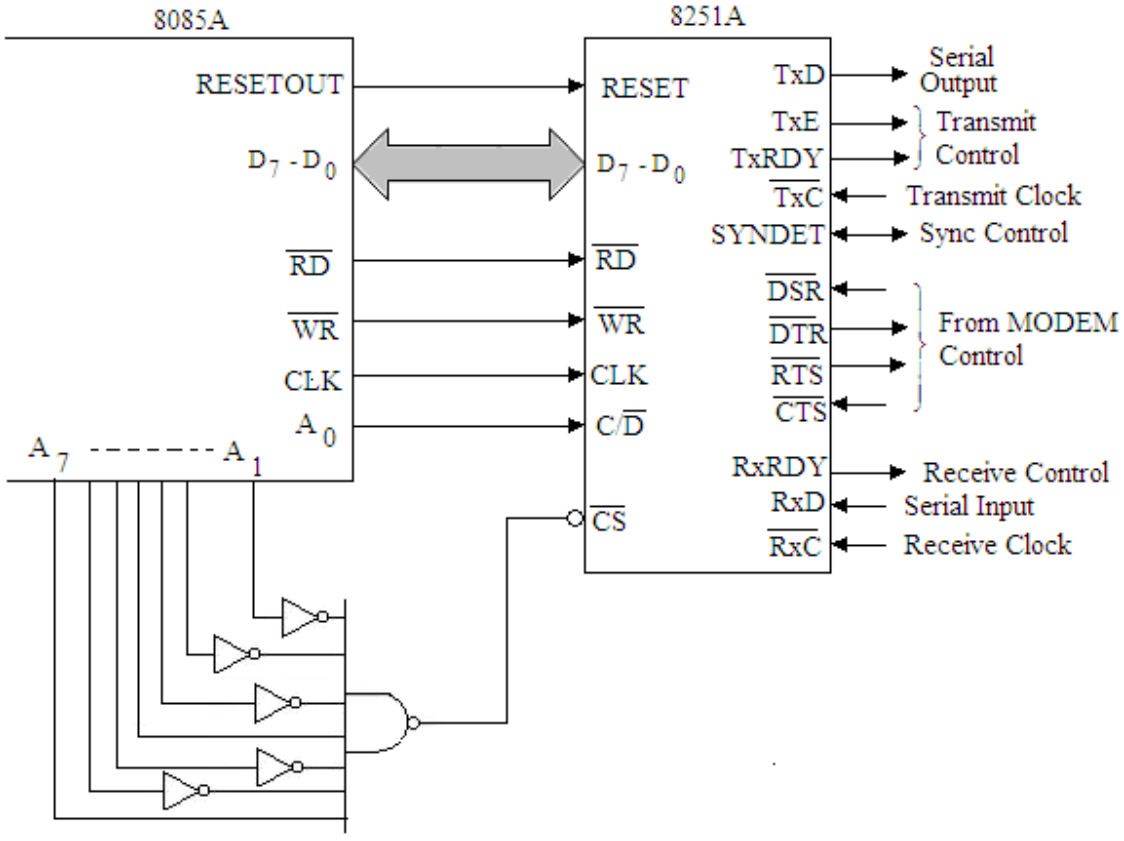


Fig. 14.15

The $\overline{RD}$ and $\overline{WR}$ lines of 8251A are to be connected to the $\overline{RD}$ and $\overline{WR}$ of the control lines of 8085A microprocessor respectively. The CLK pin of the 8251A is be connected to the CLK out terminal of the microprocessor. The RESET pin is connected to the RESETOUT of the microprocessor. The terminal $C/\overline{D}$ is used to select the internal registers either control register or data register. So it is connected to the A_0 address line of the microprocessor. The chip select terminal $\overline{CS}$ of the PCI 8251A is to be connected to the output of an address decoder circuit. The address decoder circuit uses the address lines A_1 to A_7 of the microprocessor. From the decoder circuit used in this figure, the chip select terminal of 8251A will be enabled when A_7 and A_4 are high and all other inputs are low. The port addresses for reading the data word, writing the data word, reading the status word and writing the control word will be as shown in table 14.3.

Table 14.3

A_0	$\overline{RD}$	$\overline{WR}$	Function	Port Address
0	0	1	Read Data Word	90 H
0	1	0	Write Data Word	90 H
1	0	1	Read Status Word	91 H
1	1	0	Write Control Word	91 H

14.8 PROGRAMMING OF 8251A

It has already been discussed that there is a 16 bit control register in a 8251A which contains two independent bytes (words). The first byte (word) is known as mode word which tells about the initialization parameters like mode (Asynchronous or Synchronous), baud, stop bits and parity bits etc. The second byte (word) is known as command word which tells about enabling the transmitter and receiver sections. The readiness of the peripherals can also be checked by reading the status word.

14.8.1 Initialization of 8251A in Asynchronous Mode

To initialize 8251A in asynchronous mode, a certain sequence of control words must be followed. After a reset operation (through system RESET or through instruction) a mode word must be written into the control register followed by a command word. Any control word written into the control register immediately after the mode word will be interpreted as a command word that means a command word can be changed anytime during the operation. However, the 8251A should be reset prior to writing a new mode word, and it can be reset using internal reset bit (D_6) in the command word.

Figure 14.16 shows the format of the mode word. The bits of this mode word are described below:

Bits D_1-D_0: These bits program the baud rate factor. These bits are set to 00 for synchronous operation. However, for asynchronous operation these bits specify the factor by which the transmit/receive clocks $\overline{TxC}$ and $\overline{RxC}$, exceeds the baud rate. The other clock inputs CLK to the 8251A is used to generate the internal device timing and must simply be greater than 30 times the transmitter/receiver baud rate.

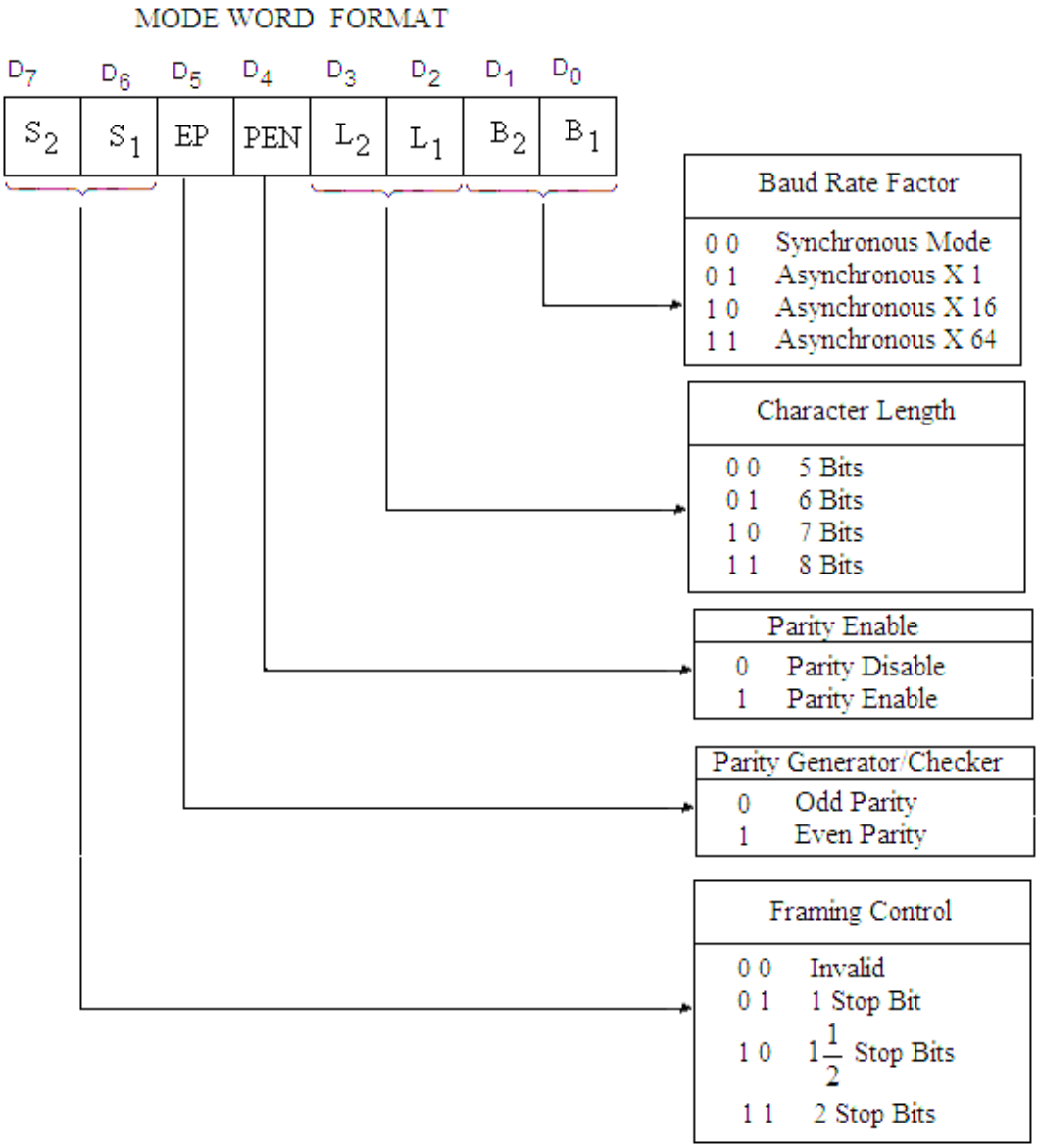


Fig. 14.16

Bits D_3-D_2: These bits specify the number of data bits in the character is to be sent or received.

Bit D_4: This bit enables the parity. If this bit is 0, the parity is disabled.

Bit D_5: This bit selects the even or odd parity. If this bit is 0, odd parity is selected. The even parity is selected if this bit is 1.

Bit D_6-D_7: These bits specify the number of stop bits.

Command Word

Once the function definition of the 8251A has been programmed by the mode word instruction, the device is ready for data communication. The command word

instruction controls the actual operation of the selected format. Functions such as Enable Transmit/ Receive, Error Reset and modem control are provided by the command word instruction. The format of the command word is shown in figure 14.17.

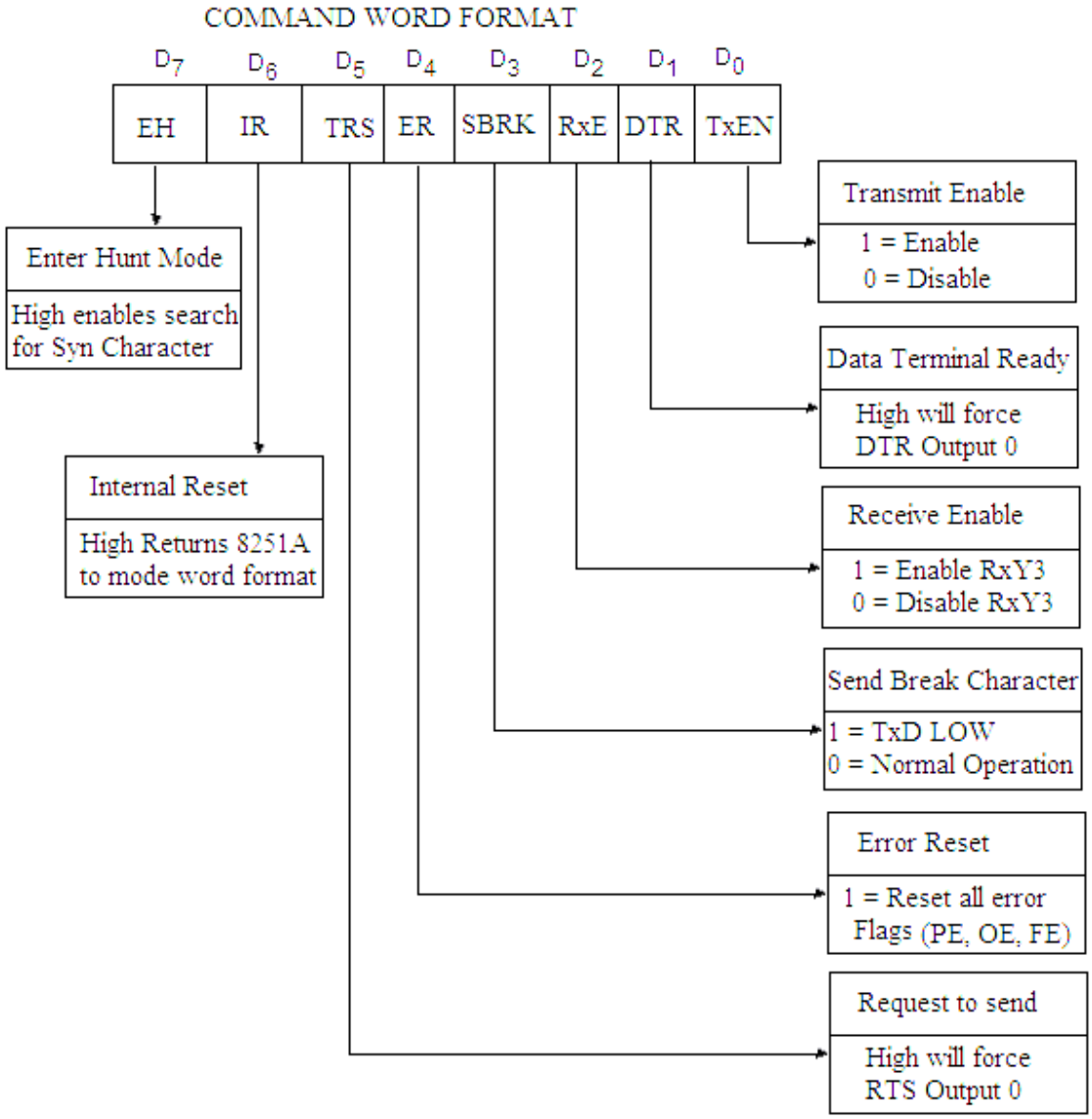


Fig. 14.17

The bits of the command word are described below:

Bit D_0: This bit enables the Transmitter.

Bit D_1: This bit controls the Data Terminal Ready.

Bit D_2: This bit enables the Receiver.

Bit D_3: This bit makes the transmitter to send continuous break characters.

Bit D_4: This resets the error flags. It resets Parity Error flag (PE), Over Run Error flag (OE) and Framing Error flag (FE).

Bit D_5: This bit controls the request to send to RTS pin of the device. A high at this bit makes the RTS pin to become active.

Bit D_6: It is used as an internal reset. A high to this bit resets the device, so that a new mode word can be entered.

Bit D_7: It is used in synchronous mode. It enables the receiver to look for the synchronous data.

Programming Sequence

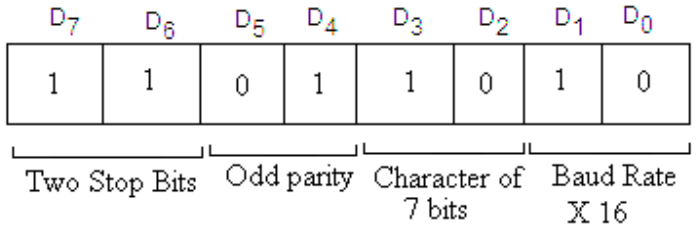
When programming the 8251A in asynchronous mode, following sequence must be followed:

1. Reset (either internal or external)
2. Mode Instruction (specify the asynchronous mode)
3. Command Instruction

The resetting of the 8251A can be done by loading the control register with a command word whose D_6 bit is high. This bit reset the device. The command word is, therefore, loaded as:

```
MVI A, 40 H     ; Command word whose D6 bit is high is loaded to accumulator.
OUT 91 H        ; Command word is loaded to control register whose port address is 91 H as discussed earlier.
```

Now the mode instruction word is loaded to the control register. The mode word is formed as per the required modes of transmission. For example, for asynchronous transmission with 7 data bits, 2 stop bit and odd parity; also a 16 X clock is used. For this the format is shown in figure 14.18 and mode word will be given as:



$= DA\ H$

Fig. 14.18

```
MVI A, DA H     ; Mode word is loaded to accumulator.
OUT 91 H        ; Mode word is loaded into control register.
```

The command word with RTS, error reset and DTR enable with be given as shown in figure 14.19.

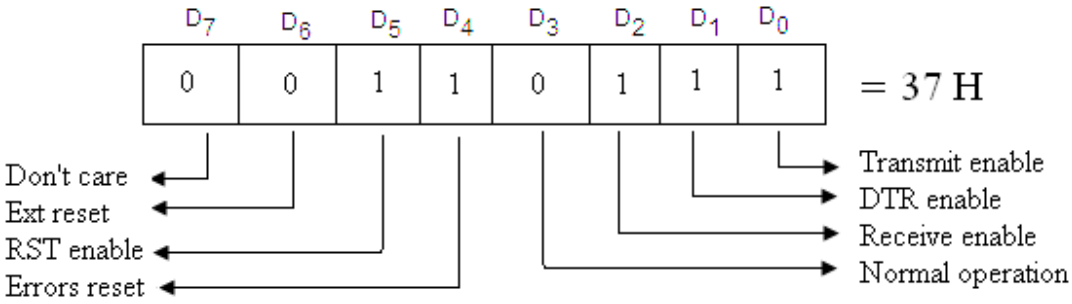


Fig. 14.19

The command word to be loaded in the control register is given below:

```
MVI A, 37 H          ; Command word is loaded to accumulator.
OUT 91 H             ; Command word is loaded into control
                       register.
```

For initializing the 8251A a dummy mode is sent before resetting it. The resetting is done using command word with D_6 bit as 1. The mode word followed by the command word is sent. With all these requirements the initialization program of 8251A is writhen as:

```
MVI A, 00 H
OUT 91 H             ; Dummy mode word.
MVIA, 40 H           ; Resetting of 8251A by using the command
                       word with D6 bit as high.
OUT 91 H
MVI A, DA H          ; Mode word is loaded to accumulator.
OUT 91 H             ; Mode word is loaded into control register.
MVI A, 37 H          ; Command word is loaded to accumulator.
OUT 91 H             ; Command word is loaded into control
                       register.
```

14.8.2 Initialization of 8251A in Synchronous Mode

When programming the 8251A in asynchronous mode, following sequence must be followed:

1. Reset (either internal or external)
2. Mode Instruction (specify the Synchronous Mode and number of synchronizing characters)

3. One or two synchronizing character
4. Command Instruction

For initializing of the 8251A in synchronous mode the mode instruction word is to be written only after resetting the PCI as in the case of asynchronous mode. Then with one or two sync characters, a command word is written in the control register. The mode word format in synchronous operation is shown in figure 14.20. As discussed earlier the D_1 and D_0 bits program the baud rate factor which are set 00 for synchronous operation.

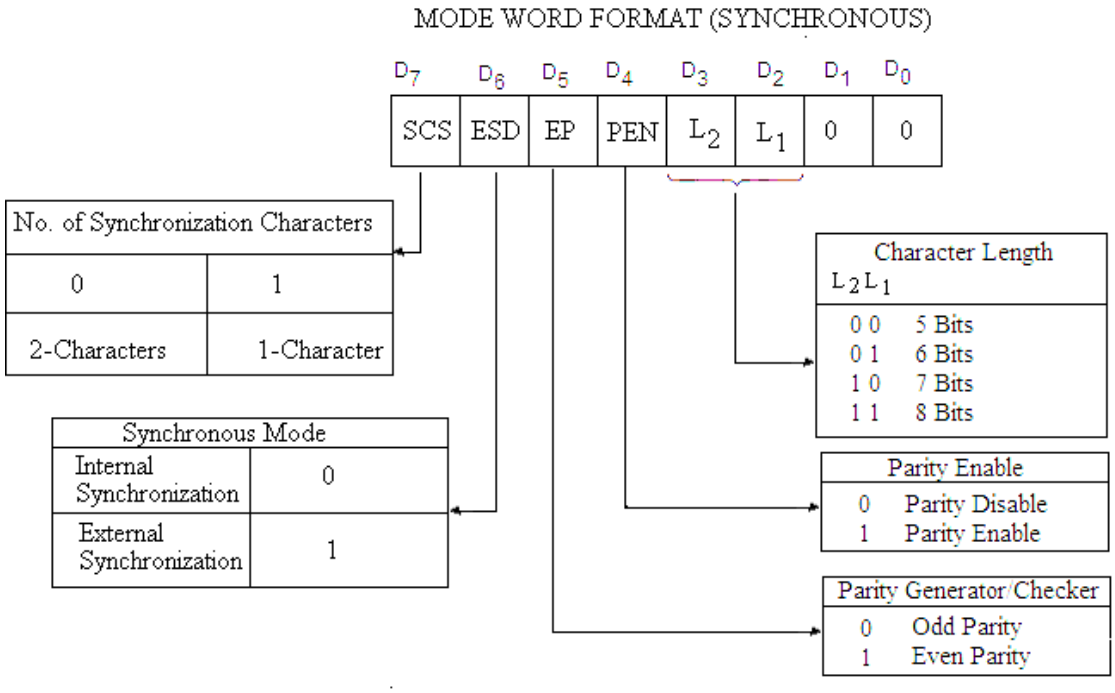


Fig. 14.20

The format to be used for command instruction format for the synchronous operation is the same as for the asynchronous operation which is shown in figure 14.21. Once the functional definition of the 8251A has been programmed by the mode instruction and the Sync characters are loaded (in synchronous mode) then the device is ready to be used for data communication. The command instruction controls the actual operation of the selected format. Functions such as Enable Transmit/ Receive, Error Reset etc are provided by the command instruction.

Once the mode instruction has been written into the 8251A and Sync characters inserted, then all further "control writes" ($C / \overline{D} = 1$) will load a command instruction. A reset operation (internal or external) will return the 8251A to the mode instruction format.

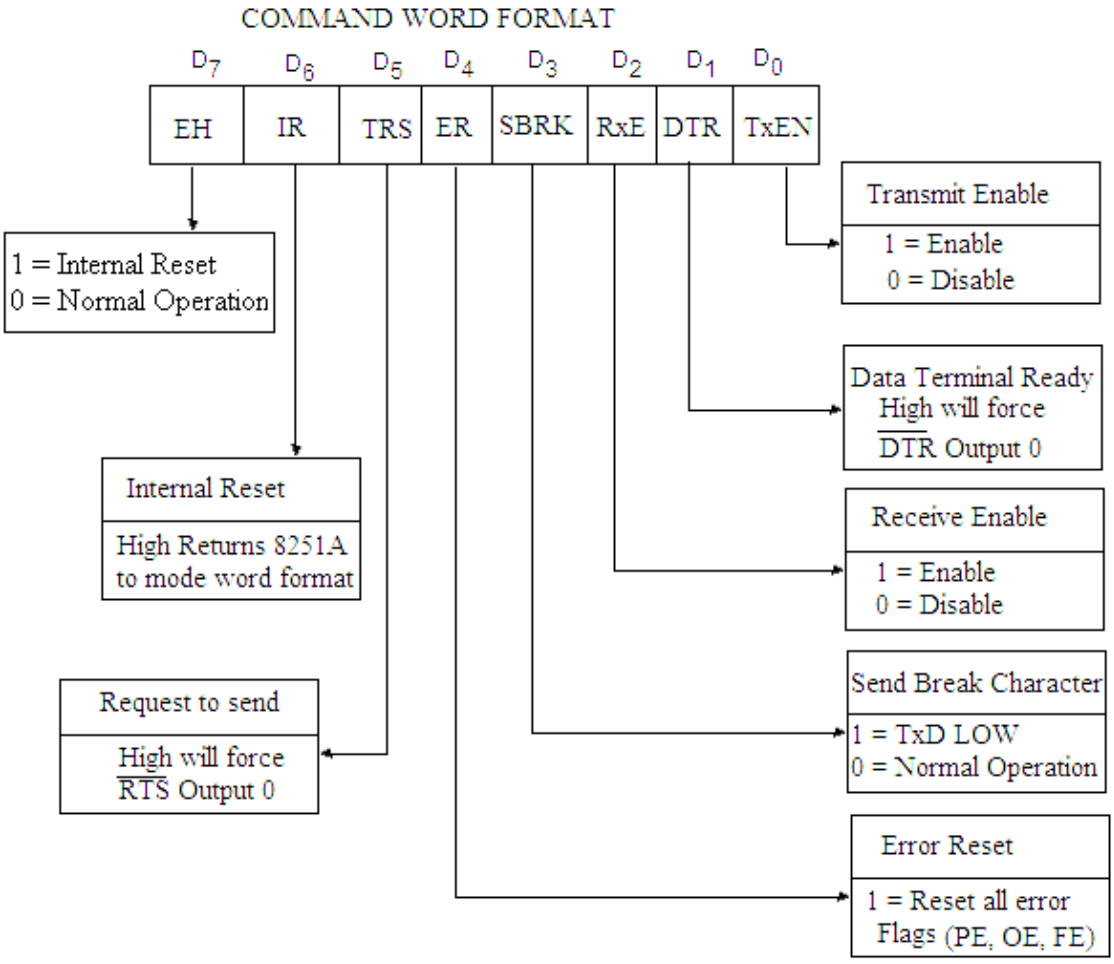


Fig. 14.21

In data communication system, it is necessary to examine the status of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251A has facilities that allow the programmer to read the status of device at any time during functional operation. A normal read command is issued by the CPU with $C/\overline{D}=1$ to accomplish the function.

The microprocessor can output the data to be transmitted only after checking the status TxRDY (D_0) bit of status word as shown in figure 14.22. The sync characters will automatically be inserted if the buffer of the transmitter becomes empty at any time.

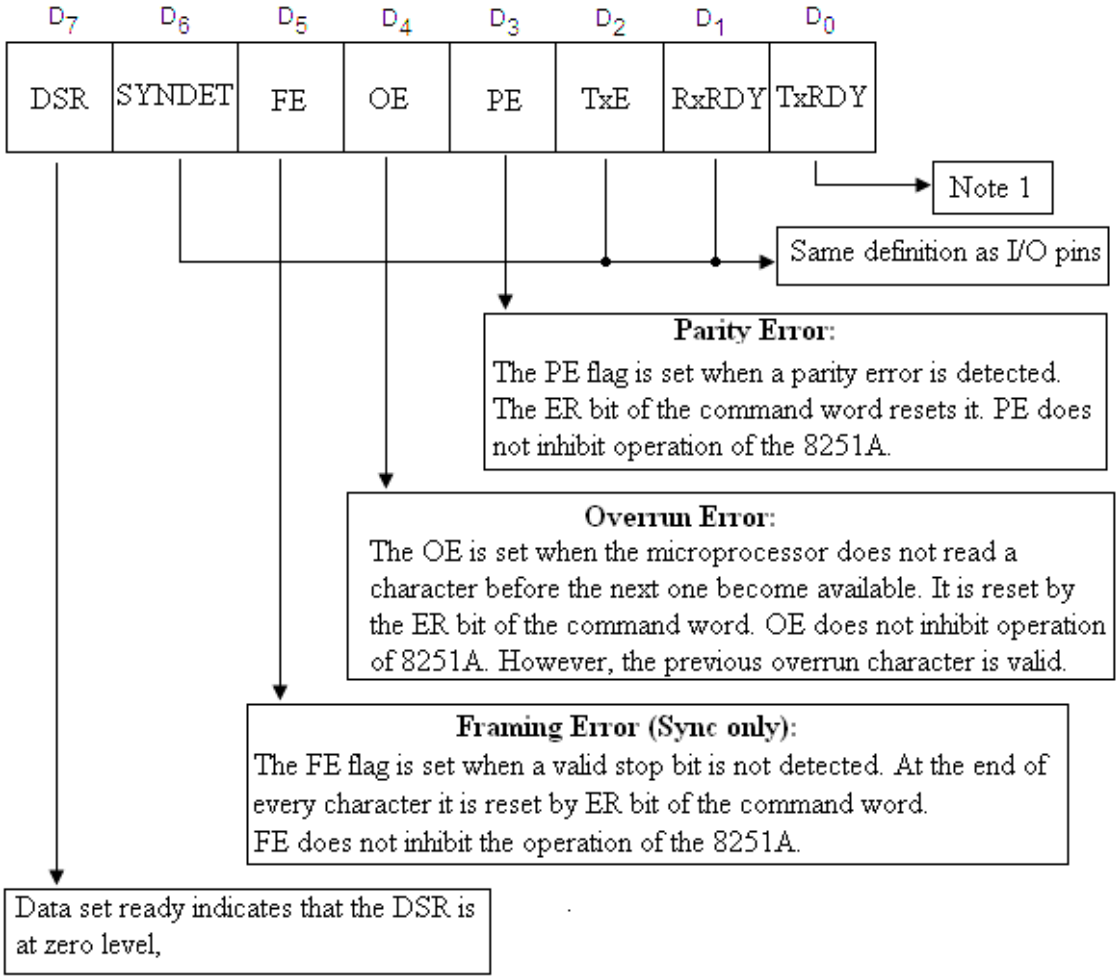


Fig. 14.22

Example 14.1 *Write a subroutine program to transmit serially 256 bytes of data stored in the memory locations starting at 3000 H. The bytes are to be transmitted in synchronous mode (without parity) with two sync characters using 8251A. Consider 90 H and 91 H are the port addresses for control/status register and data register respectively.*

Solution. The mode word format for the given problem is shown in figure 14.23.

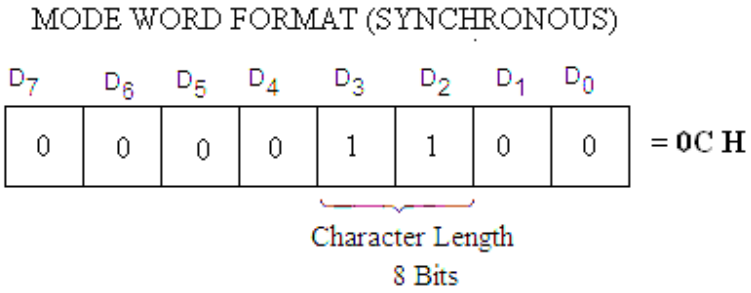


Fig. 14.23

The subroutine program may then be given as:

SUBROUTINE PROGRAM

Label	**Mnemonics**	**Operand**	**Comments**
	LXI H,	3000 H	; Initialize H-L register pair for index pointer.
	MVI C,	FF H	; Register C to be used as counter.
	MVI A,	00 H	; Dummy word
	OUT	91 H	
	MVI A,	40 H	; Command word for resetting 8251A.
	OUT	91 H	; Command word loaded to the control register.
	MVI A,	0C H	; Mode word 0C H is loaded to the accumulator.
	OUT	91 H	; The mode word for two sync character is loaded to the control register.
STATUS	IN	90 H	; Read the status word.
	ANI	01 H	; Mask all the bits except D_0 for checking the status.
	JZ	STATUS	; If D_0 bit is zero jump to STATUS to read again.
	MOV A,	M	; Loads the character to be transmitted to the accumulator.
	OUT	91 H	; Character is transmitted.
	DCR	C	; Decrements the data in C register for next byte for transmission.
	JNZ	STATUS	; If all bytes are not transmitted then jump to STATUS to read the next byte.
	RET		; Returns to main program.

PROBLEMS

1. What are the advantages and disadvantages of serial I/O data transfer over the parallel I/O data transfer?
2. Name three methods of serial mode of data transfer and explain them in brief.
3. Draw and explain the block diagram of Programmable Communication Interface 8251A.
4. Discuss the Transmission section of 8251A.
5. Discuss the Receiver section of 8251A.
6. Explain the working principle of RS 232 interface.
7. Explain how Programmable Communication Interface 8251A is interface with the CPU.
8. Draw the schematic diagram of interfacing 8251A with 8085 microprocessor. The connections should be such that the port address for control register and data registers are 71 H and 70 H respectively.
9. Explain the mode word format of 8251A.
10. Explain the command word format of 8251A.
11. Explain how the 8251A is initialized in Asynchronous mode.
12. Explain hoe the 8251A is initialized in synchronous mode.
13. Explain the following terms related to 8251A:
    $C/\overline{D}$, $\overline{RxC}$, RxD, $\overline{DSR}$, $\overline{TxC}$, TxD, $\overline{DTR}$, $\overline{CTS}$
14. Explain the following terms related to 8251A:
    $RxRDY$, $TxRDY$, $TxEMPTY$

15

Applications of Microprocessor

After the study of all the details of microprocessor 8085, including the assembly language programming and peripheral devices we are now in a position to design some microprocessor based systems. Real time clock with on/off timer, running light, washing machine control, water level control etc. are some designs will be discussed in this chapter. These designs are supposed to be very interesting and useful. The assembly language programming of these designs have been verified on M/S Vinytics Kit

15.1 REAL TIME CLOCK WITH ON/OFF TIMER

In the present section of this chapter, the details of the assembly language programming of the design of Real Time Clock with On/Off Timer will be discussed. In this microprocessor based design six FNDs of the microprocessor kit (four of address field and two of data field) have been used to display the current time. The four 7-segments of the address field would display hours and minutes of the current time and two 7- segments of data field would display the seconds of the current time. For example, current time is 10:25:30, the address and data field of the kit would display as shown in Fig. 15.1. It would then continuously update the current time after every second. In other words, it will work as the real time clock (digital clock).

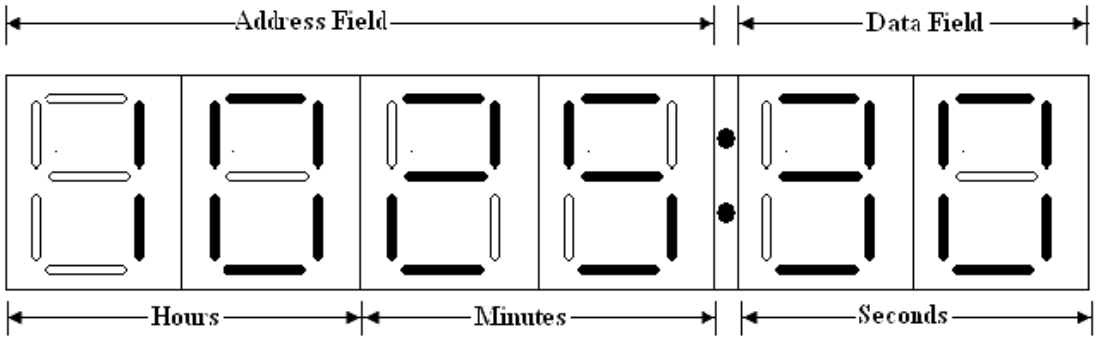


Fig. 15.1

For the working of real time clock, the software prepared was checked on the Vinytics kit (VMC-8506) and found to work satisfactorily. Two monitor programs (stored in Kit's ROM /EPROM) at the locations 0347 H (for clearing the display) and 05D0 H (displaying the contents of memory of memory locations 2050 H through 2055 H in the address and data fields respectively) have been used in the program. Please note that before calling the display routine, registers A and B are required to be initialized with either 00 or 01 to indicate to the monitor program as to where the contents of above mentioned memory locations are to be displayed (e.g. address field or data field) and whether a dot is to be displayed at the end of address field or not. The assembly language program was executed to switch ON an electrical appliance after a preset time period. This program was proved to be very useful microprocessor based system. This type of microprocessor based system has the useful requirement in research laboratories. An electronic circuit was also designed which was interfaced with the 8085A microprocessor through the programmable peripheral Interface (PPI) 8255A. The PPI-8255A was available with the microprocessor kit.

ASSEMBLY LANGUAGE PROGRAM

The assembly language program for the universal timer and for switching ON electrical device is given below:

PROGRAM

Main Program

Label	Mnemonics	Operand	Comments
	LXI SP,	27FF H	;Initialize Stack Pointer.
	MVI A,	80H	;Control word for 8255-I.
	OUT	03H	;Work all the ports of 8255-I as output port .
	CALL	0347 H	;Clears the display. It is the program stored in ROM of the kit.
AA	XRA	A	;Clears the accumulator
	MOV B,	A	;Clears the B register also.

```
        LXI H,      2050 H      ;Intialize H-L pair where the
                                 current time is stored.
        CALL        05D0 H      ;Displays the current time in
                                 address field. It is the
                                 program stored in ROM of
                                 the Kit.
        MVI A,      01 H        ;Stores 01 H in accumulator.
        MVI B,      00 H        ;Stores 00 H in B register.
        LXI H,      2054 H      ;Initialize H-L register pair
                                 with 2054 H.

        CALL        05D0 H      :Displays the current time in
                                 data field.
        CALL        ONOFF       ;Calls subroutine program to
                                 switch on the electrical
                                 appliance.
        LXI H,      2055 H      ;Initialize H-L register pair
                                 with 2055 H.
        MOV A,      M           ;Moves the least significant
                                 digit (LSD) of current
                                 seconds to the accumulator.
        ADI         01 H        ;Add 01 to the accumulator.
        CPI         0A H        ;Compares with 0A H.
        JZ          RR          ;If Acc.=0A H, then jump to
                                 RR, indicating that LSD of
                                 seconds has become 9. Else
                                 goes to next statement.
        MOV M,      A           ;Moves the accumulator
                                 contents to memory
                                 location addressed by H-L
                                 register pair i.e. current
                                 seconds are stored back in
                                 the memory locations.
DD      MVI B,      02 H        ;Stores 02 H in accumulator.
YY      LXI D,      FA00 H      ;Intialize D-E register pair
                                 with FA00H.
        CALL        DELAY       ;Calls the delay program.
        DCR         B           ;Decrement B- register.
        JNZ         YY          ;If B≠0 then jump to YY.
        JMP         AA          ;Jump to AA for the display
                                 of current time.
RR      MVI A,      00 H        ;LSD of current time
                                 becomes 0, which is stored
                                 in accumulator.
        MOV M,      A           ;Stores it to the memory
                                 location of LSD of seconds
```

```
                                        whose address is given in
                                        H-L register pair.
        DCX     H                       ;Decrement H-L register
                                        pair.
        MOV A,  M                       ;Moves MSD of the current
                                        second to accumulator.
        ADI     01H                     ;Add 01 H to it
        CPI     06 H                    ;Compaers if MSD of the
                                        current seconds is 06.
        JZ      UU                      ;If Acc.=06 then jump to
                                        UU, indicating that MSD of
                                        seconds has become 6. Else
                                        goes to next statement.
        MOV M,  A                       ;If A≠06 then stores to the
                                        memory location addressed
                                        by the H-L register pair.
        JMP     DD                      ;Jump to DD for 1 sec.delay.
UU      MVI A   00 H                    ;Store 00 to accumulator.
        MOV M,  A                       ;Stores  00  to the memory
                                        location of MSD of seconds
                                        addressed by the H-L
                                        register pair.
        DCX     H                       ;Decrement H-L register
                                        pair.
        MOV A,  M                       ;Moves the LSD of the
                                        current minutes to
                                        accumulator.
        ADI     01 H                    ;Add 01 H to the LSD of the
                                        current minutes.
        CPI     0A H                    ;Compares it with 0A H.
        JZ      VV                      ;If LSD of minutes is 0A H,
                                        then jump to VV else to the
                                        next instruction.
        MOV  M, A                       ;Stores it to the memory
                                        location of LSD of minutes
                                        addressed by H-L register
                                        pair.
        JMP     DD                      ;Jump to DD for 1 sec delay.
VV      MVI A,  00 H                    ;Stores 00 H to accumulator.
        MOV M,  A                       ;Stores 00 to the memory
                                        location of LSD of minutes
                                        addressed by H-L register
                                        pair.
        DCX     H                       ;Decrement H-L register
                                        pair.
```

```
        MOV A,    M          ;Moves the MSD of the
                               current minutes to
                               accumulator.
        ADI       01 H       ;Add 01 H to the MSD of
                               the current minutes.
        CPI       06 H       ;Compaers if MSD of the
                               current minutes is 06.
        JZ        SS         ;If Acc.=06 then jump to SS,
                               indicating that MSD of
                               minutes has become 6. Else
                               goes to next statement.
        MOV M,    A          ;If Acc≠06 then stores to the
                               memory location addressed
                               by the H-L register pair.
        JMP       DD         ;Jump to DD for delay of 1
                               sec.
SS      MVI A     00 H       ;Store 00 to accumulator.
        MOV M,    A          ;Stores  00  to the memory
                               location of MSD of
                               minutes addressed by the
                               H-L register pair.
        DCX       H          ;Decrement H-L register
                               pair.
        MOV A,    M          ;Moves the LSD of the
                               current hrs. to accumulator.
        ADI       01 H       ;Add 01 H to the LSD of the
                               current hrs.
      * CPI       03 H       ;Compares it with 03 H.
        JZ        LL         ;If LSD of hrs is 03 H, then
                               jump to LLelse to the next
                               instruction.
        MOV  M,   A          ;Stores it to the memory
                               location of LSD of hrs.
                               addressed by H-L register
                               pair.
        CPI       0A H       ;It is also compared by 0AH.
        JZ        WW H       ;If it is 0A H, then jump to
                               EE else go to next
                               instruction.
        MOV M,    A          ;Stores it to memory
                               locations addressed by H-L
                               register pair.
        JMP       DD         ;Jump to DD for delay of
                               1sec.
LL      DCX       H          ;Decrement H-L register
                               pair.
```

```
        MOV A,    M          ;Moves the contents  of
                               memory location addressed
                               by H-L register pair to Acc.
    **  CPI       01 H       ;Compares it with 01 H.
        JZ        GG         ;If it is 01 H, then jump to
                               GG H.
        INX       H          ;Increment the H-L register
                               pair.
        MOV A,    M          ;Moves the contents of M(H-L)
                               to the Acc.
        ADI       01 H       ;Add 01 H to it.
        CPI       0A H       ;Compares it with 0A H.
        JZ        WW         ;If it is 0A H then jump to
                               WW.
        MOV M,    A          ;Else stores the accumulator
                               contents in M(H-L).
        JMP   DD             ;Jumps to DD for the delay
                               of 1sec.
GG      MVI A,    00 H       ;Stores 00 H to Acc.
        MOV M,    A          ;Moves 00 H to the memory
                               location addressed by H-L
                               register pair.
        INX       H          ;Increments the H-L register
                               pair.
    ***MVI A,     01 H       ;Loads Acc. to 01 H.
        MOV M,    A          ;Moves the Acc. Contents to
                               M(H-L).
        JMP       DD         ;Jumps to DD for the delay
                               of 1sec.
WW      MVI A,    00 H       ;Stores 00 H to accumulator.
        MOV M,    A          ;Stores it to M(H-L).
        DCX       H          ;Decrements the H-L
                               register pair.
        MOV A,    M          ;Moves M(H-L)to accumulator.
        ADI       01 H       ;Add 01 H to it.
        MOV M,    A          ;Acc. Contents are loaded to
                               M(H-L).
        JMP       DD         ;Jump for delay of 1sec.
```

Delay Subroutine Program

Label	Mnemonics	Operand	Comments
DELAY	DCX	D	;Decrement D-E register pair.

	MOV A,	D	;Moves the contents stored in M_{D-E} to Acc.
	ORA	E	;The contents of A and E registers are ORed bit by bit.
	JNZ	DELAY	;If the result is not zero then jump to DELAY else next instruction.
	RET		;Return to main program.

Subroutine Program to check the time (ON time) after every second

Label	Mnemonics	Operand	Comments
ONOFF	LXI H,	2055 H	;Initialize H-L register pair with 2055 H.
	LXI D,	205C H	;Initialize D-E register pair with 205C H.
	MVI B,	06 H	;Stores 06 H to B-register.
AGAIN	LDAX D		;Loads the contents of M_{D-E} to the accumulator.
	CMP	M	;Compares it with M_{H-L}.
	RNZ		;Return if not zero.
	DCX	H	;Decrements H-L register pair.
	DCX	D	; Decrements D-E register pair.
	DCR	B	;Decrements the contents of B-register.
	JNZ	AGAIN	;Jump if not zero to AGAIN.
	MVI A,	01 H	;Stores 01 H to Acc.
	OUT	00 H	:D0 bit of port A of 8255 becomes high to energies the relay (to switch on the appliance).
	RET		;Returns to main program.

DATA

2050 H	-	MSD of hrs.	Current Time
2051 H	-	LSD of hrs.	
2052 H	-	MSD of Minutes	
2053 H	-	LSD of Minutes	
2054 H	-	MSD of Seconds	
2055 H	-	LSD of Seconds	

2056 H	-	01H (if wish to switch ON the electrical appliance)	
2057 H	-	MSD of Hrs.	
2058 H	-	LSD of Hrs.	
2059 H	-	MSD of Minutes	
205A H	-	LSD of Minutes	Time For On Electrical
205B H	-	MSD of Seconds	Appliance
205C H	-	LSD pf Seconds	

This program written in assembly language is self-explanatory. The comment column will help in understanding the operation of the project. The current time is stored in the beginning before the start of the program in the memory locations starting at 2050 H to 2055 H. In the memory location 2056 H, 01H is stored if we wish to switch on the electrical device connected with the PPI-8255 A; otherwise any number can be stored in this memory location. If 01H is stored in the memory location 2056H, then the time for switching ON the electrical appliance is stored in the memory location starting at 2057 H to 205C H. Suppose current time, when we wish to start the program is:

Hrs.	Min.	Sec.
02	25	30

And we wish to switch on the electrical appliance at:

Hrs.	Min.	Sec.
04	23	00

The data to be stored in the memory locations starting at 2050 to 205C is given below:

Memory Location	**Data**		
2050 H	00 H	02 Hrs.	
2051 H	02 H		
2052 H	02 H	25 Minutes	Current Time
2053 H	05 H		
2054 H	03 H	30 Seconds	
2055 H	00 H		
2056 H	01 H	01 for switch ON the appliance	
2057 H	00 H	04 Hrs.	
2058 H	04 H		Time when
2059 H	02 H	23 Minutes	electrical
205A H	03 H		appliance to
205B H	00 H	00 Seconds	ON
205C H	00 H		

A relay circuit to be connected to the Programmable Peripheral Interface IC 8255-I is shown in Fig. 15.2.

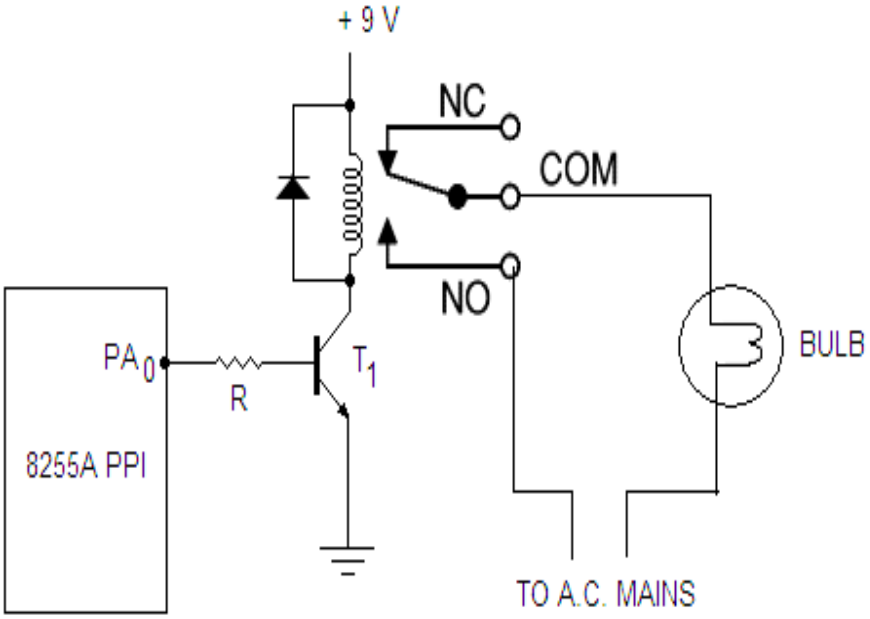


Fig.15.2

This circuit consists of a relay, a diode and transistor. The base of the transistor is connected to PA_0 of 8255 although a 10 k resistance. A diode IN4007 is connected in parallel with coil of 12V relay. The electrical appliance is connected to the a.c.mains through the N/O pins of the relay.

The working of this circuit may be explained as:

When a high signal is available through the software to a PA_0 (D_0 bit of port A) of 8255-I, the transistor goes into saturation. The relay is energized. The N/O terminals of the relay get connected and electrical appliance become ON.

When the program is executed, the current time is displayed in the address and data fields of the microprocessor kit. The current time is updated after every second by the programming. For this LSD (Least significant Digit) of the seconds is transferred to accumulator, which is added with 1. The contents after addition are compared with 0A H (decimal number 10). In other words LSD of the second is continuously increased (after every one second) by 1 to reach to 0A H. When it reaches to 0A H then program will jump to MSD (Most Significant Digit) of the seconds after storing 00H in the memory location 2050 H.

Now MSD of the second is moved into accumulator and then increased by 1 which then compared with 06H.

When MSD of the second reaches to 06H, it will make 00H to its corresponding location. The program will then jump to LSD of the minutes. The program will execute in such a way the minutes will be checked after every one minute and it will be updated in the address field so that it displays up to 59H. After that, it becomes 00 H.

In the similar fashion, Hrs. will be checked after every hour so that it displays the updated current time.

To switch ON the electrical appliance at the predetermined time the subroutine program was developed. The subroutine program is executed after every second to check if the current time is equal to the time when the electrical appliance is to switched ON. When the current time is equal to the ON time (required time), 8255A sends a high signal to PA_0 (D_0 bit of Port A). The high signal energies the relay and appliance becomes ON. After switching ON the appliance the program will, however, continue displaying the updated time.

The observations were made to check the ON time and found to work fine. The working of the real time clock was also very accurate. It was proved to be satisfactory assembly language programming of the 8085A.

Further, to change the real time clock to display the timing in address and data field in 24 Hrs., i.e. after 12:59:59 it should display 13:00:00 onwards, rather than 01:00:00 onwards. So certain changes in the main program should be incorporated. The instructions marked by stars (*****, ******, *******) be replaced as given below:

***** Replace the instruction CPI 03 H by CPI 05 H

****** Replace the instruction CPI 01 H by CPI 02 H

******* Replace the instruction MVI A, 01 H by MVI A, 00 H.

15.2 MICROPROCESSOR BASED LED DIAL CLOCK

Here the design of microprocessor based LED dial clock has been discussed which runs using microprocessor 8085. It works purely on the basis of the software; and some hardware has also been used. The software is prepared in the assembly language of 8085A microprocessor. This program was tested on Vinytics 8085 µp kit and found to work very fine.

In this dial clock 73 LEDs are used which are interfaced with the different ports of two 8255 PPI ICs available on µp kit. Out of 73 LEDs, 12 LEDs of Amber color are used to show hours, 60 LEDs of red color are used to show minutes and one green LED alternately glows for half second showing that the seconds are continuously being counted by the clock. All these LEDs are connected in a circle in the form of a dial of a wall clock. The beauty of this project is the software part; and the students of Electronics discipline would definitely like to design this project or this idea may be used in other

projects also. It is worthwhile to mention that no electronic circuit was used, except the LEDs are connected to the two PPI-8255 ICs available with the microprocessor kit.

Figures 15.3 and 15.4 show the circuit diagrams to be interfaced with the μp-Kit. The cathodes of all the 12 Amber colored Hours LEDs are connected to the ground. The Anodes of these LEDs are connected to the Port A and Port B of 8255 –II, the anodes of LEDs showing Hrs 1 to 8 are connected respectively to D0 –D7 Bits of Port A of 8255-II and the anodes of LEDs showing Hours 9 to 12 are connected respectively to D0 – D3 bits of Port B of 8255 – II. Look – up table 15.1 provides Logic 1 to Hours LEDs 1-8. However, logic 1 to Hours LEDs 9 – 12 are provided using the software. The bit D0 of Port C lower of 8255 – II is connected to the anode of Green LED (showing second) and cathode of this LED is also grounded.

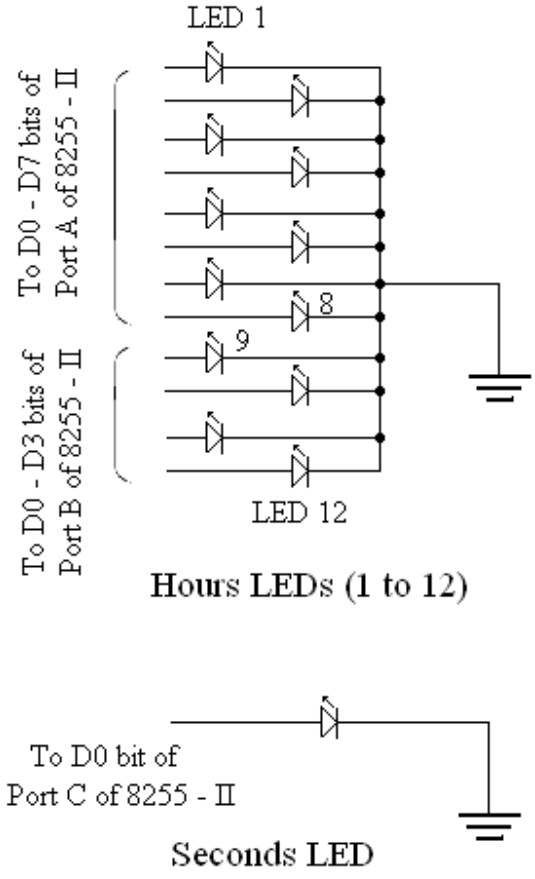


Fig. 15.3

The minutes red colored 60 LEDs are divided into 6 Groups (Group 0 to Group 5) and each group contains 10 LEDs. The cathodes of each group are connected together. The D0 – D5 bits of Port B of 8255 – I are connected to the cathodes (Common Points) of Group 0 to Group 5 LEDs respectively. The ground signal is provided to the cathode pin of each Group through the software using Look up table 15.2.

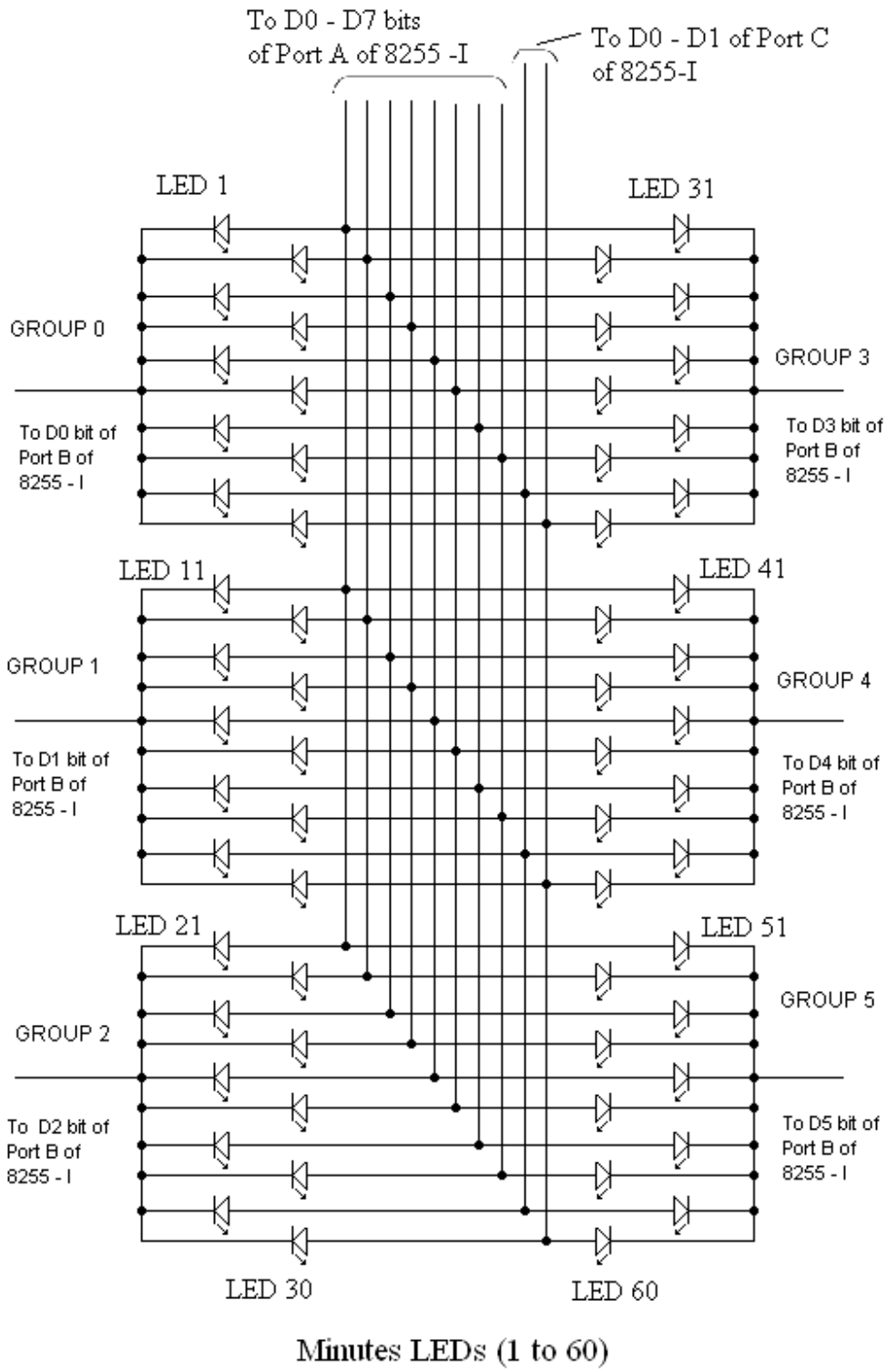


Fig. 15.4

The anodes of 10 LEDs (1 to 10) of each of 6 Groups are connected in parallel, i.e. LED No. 1 of each Group is connected together, and similarly LED No 2 of each

Group is connected together and so on. In this way 10 different Anode leads (1 to 10) are finally available to be connected to different pins of Port A and Port C of 8255 – I. D0 – D7 bits of Port A of 8255 – I are connected to 1 – 8 Anodes leads and 9 – 10 Anode leads are connected to D0 – D1 bits of Port C of 8255– I respectively. Look up table III provides Logic 1 to 1 – 8 Anode Leads; and to the 9 – 10 Anode leads the Logic 1, is provided by the software.

Table –15.1

Memory Location	Data Stores	Binary Equivalent	Comments (Port A of 8255-II)
2101 H	01H	0000 0001	Provides Logic 1 to D0 bit (Hrs.1)
2102 H	02H	0000 0010	Provides Logic 1 to D1 bit (Hrs.2)
2103 H	04H	0000 0100	Provides Logic 1 to D2 bit (Hrs.3)
2104 H	08H	0000 1000	Provides Logic 1 to D3 bit (Hrs.4)
2105 H	10H	0001 0000	Provides Logic 1 to D4 bit (Hrs.5)
2106 H	20H	0010 0000	Provides Logic 1 to D5 bit (Hrs.6)
2107 H	40H	0100 0000	Provides Logic 1 to D6 bit (Hrs.7)
2108 H	80H	1000 0000	Provides Logic 1 to D7 bit (Hrs.8)

Table – 15.2

Memory Location	Data Stores	Binary Equivalent	Comments (Port B of 8255 - I)
2200 H	FEH	1111 1110	Provides Logic 0 to D0 bit (Group 0)
2201 H	FDH	1111 1101	Provides Logic 0 to D1 bit (Group 1)
2202 H	FBH	1111 1011	Provides Logic 0 to D2 bit (Group 2)
2203 H	F7H	1111 0111	Provides Logic 0 to D3 bit (Group 3)
2204 H	EFH	1110 1111	Provides Logic 0 to D4 bit (Group 4)
2205 H	DFH	1101 1111	Provides Logic 0 to D5 bit (Group 5)

Table – 15.3

Memory Location	Data Stores	Binary Equivalent	Comments (Port A of 8255 – I)
2300 H	01H	00000001	Provides Logic 1 to D0 bit
2301 H	02H	00000010	Provides Logic 1 to D1 bit
2302 H	04H	00000100	Provides Logic 1 to D2 bit
2303 H	08H	00001000	Provides Logic 1 to D3 bit
2304 H	10H	00010000	Provides Logic 1 to D4 bit
2305 H	20H	00100000	Provides Logic 1 to D5 bit
2306 H	40H	01000000	Provides Logic 1 to D6 bit
2307 H	80H	10000000	Provides Logic 1 to D7 bit

The software provides a signal continuously to Green LED so that it remains ON and OFF alternately for half a second. The minutes 60 LEDs glow one by one after every one minute. Port B of 8255 – I provides a low voltage (0 volt) to the cathodes of one block for 10 minutes and to the anode of these LEDs (1 to 10) are provided positive voltage (logic 1) alternately for one minute through D0 – D7 bits of Port A and D0 – D2 bits of Port B of 8255 – I. However, Hours' 12 Amber colored LEDs glow one by one after every an Hour time, as positive voltage (logic 1) is provided to the anodes of these LEDs after every an Hour. As discussed earlier positive voltage to Hours LEDs (1 to 8) are provided by D0 – D7 bits of Port A of 8255 – II and to Hours LEDs (9 to 12) are provided by D0 –D3 bits of Port B of 8255 – II after every an hour time. All the LEDs glow in a sequence as the minutes and hours hands move in an analog clock.

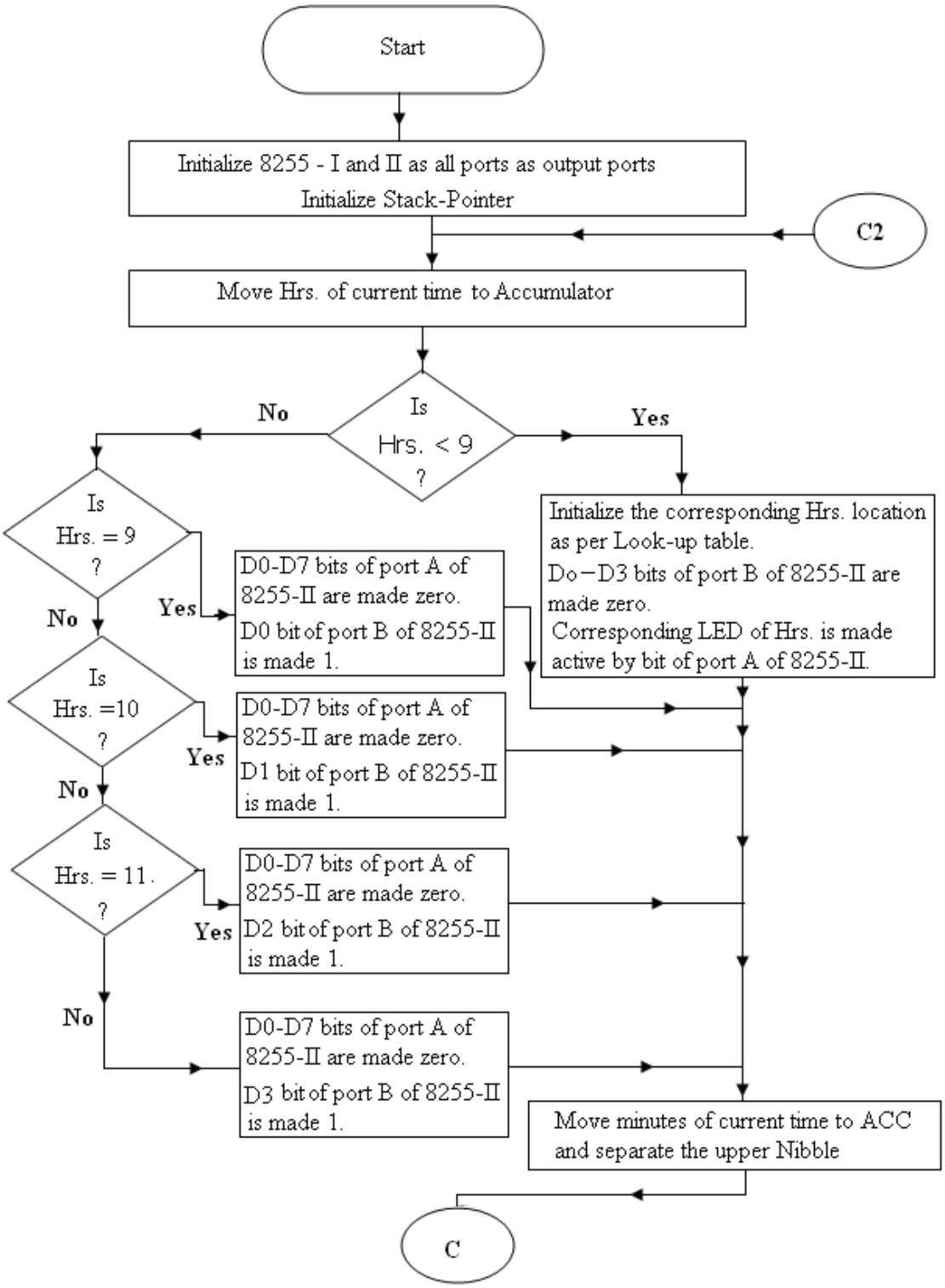
The current time is stored in the beginning in some memory locations. Hours, minutes and seconds of the current time are stored in the memory locations say 2000 H to 2002 H respectively. Figures 15.5 show the flow chart of the program. The software in the assembly language of 8085A for providing appropriate signal to different LEDs is given below:

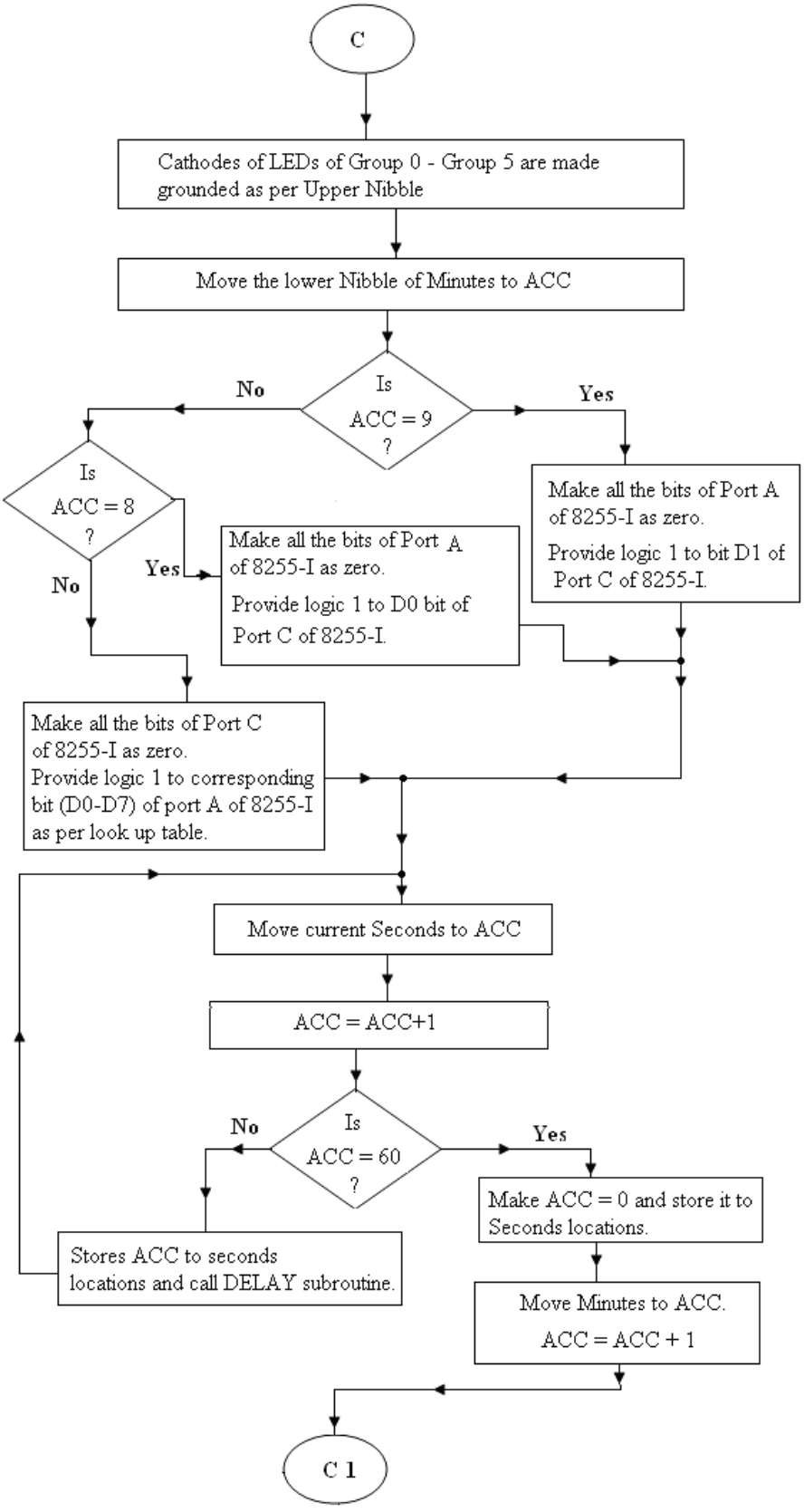
Main Program

MEMORY LOCATIONS WHERE CURRENT TIME IS STORED

2000H	-	CURRENT HRS
2001H	-	CURRENT MINUTES
2002H	-	CURRENT SECONDS

Label	Mnemonics	Operand	Comments
	LXI SP,	XXXX H	;Initialize Stack Pointer.
	MVI A,	80H	;Control word for 8255-I.
	OUT	03H	;Works all the ports of 8255-I as output port
	OUT	0B H	;Works all the ports of 8255-II as output port.
START	LXI H,	2000 H	;Loads the H-L register pair with the address of Hrs. of current time.
	MOV A,	M	;Moves the Hrs. of current time to accumulator.
	CPI	09 H	;It is compared with 09 H.


Start
Initialize 8255 - I and II as all ports as output ports
Initialize Stack-Pointer
C2
Move Hrs. of current time to Accumulator
Is Hrs. < 9 ?
No
Yes
Initialize the corresponding Hrs. location as per Look-up table.
Do−D3 bits of port B of 8255-II are made zero.
Corresponding LED of Hrs. is made active by bit of port A of 8255-II.
Is Hrs. = 9 ?
Yes
D0-D7 bits of port A of 8255-II are made zero.
D0 bit of port B of 8255-II is made 1.
No
Is Hrs. =10 ?
Yes
D0-D7 bits of port A of 8255-II are made zero.
D1 bit of port B of 8255-II is made 1.
No
Is Hrs. = 11. ?
Yes
D0-D7 bits of port A of 8255-II are made zero.
D2 bit of port B of 8255-II is made 1.
No
D0-D7 bits of port A of 8255-II are made zero.
D3 bit of port B of 8255-II is made 1.
Move minutes of current time to ACC and separate the upper Nibble
C


C
Cathodes of LEDs of Group 0 - Group 5 are made grounded as per Upper Nibble
Move the lower Nibble of Minutes to ACC
Is ACC = 9 ?
No
Yes
Is ACC = 8 ?
Yes
No
Make all the bits of Port A of 8255-I as zero.
Provide logic 1 to bit D1 of Port C of 8255-I.
Make all the bits of Port A of 8255-I as zero.
Provide logic 1 to D0 bit of Port C of 8255-I.
Make all the bits of Port C of 8255-I as zero.
Provide logic 1 to corresponding bit (D0-D7) of port A of 8255-I as per look up table.
Move current Seconds to ACC
ACC = ACC+1
Is ACC = 60 ?
No
Yes
Stores ACC to seconds locations and call DELAY subroutine.
Make ACC = 0 and store it to Seconds locations.
Move Minutes to ACC.
ACC = ACC + 1
C 1

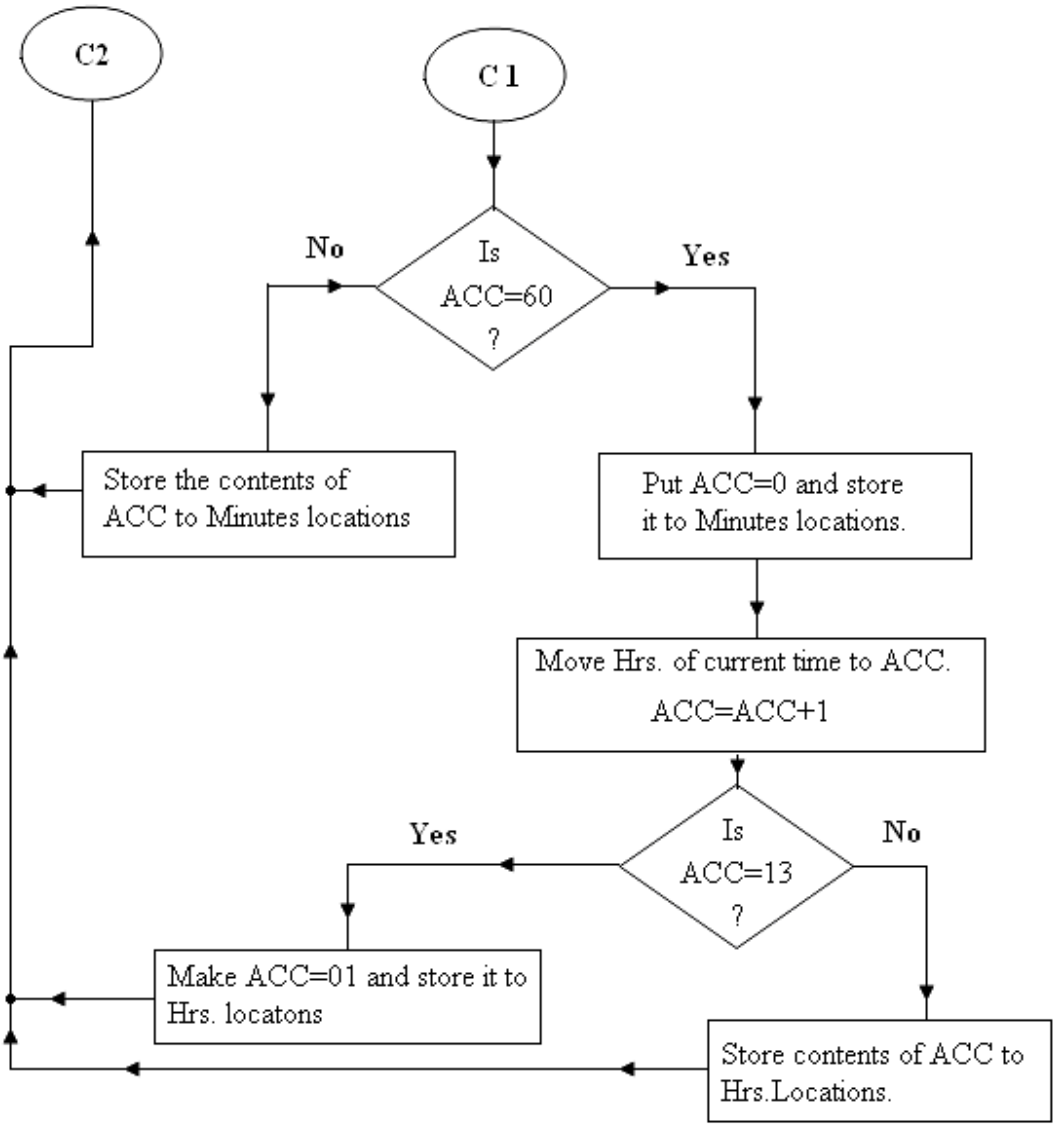


Fig. 15.5

JC	PT0	;If ACC<9, then jump to PT0.
JZ	PT1	;If ACC=9, then jump to PT1.
CPI	10 H	;Accumulator contents are again compared with 10 H.
JZ	PT2	;If ACC=10 then jump to PT2.
CPI	11 H	;If ACC=11 then jump to PT3.
MVI A,	00 H	

```
        OUT     08 H    ;Provides logic 0 to all the
                         bits of Port A of 8255-II.
        MVI A,  08 H
        OUT     09 H    ;Provides logic 1 to D3 bit
                         of port B of 8255-II.
        JMP     PT5     ;Jumps to PT5.
PT1     MVI A,  00 H
        OUT     08 H    ;Provides logic 0 to all the
                         bits of port A of 8255-II.
        MVI A,  01 H
        OUT     09 H    ;Provides logic 1 to D0 bit
                         of port B of 825-II.
        JUMP    PT5     ;Jumps to PT5.
PT2     MVI A,  00 H
        OUT     08 H    ;Provides logic 0 all the bits
                         of port A of 8255-II.
        MVI A,  02 H
        OUT     09 H    ;Provides logic 1 to D1 bit
                         of Port B of 8255-II.
        JMP     PT5     ;Jumps to PT5.
PT3     MVI A,  00 H
        OUT     08 H    ;Provides logic 0 all the bits
                         of port A of 8255-II.
        MVI A,  04 H
        OUT     09 H    ;Provides logic 1 to D2 bit
                         of port B of 8255-II.
        JMP     PT5     ;Jumps to PT5.
PT0     MOV E,  M       ;Hrs. of current time is less
                         than 9 which is loaded to E
                         register.
        MVI A,  00 H
        OUT     09 H    ;Provides logic 0 to all the
                         bits of port B of 8255-II.
        MVI D,  21 H
        LDAX  D         ;Loads the Acc with the data
                         as per the table 15.1in
                         respect of the current time,
                         which is less than 9.
        OUT     08 H    ;Provides logic w to one of
                         the bits of port A of 8255-II
                         as per the look up table
                         15.1.
PT5     INX H           ;H-L register pair will
                         indicate the minutes of the
                         current time.
```

```
        MOV A,      M       ;Moves current minutes to
                            ACC.
        ANI         F0 H    ;Make lower Nibble of ACC
                            to zero.
        MVI B,      04 H
LOOP    RLC                 ;Rotate ACC contents four
                            times to shift higher Nibble
                            of ACC to lower Nibble
                            and lower which are zero is
                            shifted to higher Nibble.
        DCR         B
        JNZ         LOOP
        MOV E,      A       ;ACC contents are loaded to
                            E register.
        INR         D
        LDAX D
        OUT         01 H    ;Provides logic 0 to the
                            cathodes of one group from
                            group 0 to group 5 as per
                            the look up table 15.2.
        MOV A,      M
        ANI         0F H    ;Provides logic 0 to higher
                            Nibble of ACC.
        CPI         09 H    ;Compared with 09.
        JZ          AK      ;If ACC=09 then jump to
                            AK.
        CPI         08 H    ;Compared with 08.
        JZ          AK1     ;If ACC=08 then jump to
                            AK1.
        MVI A.      00 H
        OUT         02 H    ;Provides logic 0 to all the
                            bits of Port C of 8255-I.
        MOV E,      A
        INR         D
        LDAX D              ;Provides logic 1 to one of
                            the bits of D0-D7 of port A
                            of 8255-I as per the look-up
                            table 15.3.
        OUT         00 H
        JMP         AK2     ;Jump to AK2.
AK1     MVI A,      00 H
        OUT         00 H    ;Provides logic 0 to all the
                            bits of Port A  of 8255-I.
        MVI A,      01 H
        OUT         02 H    ;Provides logic 1 to D0 bit
                            of port C of 8255-I.
```

```
        JMP     AK2     ;Jump to AK2.
AK      MVI A,  00 H
        OUT     00 H    ;Provides logic 0 to all the
                         bits of Port A  of 8255-I.
        MVI A,  02 H
        OUT     02 H    ;Provides logic 1 to D1 bit
                         of port C of 8255-I.
AK2     INX H
PT7     MOV A,  M       ;Moves current seconds to
                         ACC.
        ADI     01 H    ;ACC=ACC+1
        DAA             ;Decimal adjust the
                         Accumulator.
        CPI     60 H    ;Is ACC=60.
        JZ      PT6     ;If yes jump to PT6.
        MOV M,  A       ;Else stores the current
                         seconds to its
                         corresponding locations.
        CALL    DELAY   ;Calls subroutine program
                         for one second dealay.
        JMP     PT7     ;Jumps to PT7.
PT6     MVI A,  00 H
        MOV M,  A       ;Stores 00 to Memory
                         locations of current
                         seconds.
        DCX H
        MOV A,  M       ;Minutes of the current time
                         is loaded to the
                         accumulator.
        ADI     01 H    ;ACC=ACC+1
        DAA             ;Decimal adjust the
                         Accumulator.
        CPI     60 H    ;Is ACC=60.
        JZ      PT8     ;If yes jump to PT8.
        MOV M,  A       ;Stores minutes to the
                         Minutes locations.
        JMP     START   ;Jump to START to glow
                         minutes and Hours.
PT8     MVI A,  00 H
        MOV M,  A       ;Stores 00 to Memory
                         locations of current
                         minutes.
        DCX H
        MOV A,  M       ;Hrs. of the current time is
                         loaded to the accumulator.
        ADI     01 H    ;ACC=ACC+1
```

```
        DAA                         ;Decimal adjust the
                                     Accumulator.
        CPI     13 H                ;Is ACC=13.
        JZ      PT9                 ;If yes jump to PT9.
        MOV M,  A                   ;Stores Hrs. to the Hrs.
                                     locations.
        JMP     START               ;Jump to START to glow
                                     minutes and Hours.
PT9     MVI A,  01 H
        MOV M,  A                   ;Stores Hrs.=01 in Hrs,
                                     locations instead of 13.
        JMP     START               ;Jump to START to glow
                                     minutes and Hours.
```

Delay Subroutine Program

Label	Mnemonics	Operand	Comments
DELAY	MVI A,	01 H	
	OUT	0A H	;Seconds LED glows for half second.
	LXI D,	FA00 H	
NXT1	DCX	D	;Decrement D-E register pair.
	MOV A,	D	;Moves the contents stored in M_{D-E} to Acc.
	ORA	E	;The contents of A and E registers are ORed bit by bit.
	JNZ	NXT1	;If the result is not zero then jump to nxt1 else next instruction.
	MVI A,	00 H	
	OUT	0A H	;Seconds LED remains off for half second.
	LXI D,	FA00 H	
NXT2	DCX	D	;Decrement D-E register pair.
	MOV A,	D	;Moves the contents stored in M_{D-E} to Acc.

```
        ORA     E       ;The contents of A and E
                          registers are ORed bit by
                          bit.
        JNZ     NXT2    ;If the result is not zero then
                          jump to nxt1 else next
                          instruction.
        RET             ;Return to main program.
```

15.3 DESIGN OF MICROPROCESSOR BASED RUNNING LIGHT

The details discussed in the foregoing section of this chapter, may also be used to design the running light. In this project it is expected that 60 lEDs should glow in a sequence one after another in a preset time say one second. The LEDs are connected as shown in figure 15.6. The logic used in this project is the same as in discussed in the microprocessor based dial clock. The assembly language program is self explanatory which is given below. The look-up table for the same is given in table 15.4 and 15.5.

```
Label   Mnemonics   Operand     Comments
        MVI A,      80H         ;Control word for 8255-I.
        OUT         03H         ;Works all the ports of 8255-
                                  I as output port
        MVI A,      00 H        ;00 is loaded to accumulator.
        STA         2054 H      ;Store 00 to some location
                                  say 2054 H
        STA         2055 H      and 2055 H
START   LXI H,      2054 H      ;Loads the H-L register pair
                                  with the address 2054 H.
        MOV E,      M           ;Data of 2054 is loaded to
                                  the E register.
        MVI D,      21 H
        LDAX  D                 ;Data of one memory
                                  location 2100 H to 2105 H
                                  is loaded to accumulator.
        OUT         01 H        ;Cathode of one block (1 –
                                  5) will be low.
        INX H                   ;H-L register pair will
                                  incremented.
        MOV A,      M           ;Moves the data in next
                                  location will be loaded to
                                  accumulator.
        CPI         09 H        ;Compared with 09.
        JZ          AK          ;If ACC=09 then jump to
                                  AK.
        CPI         08 H        ;Compared with 08.
```

```
        JZ          AK1         ;If ACC=08 then jump to
                                 AK1.
        MVI A.      00 H
        OUT         02 H        ;Provides logic 0 to all the
                                 bits of Port C of 8255-I.
        MOV E,      M
        INR         D
        LDAX  D                 ;Provides logic 1 to one of
                                 the bits of D0-D7 of port A
                                 of 8255-I as per the look-up
                                 table.
        OUT         00 H
        JMP         AK2         ;Jump to AK2.
AK1     MVI A,      00 H
        OUT         00 H        ;Provides logic 0 to all the
                                 bits of Port A  of 8255-I.
        MVI A,      02 H
        OUT         02 H        ;Provides logic 1 to D1 bit
                                 of port C of 8255-I.
        JMP         AK2         ;Jump to AK2.
AK      MVI A,      00 H
        OUT         00 H        ;Provides logic 0 to all the
                                 bits of Port A  of 8255-I.
        MVI A,      01 H
        OUT         02 H        ;Provides logic 1 to D0 bit
                                 of port C of 8255-I.
AK2     MOV A,      M           ;Moves current value to
                                 ACC.
        ADI         01 H        ;ACC=ACC+1
        CPI         0A H        ;Is ACC=0A H.
        JZ          RR          ;If yes jump to RR.
        MOV M,      A           ;Else stores the current
                                 seconds to its
                                 corresponding locations.
PP      MVI B,      02 H        ;02 is loaded to B register
                                 for the loop.
YY      LXI D,      FA00 H
NXT1    DCX         D           ;Decrement D-E register
                                 pair.
        MOV  A,     D           ;Moves the contents stored
                                 in M_D-E to Acc.
        ORA         E           ;The contents of A and E
                                 registers are ORed bit by
                                 bit.
```

```
        JNZ     NXT1    ;If the result is not zero then
                         jump to nxt1 else next
                         instruction.
        DCR     B       :Decrement B reguster
        JNZ     YY
        JMP     START   ;Jumps to START.
RR      MVI A,  00 H
        MOV M,  A       ;Stores 00 to current
                         Memory locations.
        DCX H
        MOV A,  M       ;Current value is loaded to
                         the accumulator.
        ADI     01 H    ;ACC=ACC+1
        CPI     06 H    ;Is ACC=06.
        JZ      UU      ;If yes jump to UU.
        MOV M,  A       ;Stores in memory location.
        JMP     PP      ;Jump to PP for delay.
UU      MVI A,  00 H
        MOV M,  A       ;Stores 00 to Memory
                         locations of current.
        JMP     PP      ;Jump to PP for delay.
```

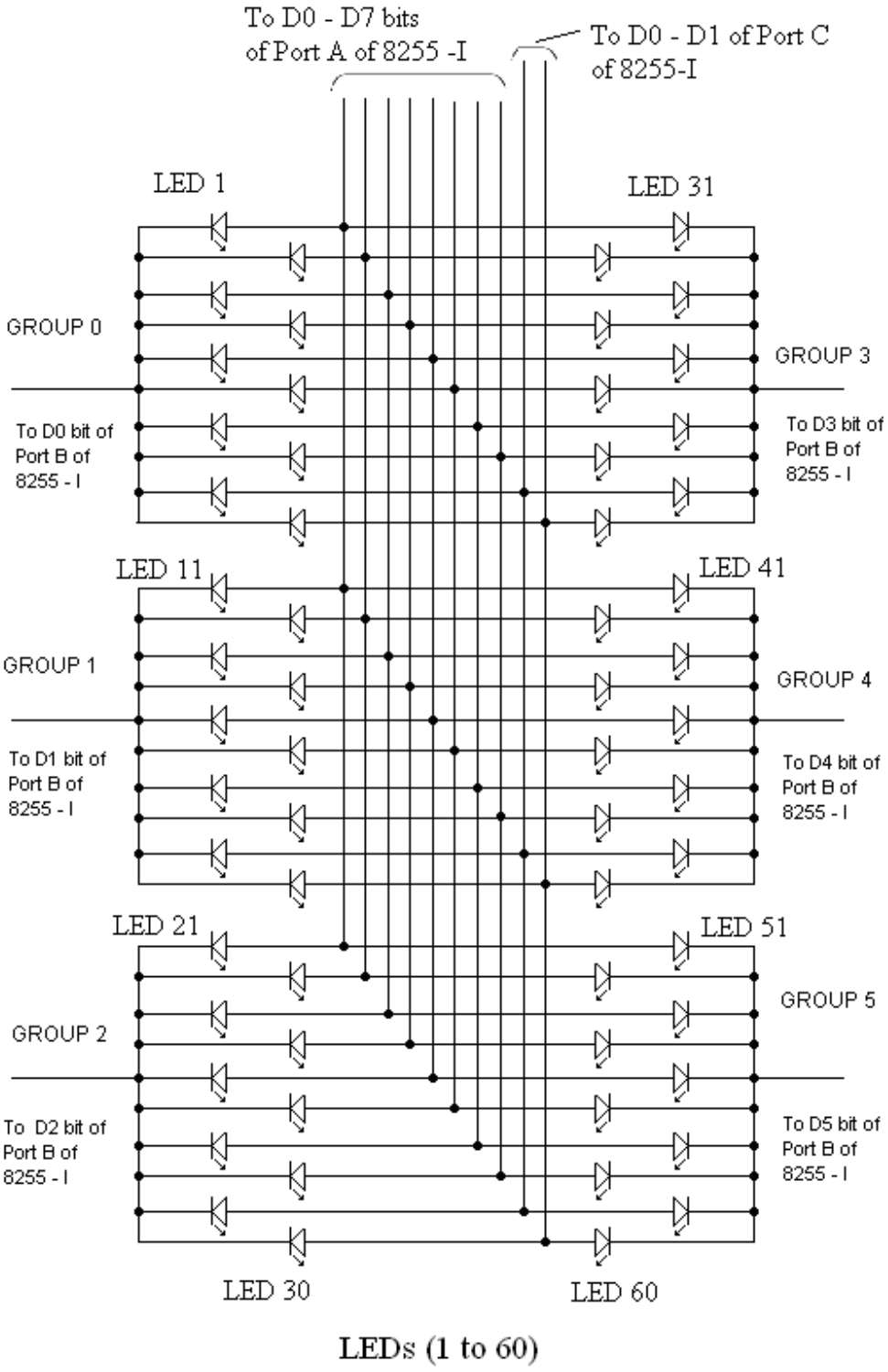

To D0 - D7 bits
of Port A of 8255 -I
To D0 - D1 of Port C
of 8255-I
LED 1
LED 31
GROUP 0
GROUP 3
To D0 bit of
Port B of
8255 - I
To D3 bit of
Port B of
8255 - I
LED 11
LED 41
GROUP 1
GROUP 4
To D1 bit of
Port B of
8255 - I
To D4 bit of
Port B of
8255 - I
LED 21
LED 51
GROUP 2
GROUP 5
To D2 bit of
Port B of
8255 - I
To D5 bit of
Port B of
8255 - I
LED 30
LED 60
LEDs (1 to 60)

Fig. 15.6

Table – 15.4

Memory Location	Data Stores	Binary Equivalent	Comments (Port B of 8255 - I)
2100 H	FEH	1111 1110	Provides Logic 0 to D0 bit (Group 0)
2101 H	FDH	1111 1101	Provides Logic 0 to D1 bit (Group 1)
2102 H	FBH	1111 1011	Provides Logic 0 to D2 bit (Group 2)
2103 H	F7H	1111 0111	Provides Logic 0 to D3 bit (Group 3)
2104 H	EFH	1110 1111	Provides Logic 0 to D4 bit (Group 4)
2105 H	DFH	1101 1111	Provides Logic 0 to D5 bit (Group 5)

Table – 15.5

Memory Location	Data Stores	Binary Equivalent	Comments (Port A of 8255 – I)
2200 H	01H	00000001	Provides Logic 1 to D0 bit
2201 H	02H	00000010	Provides Logic 1 to D1 bit
2202 H	04H	00000100	Provides Logic 1 to D2 bit
2203 H	08H	00001000	Provides Logic 1 to D3 bit
2204 H	10H	00010000	Provides Logic 1 to D4 bit
2205 H	20H	00100000	Provides Logic 1 to D5 bit
2206 H	40H	01000000	Provides Logic 1 to D6 bit
2207 H	80H	10000000	Provides Logic 1 to D7 bit

15.4 MICROPROCESSOR BASED AUTOMATIC SCHOOL BELL SYSTEM

This is an effective and useful project for educational institutes. In most of educational institutes, the peon rings the bell after every period (usually a period is of 40 minutes duration). The peon has to depend on his wrist watch or clock and sometimes he would even forget to ring the bell in time. In the present system, the human error has been eliminated. Every morning, when the school starts, someone has just to switch on the system and it will thereafter work automatically.

The automatic microprocessor based school bell system presented here has been tested on a Vinytics' microprocessor -8085 kit (VMC-8506). The kit displays the period number on two most significant digits of address field and minutes of the period elapsed on the next two digits of the address field. The data field of the kit displays seconds continuously.

The idea used here is very simple. The programmable peripheral interfacing (PPI) IC INTEL 8255-I available with the kit has been used. The bit 0 of port A (PA0) is connected to the base of a transistor through a resistance as shown in figure 15.7. It is used to energize the relay when PA0 pin of 8255-I is high. The siren, hooter or any bell voice system with an audio amplifier of proper wattage (along with 2 to 3 loudspeakers)

may be installed in the school campus. The relay would get energized after every 40 minutes for a few seconds (say 6 Seconds).

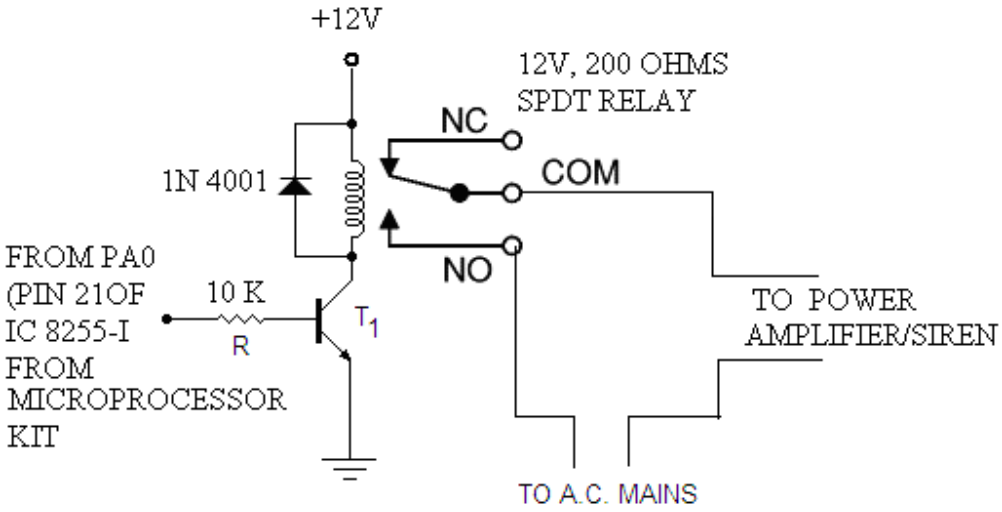


Fig. 15.7

The assembly language program and data used for the purpose are given below in the mnemonic and machine code. The program is self explanatory.

Address	Opcode	Label	Mnemonic & Operand	Comments
20FC H	3E 80		MVI A, 80 H	;Initialize 8255-1 as output port.
20FE H	DE 03		OUT 03 H	
2100 H	31 FF 27		LXI SP, 27FF H	;Initialize the stack pointer.
2103 H	CD 47 03		CALL 0347 H	;Clear the display.
2106 H	C3 69 21		JMP TT	;Jump to ring the bell.
2109 H	AF	AA	XRA A	;Put A = 0.
210A H	47		MOV B, A	;Put B=0.
210B H	21 50 05		LXI H, 2050 H	;Initialize address of the display.
210E H	CD D0 05		CALL 05D0 H	;To display period no. and minutes to address field.
2111 H	3E 01		MVI A, 01 H	;Put A = 01.
2113 H	06 00		MVI B, 00 H	;Put A = 00.
2115 H	21 54 20		LXI H, 2054 H	;Initialize seconds locations.
2118 H	CD D0 05		CALL 05 D0 H	;Displays seconds in data field.

211B H	21 55 20		LXI H, 2055 H	;Initialize address of LSD of current seconds.
211E H	7E		MOV A, M	;Moves LSD of current seconds to accurmulator.
211F H	C6 01		ADI 01 H	;Add 1 to accumulator
2121 H	FE 0A		CPI 0A H	;Compares LSD of seconds with 0A (decimal no. 10).
2123 H	CA 36 21		JZ RR	;If LSD of seconds completes 09 then jump to RR.
2126 H	77		MOV M, A	;Moves the ACC contents to 2055 H location.
2127 H	06 02	DD	MVI B, 02 H	; Delay program
2129 H	11 00 FA	YY	LXI D, FA 00 H	for
212C H	CD 00 25		CALL 2500 H	one
212F H	05		DCR B	second.
2130 H	C2 29 21		JNZ YY	
2133 H	C3 09 21		JMP AA	;Jump to AA for display the time.
2136 H	3E 00	RR	MVI A, 00 H	;Put A=00.
2138 H	77		MOV M, A	;Store ACC to memory location.
2139 H	2B		DCX H	;Decrement the H-L register pair content.
213A H	7E		MOVE A, M	;Moves the MSD of second to ACC.
213B H	C6 01		ADI 01 H	;Add 01 to ACC.
213D H	FE 06		CPI 06 H	;Compares MSD of seconds with 06H.
213F H	CA 46 21		JZ UU	;If seconds completes 59 jump to UU.
2142 H	77		MOV M, A	;Moves Acc. contents to memory location.
2143 H	C3 27 21		JMP DD	;Jumps for delay of 1 second.
2146 H	3E 00	UU	MVI A, 00 H	;Put A=0 after completing 59 Sec.
2148 H	77		MOV M, A	
2149 H	2B		DCX H	
214A H	7E		MOV A, M	;Moves LSD of current minutes to accumulator.

214B H	C6 01		ADI 01 H	;Adds 01 to ACC.
214D H	FE 0A		CPI 0A	;Compares ACC to 0A.
214F H	CA 56 21		JZ VV	;Jumps to VV if LSD of minutes completes 09.
2152 H	77		MOV M, A	;Moves ACC contents to memory location.
2153 H	C3 27 21		JMP DD	;Jumps for delay of 1 second.
2156 H	3E 00	VV	MVI A, 00 H	
2158 H	77		MOV M, A	
2159 H	2B		DCX H	;Decrement the contents of H-L register pair.
215A H	7E		MOV A, M	;Moves MSD of minutes to ACC.
215B H	C6 01		ADI 01 H	;Add 1 to it.
215D H	FE 04		CPI 04 H	;Compares ACC contents with 04 H.
215F H	CA 66 21		JZ SS	;If minutes 40 then jumps to SS.
2162 H	77		MOV M, A	
2163 H	C3 27 21		JMP DD	;Jumps for delay of 1 second.
2166 H	3E 04	SS	MVI A, 04 H	;Put A=4.
2168 H	77		MOV M, A	
2169 H	AF	TT	XRA A	;Put A=0.
216A H	47		MOV B, A	;Put b=0.
216B H	21 50 20		LXI H, 2050 H	
216E H	CD D0 05		CALL 05 D0 H	;Displays the period No. and minutes in address field.
2171 H	3E 01		MVI A, 01 H	;Put A=01.
2173 H	06 00		MVI B, 00 H	;Put B=00.
2175 H	21 54 20		LXI H, 2054 H	
2178 H	CD D0 05		CALL 05 D0 H	;Displays the seconds in data field.
217B H	3E 01		MVI A, 01 H	
217D H	D3 00		OUT 00 H	;Excite 8255-I to energize the relay (rings the bell).
217F H	21 55 20		LXI H, 2055 H	

```
2182 H    3E  00              MVI A, 00 H     ;  Puts 00 to
2184 H    77                  MOV M, A          2055 H
2185 H    2B                  DCX H
2186 H    77                  MOV M, A          to
2187 H    2B                  DCX H
2188 H    77                  MOV M, A          2052 H
2189 H    2B                  DCX H
218A H    77                  MOV M, A
218B H    2B                  DCX H
218C H    7E                  MOV A, M        ;Brings the LSD of
                                                current period
                                                number to ACC.
218D H    C6  01              ADI 01 H        ;Add 1 to it.
218F H    FE  0A              CPI 0A H        ;Compares with 0A.
2191 H    CA  98  21          JZ XX           ;If LSD of period
                                                number complete 09
                                                then jump to XX.
2194 H    77                  MOV M, A        ;Else stores it to
                                                memory location.
2195 H    C3  A0  21          JMP XY          ;Jump to XY.
2198 H    3E  00        XX    MVI A, 00 H     ;Puts A=00.
219A H    77                  MOV M, A        ;Stores MSD of
                                                period number to
                                                accumulator.
219B H    2B                  DCX H
219C H    7E                  MOV A, M        ;Stores MSD of
                                                period number to
                                                accumulator.
219D H    C6 01               ADI 01 H        ;Adds 1 to it.
219F H    77            XXX   MOV M, A        ;Stores it to memory
                                                location.
21A0 H    06  02        XY    MVI B, 02 H     ; Delay program
21A2 H    11  00  FA    XYZ   LXI D, FA  00 H   for
21A5 H    CD  00  25          CALL 2500 H       one
21A8 H    05                  DCR B             second.
21A9 H    C2  A2  21          JNZ  XYZ
21AC H    AF                  XRA A           ;Puts A=0.
21AD H    47                  MOV B, A        ;Puts B=0 also.
21AE H    21  50  20          LXI H, 2050 H   ;Program
21B1 H    CD  D0  05          CALL 05D0 H       to
21B4 H    3E  01              MVI A, 01 H     display the
21B6 H    06  00              MVI B, 00 H       period number
21B8 H    21  54  20          LXI H, 2054 H     minutes
21BB H    CD  D0  05          CALL 05D0 H       and
21BE H    21  55  20          LXI H, 2055 H     seconds.
```

21C1 H	7E	MOV A, M	;Stores LSD of current seconds to ACC.
21C2 H	C6 01	ADI 01 H	;Adds 1 to it.
21C4 H	FE 06	CPI 06 H	;Compares it with 06.
21C6 H	C2 9F 21	JNZ XXX	;If not 06 then jump to XXX.
21C9 H	3E 00	MVI A, 00 H	;Put A=0 .
21CB H	D3 00	OUT 00 H	;Output to 8255-I to de-energize the relay.
21CD H	C3 09 21	JMP AA `	;Repeat for the next period.

DELAY SUBROUTINE AT MEMORY LOCATION 2500 H

2500 H	1B	NEXT	DCX D
2501 H	7A		MOV A, D
2502 H	B3		ORA E
2503 H	C2 00 25		JNZ NEXT
2506 H	C9		RET

DATA TO BE FILLED BEFORE THE EXECUTION OF THE PROGRAM

2050 H	00	MSD OF PERIOD NO. IS STORED 00.
2051 H	00	LSD OF PERIOD NO. IS STORED 00.
2052 H	00	MSD OF MINUTES IS STORED 00.
2053 H	00	LSD OF MINUTES IS STORED 00.
2054 H	00	MSD OF SECONDS IS STORED 00.
2055 H	00	LSD OF SECONDS IS STORED 00.

First, the program and data are entered in the microprocessor kit. When the program is run a bell sound will be heard for few seconds (say 6 seconds). This will be repeated after every 40 minutes. The period number will be displayed in the two higher digits of the address field. The minutes and seconds of the time elapsed will be displayed in the rest two digits of lower digits of address field and two digits of the data field.

The address and data fields of the microprocessor kit would display the period number and elapsed minutes and seconds in the address and data fields as given below (fig. 15.8). This figure indicate that period No. is 1, minutes and seconds elapsed are 0, 0 respectively.

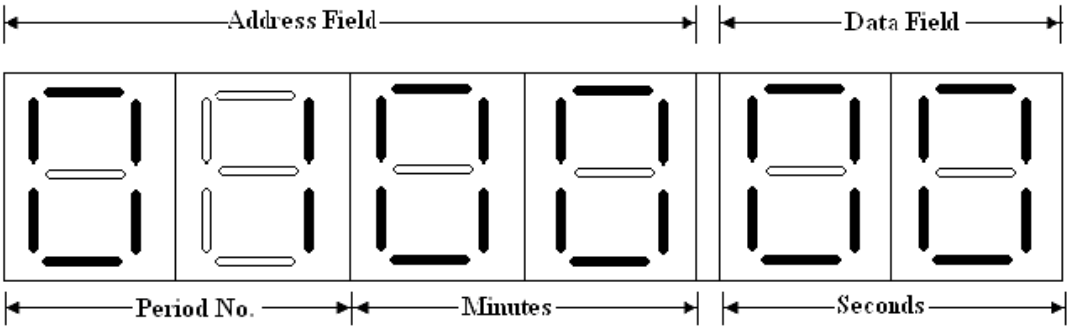


Fig. 15.8

15.5 MICROPROCESSOR BASED TRAFFIC LIGHT

The traffic light controller, which is generally seen at the crossing of the roads, may be designed using the assembly language programming of the microprocessor 8085A. The traffic light controller suggested in this section uses four sets of three LEDs (red, yellow and green) arranged at the four sides of the crossing as shown in figure 15.9. The microprocessor kit (M/S Vinytics) was used to design the traffic light controller.

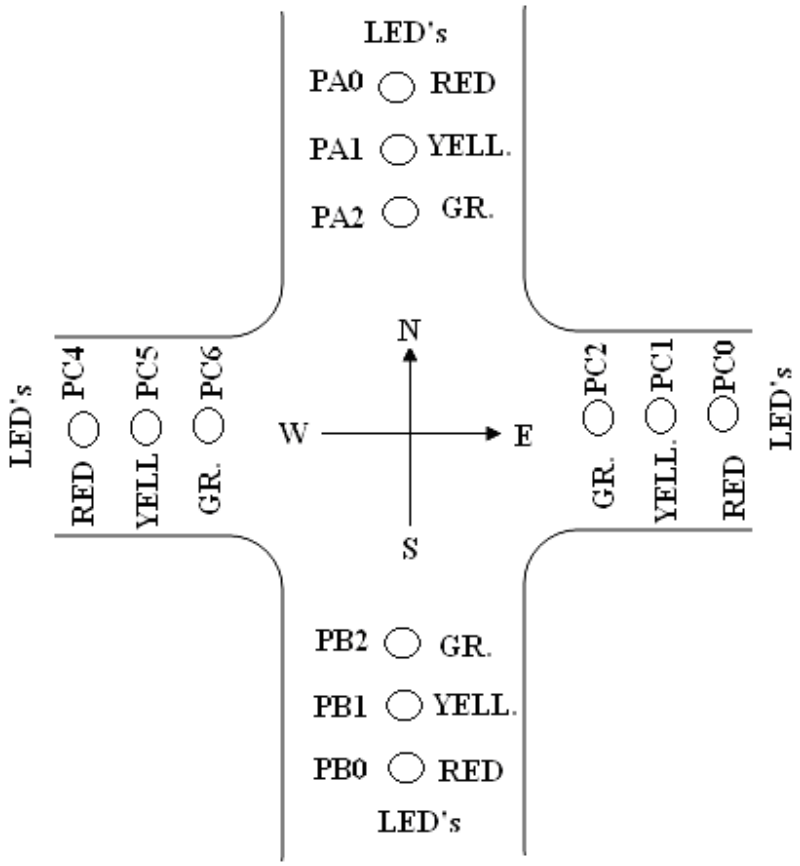


Fig. 15.9

The ports of PPI 8255-I (Port A, Port B, Port C_{Lower} and Port C_{Upper}) available with the kit were used as the output ports. The Red, Yellow and Green LEDs of North are connected to PA_0, PA_1 and PA_2 (bits 0, 1 and 2 of port A) pins of 8255-I respectively. The bits 0, 1, 2 of port B (PB_0, PB_1 and PB_2) were connected to the red, yellow and green LEDs of South respectively. Similarly, PC_0, PC_1 and PC_2 of port C_{Lower} and PC_4, PC_5 and PC_6 of port C_{Upper} were connected to the red, yellow and green LEDs of East and West sides.

Now the following four points occurs for the traffic to control:

(1) First traffic to east west direction is allowed for some time (delay time of 60 seconds say) and traffic to north and south side is not allowed for the sane delay time. This is possible if Red LEDs of north and south (PA_0 and PB_0 both are high) and Green LEDs of East and West (PC_2 and PC_6 are also high) glow.

(2) The yellow lights of east-west and north-south sides are allowed to glow to clear the traffic of these directions for some time (say 20 seconds). The green LEDs of east and west, and red LEDs of north and south are to be remained off for the same time. This condition will be satisfied if PC_1 and PC_5, and PA_1 and PB_1 are high; and PA_0 and PB_0, PC_2 and PC_6 are low.

(3) Now the traffic to north and south side is allowed for the delay time of 60 seconds and traffic to east west direction is not allowed for the sane delay time. This is possible if green LEDs of north and south (PA_2 and PB_2 both are high) and red LEDs of East and West (PC_0 and PC_4 are also high) glow.

(4) Again the yellow lights of east-west and north-south sides are allowed to glow to clear the traffic of these directions for some time (say 20 seconds). The green LEDs of north and south, and red LEDs of east and west are to be remained off for the same time. This condition will be satisfied if PC_1 and PC_5, and PA_1 and PB_1 are high.

The assembly language program for the above conditions is given below, which is self explanatory:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	;Initialize the stack pointer.
	MVI A,	80H	;Control word for 8255-I.
	OUT	03H	;Works all the ports of 8255-I as output ports.
AGAIN	MVI A,	01 H	;01 is loaded to accumulator.
	OUT	00 H	;PA_0 is high (red LED of north glows).
	OUT	01 H	;PB_0 is high (red LED of south glows).

```
MVI A,      44 H      ;PC2 and PC6 are high
                        (green LEDs of east and
                        west glows).
OUT         02 H
CALL        DELAY I   ; Delay program of 60
                        seconds.
MVI A,      22 H
OUT         02 H      ; PC1 and PC5 are high
                        (Yellow LEDs of east and
                        west glow).
MVI A,      02 H
OUT         00 H      ; PA1 is high (yellow LED
                        of north glows).
OUT         01 H      ; PB1 is high (yellow LED
                        of south glows).
CALL        DELAY II  ; Delay program of 20
                        seconds.
MVI A,      11 H
OUT         02 H      ; PC0 and PC4 are high (red
                        LEDs of east and west
                        glow).
MVI A,      04 H
OUT         00 H      ; PA2 is high (green LED of
                        north glows).
OUT         01 H      ; PB2 is high (green LED of
                        south glows).
CALL        DELAY I   ; Delay program of 60
                        seconds.
MVI A,      22 H
OUT         02 H      ; PC1 and PC5 are high
                        (Yellow LEDs of east and
                        west glow).
MVI A,      02 H
OUT         00 H      ; PA1 is high (yellow LED
                        of north glows).
OUT         01 H      ; PB1 is high (yellow LED
                        of south glows).
CALL        DELAY II  ; Delay  program of 20
                        seconds.
JMP         AGAIN     ; Jump to repeat the program
                        from the beginning.
```

Subroutine programs to introduce delay for 6o seconds and 20 seconds are given below:

Subroutine Program for DELAY I

Label	Mnemonics	Operand	Comments
DELAY I	MVI B,	78 H	;78 H (120 decimal number) is loaded to B register for the loop.
YY	LXI D,	FA00 H	
NXT1	DCX	D	;Decrement D-E register pair.
	MOV A,	D	;Moves the contents stored in M_{D-E} to Acc.
	ORA	E	;The contents of A and E registers are ORed bit by bit.
	JNZ		NXT1;If the result is not zero then jump to nxt1 else next instruction.
	DCR	B	:Decrement B register
	JNZ	YY	
	RET		; Returns to main program.

Subroutine Program for DELAY II

Label	Mnemonics	Operand	Comments
DELAY II	MVI B,	28 H	;28 H (40 decimal number) is loaded to B register for the loop.
	JMP	YY	; Jump to YY of DELAY I program.

Figure 15.10 shows the LED connections to 8255-I available with the microprocessor kit.

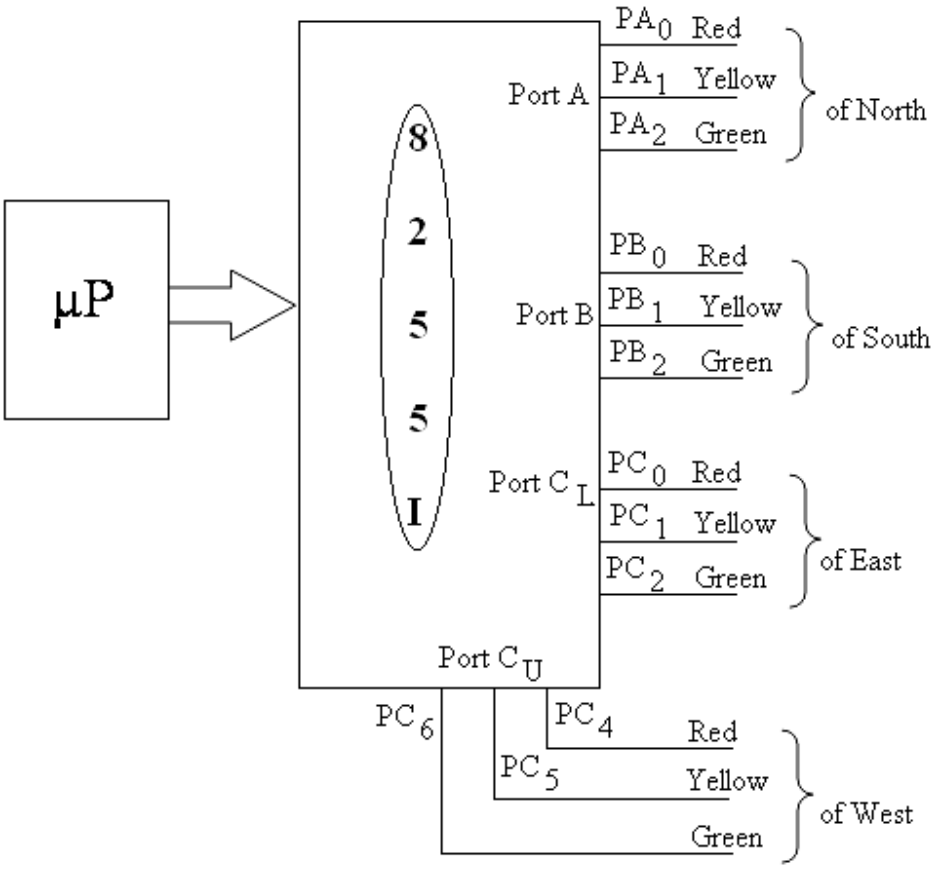


Fig. 15.10

15.5.1 Another Design of Microprocessor Based Traffic Light

Here another design of Traffic light controller is given in which there are more degree of freedom for the traffic. Further, the display of time delay (count down) on the microprocessor kit is also shown. In this design, the divider lines at the four path ways are shown (ref. figure 15.11). At each divider line of four path ways near the crossing, five LEDs (one red, one yellow and three green) are installed on four pillars. These LEDs are connected to the ports of PPI 8255-I available with the microprocessor kit. All the ports of 8255 are used as output ports. The status of LEDs will indicate the direction of the traffic at the crossing.

In this design the delay of 90 seconds are allowed for the normal traffic and say 10 seconds delay is introduced for yellow light to clear the traffic, before the next traffic direction. The delay will be displayed on the data field with down counting. The assembly language programming of the design is given below. The program is simple and self-explanatory.

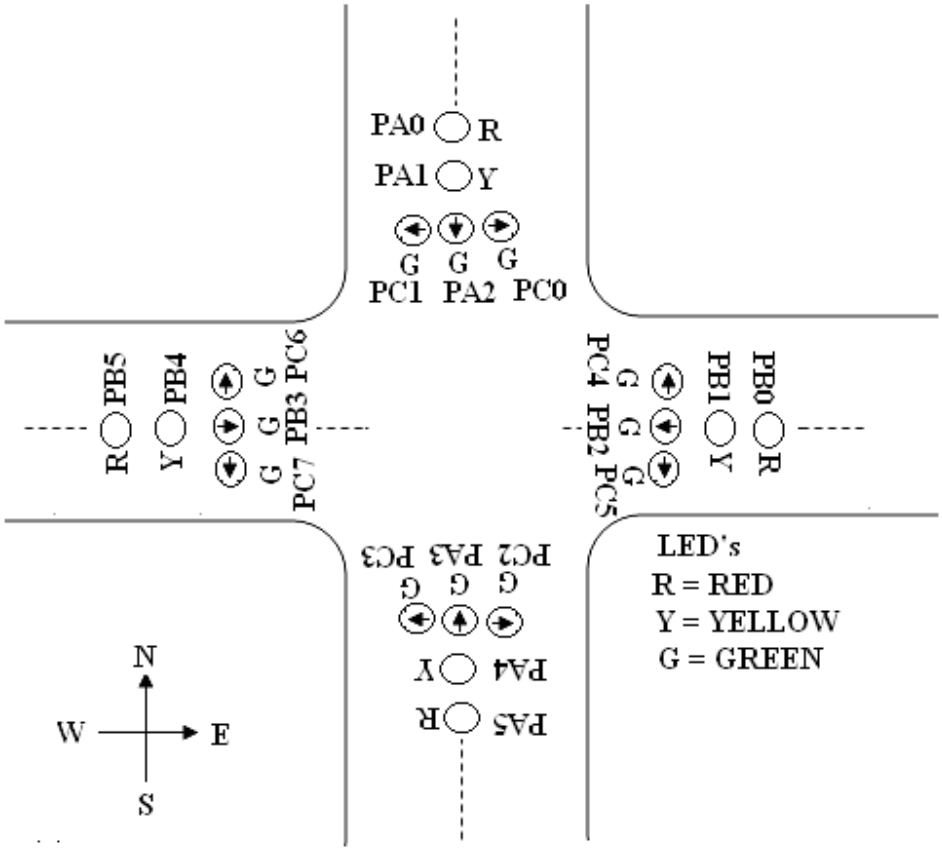


Fig. 15.11

Main Program:

LOOK UP TABLE:

2050 H	**16**	**BLANK**
2051H	**16**	**BLANK**
2052 H	**16**	**BLANK**
2053 H	**16**	**BLANK**

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	;Initialize the stack pointer.
	MVI A,	80H	;Control word for 8255-I.
	OUT	03H	;Works all the ports of 8255-I as output ports.
START	MVI A,	21 H	; 21 H is loaded to accumulator.

```
OUT     00 H    ;PA0 and PA5 are high (red
                 LEDs of north and south
                 glow).
MVI A,  0C H    ; 0C H is loaded to
                 accumulator.
OUT     01 H    ; PB2 and PB3 are high
                 (green LEDs of East and
                 west glow).
MVI A,  60 H    ; 60 H is loaded to acc.
OUT     02 H    ; PC5 and PC6 are high
                 (Traffic from west to north
                 and east to south is
                 allowed).
CALL    DELAY I ; Calls the delay subroutine
                 program (DELAY I).
CALL    DISPLAY ; Calls the Display
                 subroutine program for the
                 display of delay time.
MVI A,  21 H    ;21 H is loaded to
                 accumulator.
OUT     00 H    ;PA0 and PA5 are high (red
                 LEDs of north and south
                 remain glow).
MVI A,  12 H    ; 12 H is loaded to
                 accumulator.
OUT     01 H    ; PB1 and PB4 are high
                 (yellow LEDs of East and
                 west glow).
MVI A,  00 H    ; 00 H is loaded to acc.
OUT     02 H    ; PC0 to PC7 are low.
CALL    DELAY II; Calls the delay subroutine
                 program (DELAY II).
CALL    DISPLAY ; Calls the Display
                 subroutine program for the
                 display of delay time.
MVI A,  21 H    ; 21 H is loaded to
                 accumulator.
OUT     00 H    ;PA0 and PA5 are high (red
                 LEDs of north and south
                 glow).
OUT     01 H    ; PB0 and PB5 are high (red
                 LEDs of East and west
                 glow).
MVI A,  96 H    ; 96 H is loaded to acc.
OUT     02 H    ; PC1, PC2, PC4 and PC7
                 are high (Traffic from west
```

```
                                    to south and east to north is
                                    allowed).
CALL        DELAY I                 ; Calls the delay subroutine
                                    program (DELAY I).
CALL        DISPLAY                 ; Calls the Display
                                    subroutine program for the
                                    display of delay time.
MVI A,      12 H                    ; 12 H is loaded to
                                    accumulator.
OUT         00 H                    ;PA1 and PA4 are high
                                    (yellow LEDs of north and
                                    south glow).
OUT         01 H                    ; PB1 and PB4 are high
                                    (yellow LEDs of East and
                                    west glow).
MVI A,      00 H                    ; 00 H is loaded to acc.
OUT         02 H                    ; PC0 to PC7 are low.
CALL        DELAY II                ; Calls the delay subroutine
                                    program (DELAY II).
CALL        DISPLAY                 ; Calls the Display
                                    subroutine program for the
                                    display of delay time.
MVI A,      0C H                    ; 0C H is loaded to
                                    accumulator.
OUT         00 H                    ;PA2 and PA3 are high
                                    (green LEDs of north and
                                    south glow).
MVI A,      21 H                    ; 21 H is loaded to
                                    accumulator.
OUT         01 H                    ; PB0 and PB5 are high (red
                                    LEDs of East and west
                                    glow).
MVI A,      09 H                    ; 09 H is loaded to acc.
OUT         02 H                    ; PC0 and PC3 are high
                                    (Traffic from north to east
                                    and south to west is
                                    allowed).
CALL        DELAY I                 ; Calls the delay subroutine
                                    program (DELAY I).
CALL        DISPLAY                 ; Calls the Display
                                    subroutine program for the
                                    display of delay time.
MVI A,      12 H                    ; 12 H is loaded to
                                    accumulator.
```

```
OUT         00 H        ;PA1 and PA4 are high
                         (yellow LEDs of north and
                         south glow).
MVI B,      21 H        ; 21 H is loaded to acc.
OUT         01 H        ; PB0 and PB5 are high (red
                         LEDs of East and west
                         glow).
MVI A,      00 H        ; 00 H is loaded to acc.
OUT         02 H        ; PC0 to PC7 are low.
CALL        DELAY II    ; Calls the delay subroutine
                         program (DELAY II).
CALL        DISPLAY     ; Calls the Display
                         subroutine program for the
                         display of delay time.
MVI A,      21 H        ; 21 H is loaded to
                         accumulator.
OUT         00 H        ;PA0 and PA5 are high (red
                         LEDs of north and south
                         glow).
OUT         01 H        ; PB0 and PB5 are high (red
                         LEDs of East and west
                         glow).
MVI A,      96 H        ; 99 H is loaded to acc.
OUT         02 H        ; PC1, PC2, PC4 and PC7
                         are high (Traffic from north
                         to west and south to east is
                         allowed).
CALL        DELAY I     ; Calls the delay subroutine
                         program (DELAY I).
CALL        DISPLAY     ; Calls the Display
                         subroutine program for the
                         display of delay time.
MVI A,      12 H        ; 12 H is loaded to
                         accumulator.
OUT         00 H        ;PA1 and PA4 are high
                         (yellow LEDs of north and
                         south glow).
OUT         01 H        ; PB1 and PB4 are high
                         (yellow LEDs of East and
                         west glow).
MVI A,      00 H        ; 00 H is loaded to acc.
OUT         02 H        ; PC0 to PC7 are low.
CALL        DELAY II    ; Calls the delay subroutine
                         program (DELAY II).
```

	CALL	DISPLAY	; Calls the Display subroutine program for the display of delay time.
	JMP	START	; Jump to START.

Subroutine Program for DELAY I:

Label	**Mnemonics**	**Operand**	**Comments**
DELAY I	MVI A,	09 H	; 09 H is loaded to accumulator.
	STA	2054	; Store it to memory location 2054 H.
	MVI A,	00 H	
	STA	2055 H	; for display of delay of 90 seconds for normal traffic.
	RET		; Returns to main program.

Subroutine Program for DELAY II:

Label	**Mnemonics**	**Operand**	**Comments**
DELAY II	MVI A,	01 H	; 01 H is loaded to accumulator.
	STA	2054	; Store it to memory location 2054 H.
	MVI A,	00 H	
	STA	2055 H	; for display of delay of 10 seconds for clearing the traffic during yellow light at the crossing.
	RET		; Returns to main program.

Subroutine Program for the display of Delay time (count down):

Label	**Mnemonics**	**Operand**	**Comments**
DISPLAY	CALL	0347 H	;Clears the display. It is the program stored in ROM of the kit.
AA	XRA	A	;Clears the accumulator
	MOV B,	A	;Clears the B register also.
	LXI H,	2050 H	;Intialize H-L pair where the current delay time is stored.

```
            CALL        05D0 H      ;Displays the current time in
                                      address field. It is the
                                      program stored in ROM of
                                      the Kit. Here blank is
                                      displayed in the address
                                      field.
            MVI A,      01 H        ;Stores 01 H in accumulator.
            MVI B,      00 H        ;Stores 00 H in B register.
            LXI H,      2054 H      ;Initialize H-L register pair
                                      with 2054 H.
            CALL        05D0 H      :Displays the current delay
                                      time in data field (90
                                      seconds or 10 seconds).
            LXI H,      2055 H      ;Initialize H-L register pair
                                      with 2055 H.
            MOV A,      M           ;Moves the least significant
                                      digit (LSD) of seconds to
                                      the accumulator.
            CPI         00 H        ;Compares with 00 H.
            JNZ         PROCEED     ;If Acc.is not 00 H, then
                                      jump to PROCEED.
            DCX         H           ;Decrement H-L register
                                      pair.
            MOV A,      M           ;Moves MSD of the
                                      seconds to accumulator.
            CPI         00 H        ;Compaers it with 00.
            RZ                      ; Return if both are zero.
PROCEED     MVI B,      02 H        ; Program for delay of 1 sec
                                      starts. It stores 02 H in
                                      accumulator.
YY          LXI D,      FA00 H      ;Intialize D-E register pair
                                      with FA00H.
NXT         DCX         D           ;Decrement D-E register
                                      pair.
            MOV  A,     D           ;Moves the contents stored
                                      in M(D-E) to Acc.
            ORA         E           ;The contents of A and E
                                      registers are ORed bit by
                                      bit.
            JNZ         NXT         ;If the result is not zero then
                                      jump to NXT else next
                                      instruction.
            DCR         B
            JNZ         YY          ; End of 1 sec delay.
            LXI H,      2055 H      ;Initialize H-L register pair
                                      with 2055 H.
```

```
        MOV A,      M       ;Moves the least significant
                            digit (LSD) of seconds to
                            the accumulator.
        CPI         00 H    ;Compares with 00 H.
        JZ          BB      ;If Acc.=00 H, then jump to
                            BB. Else goes to next
                            statement.
        SUI         01 H    ; Subtract 01 from the acc.
        MOV M,      A       ; Store it to memory
                            location.
        JMP         AA      ; Jump to AA for the
                            display.
BB      MVI A,      09 H    ; Move 09 to Acc.
        MOV  M,     A       ; It is stored in the memory
                            location.
        DCX  H              ; Decrement of H-L register
                            pair.
        MOV A,      M       ; Contents of this location is
                            moved to Acc.
        CPI         00 H    ; Compared with 00.
        JNZ         CC      ; If not zero jump to CC.
        INX         H
        MVI A,      00 H    ; Get back 0 in the same
                            location.
        MOV M,      A
        JMP         AA      ; Jump to AA for the
                            display.
CC      SUI         01 H    ; Subtract 01 from the
                            location.
        MOV M,      A       ; After subtraction store to
                            the memory location.
        JMP         AA      ; Jump to AA for the
                            display.
```

15.6 MICROPROCESSOR BASED STEPPER MOTOR CONTROL

In this section the design of microprocessor based stepper motor control will be discussed. The stepper motor has four windings such that the motor rotates in precise steps from one fixed position to another. A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motor's rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied. A stepper motor can be a good choice whenever controlled movement is required. They can be used where the control rotation angle, speed, position and synchronism are needed. Because of the inherent advantages

stepper motors have found their place in many different applications. Some of these include printers, plotters, high end office equipment, hard disk drives, medical equipments, fax machines, automotive and many more.

The stepper motor is to be interfaced with microprocessor by 8255A PPI and some buffers/ amplifiers. The four windings (coils) of the stepper motor are shown in figure 15.12. However, figure 15.13 shows the block diagram of the interfacing connections for the stepper motor. One set of coils of the stepper motor is energized using 12 V d. c. supply in the form of pulse. After energizing one set of coil windings some time delay is provided through software, then the supply is given to other set of windings. The speed of the motor will be governed by the delay introduced in the system.

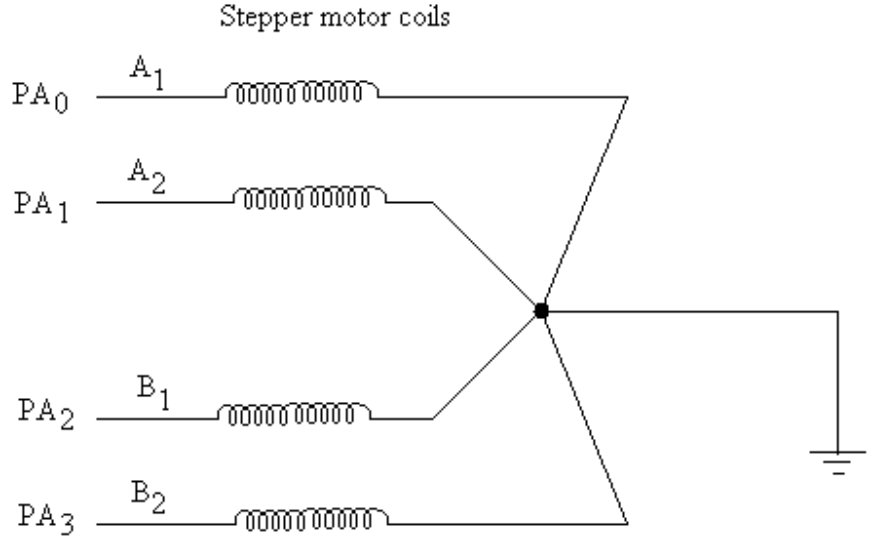


Fig. 15.12

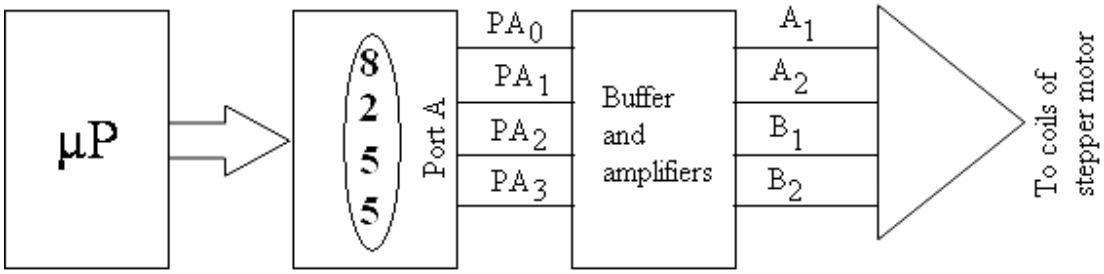


Fig. 15.13

For the clockwise movement of the stepper motor, it is energized in the sequence:

A_1 A_2, **A_2 B_1,** **B_1 B_2,** **B_2 A_1** and then repeats.

Similarly, in the anti-clock wise movement of the stepper motor, it is energized in the sequence:

A_1 A_2, **B_2 A_1,** **B_1 B_2,** **A_2 B_1** and then repeats.

Number of steps or movements, for the stepper motor to be moved, are loaded in the register C. For example the stepper motor to be moved for 20 steps then it equivalent hexadecimal number 14 H is to be loaded this register before the start of the run. Similarly, if the stepper motor is to be rotated in clock wise direction then load 00 in

register B or load 01 in register B if the motor is to be moved in anti-clock wise direction. The flow chart for the movement of the stepper motor is shown in figure 15.14.

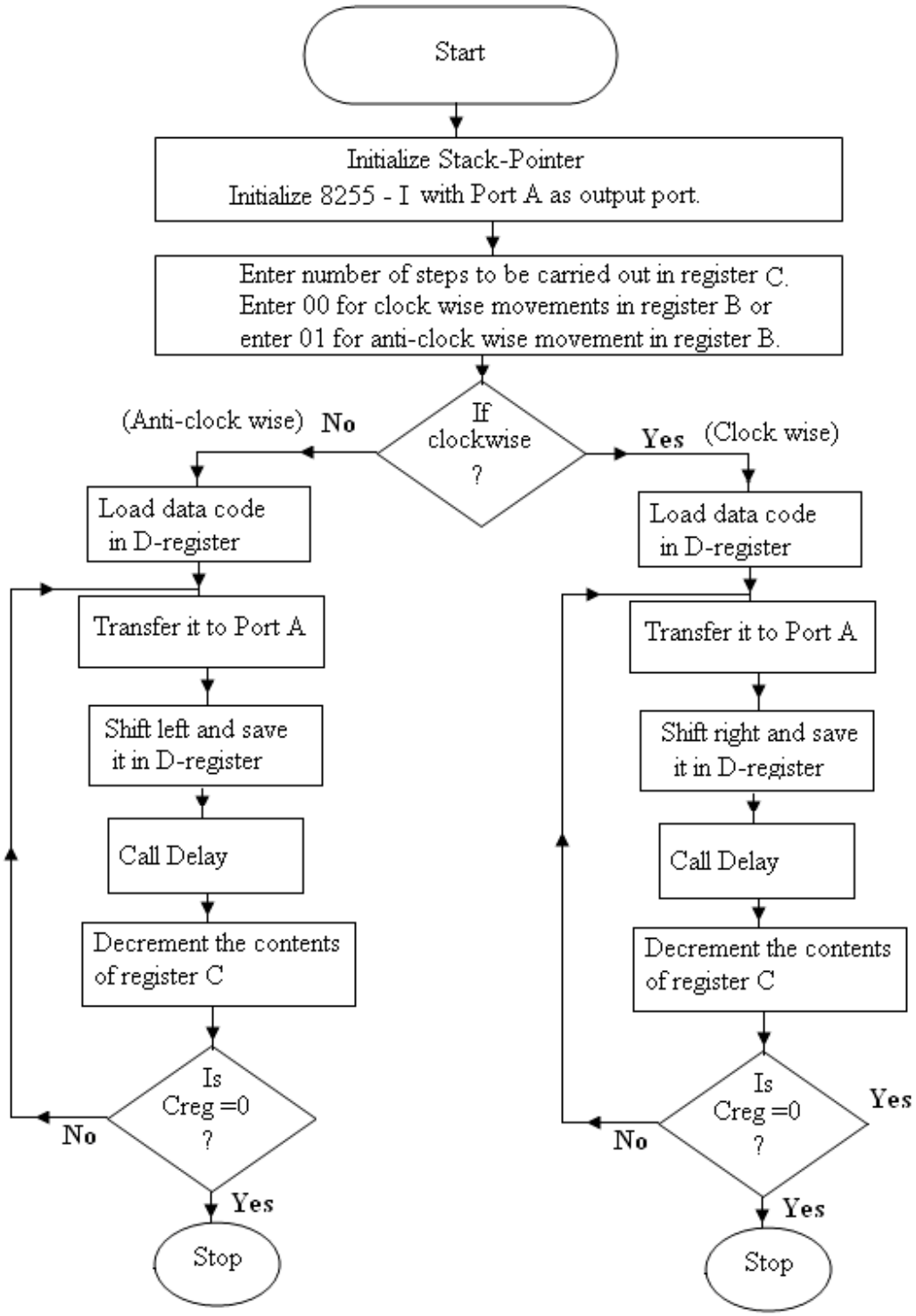
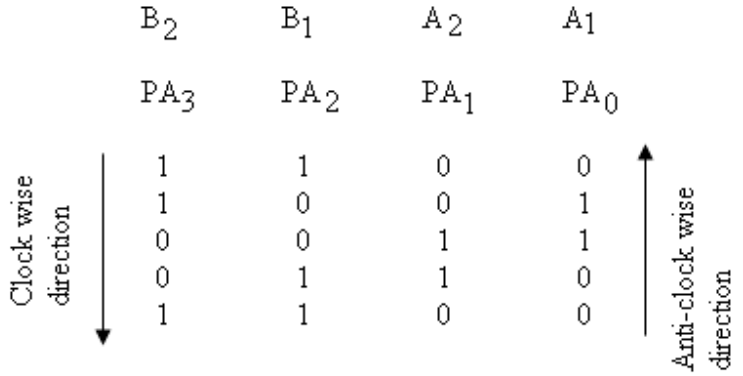
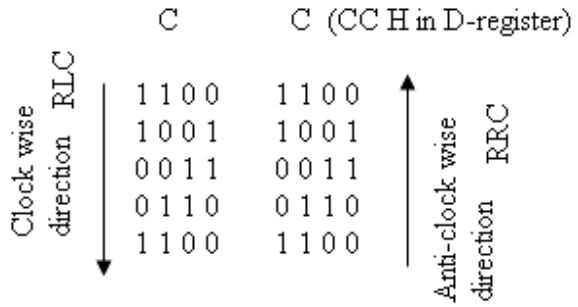


Fig. 15. 14

Further, the bits (PA_0, PA_1, PA_2 and PA_3) of port A of 8255A are to be connected to A_1, A_2, B_1 and B_2 coils of the motor respectively. The movements of the motor in clockwise or anti-clock wise direction should be as:



This is possible if CC H is loaded to the D-register and then data is rotated clockwise or anti-clock wise direction with RRC or RLC instructions as shown below:



The assembly language program for the above description is given below which is self explanatory:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	;Initialize the stack pointer.
	MVI A,	80H	;Control word for 8255-I.
	OUT	03H	;Works all the ports of 8255-I as output ports.
	MVI C,	14 H	;14 H (20 steps for movement) is loaded to C-register. It is used as counter.
	MVI B,	YY H	; YY may be 00 H or 01 H for clockwise or anti-clock wise direction respectively.
	MOV A,	B	; Move the contents of B-register to accumulator.

```
        CPI     01 H        ; Check if movement is
                              clockwise or anti-clock
                              wise.
        JNZ     CLKWISE     ;If condition is satisfied then
                              clockwise direction else
                              anti-clock wise.
        MVI D,  CC H        ; Move CC H to D-register.
LOOP1   MOV A,  D           ; This data is moved to
                              accumulator.
        OUT     00 H        ; Data is sent to coils of the
                              motor through Port A of
                              8255A.
        RLC                 ; Data is rotated left for anti-
                              clock wise movement of
                              the motor.
        MOV D,  A           ; After rotation data is stored
                              in D register.
        PUSH  PSW           ; Push Acc content and flag
                              content to Stack.
        PUSH  D             ; Push content of D-E
                              register pair to Stack.
        PUSH  B             : Push content of B-C
                              register pair to Stack.
        CALL    DELAY       ; Call subroutine program
                              for delay as per the
                              requirement.
        POP  B              ; Get back the content of B-
                              C register pair from the
                              Stack.
        POP  D              ; Get back the content of D-
                              E register pair from the
                              Stack.
        POP  PSW            ; Get back the  Acc and flag
                              content from the Stack.
        DCR     C           ; Decrement in number of
                              counts.
        JNZ     LOOP1       ; If counts are not 0 then
                              jump to LOOP1 for next
                              movement, else stop
                              processing.
        HLT
CLKWISE MVI D,  CC H        ; Move CC H to D-register.
LOOP2   MOV A,  D           ; This data is moved to
                              accumulator.
```

```
OUT         00 H        ; Data is sent to coils of the
                          motor through Port A of
                          8255A.
RRC                     ; Data is rotated right for
                          clock wise movement of
                          the motor.
MOV D,      A           ; After rotation data is stored
                          in D register.
PUSH  PSW               ; Push Acc content and flag
                          content to Stack.
PUSH  D                 ; Push content of D-E
                          register pair to Stack.
PUSH  B                 : Push content of B-C
                          register pair to Stack.
CALL        DELAY       ; Call subroutine program
                          for delay as per the
                          requirement.
POP  B                  ; Get back the content of B-
                          C register pair from the
                          Stack.
POP  D                  ; Get back the content of D-
                          E register pair from the
                          Stack.
POP  PSW                ; Get back the  Acc and flag
                          content from the Stack.
DCR         C           ; Decrement in number of
                          counts.
JNZ         LOOP2       ; If counts are not 0 then
                          jump to LOOP2 for next
                          movement.
HLT                     ; stop processing.
```

Subroutine programs to introduce delay for required time may be written as discussed earlier.

15.7 MICROPROCESSOR BASED WASHING MACHINE CONTROLLER

The automatic washing machines available in the market are microprocessor based. Here conditional sequences are considered which are controlled by microprocessor along with interfacing devices. It basically involves trigger inputs as well as specific delays. In other words there are some trigger inputs which leads the washing machine to work with certain time delays as well.

The port A and Port B of 8255-I are used for the purpose of control of certain functions of washing machine. Let the bits D0 and D1 of port A are used for introducing time delay for certain time as given below:

D1 D0

0	0	No time delay required.
0	1	Delay 1 for certain time delay (say for t1)
1	0	Delay 2 for certain other time delay (say for t2 seconds)
1	1	Not used

D2 bit of the Port A is used for the tub motor rotating at the drying speed.
D3 bit of the port A is used for water outlet.
D4 bit of this port is used for the tub motor rotating at wash or rinse speed.
D5 bit of this port is used for water inlet.
D6 and D7 bits are not used.

The devices connected to the Port A of 8255-I for the purpose of controlling events during the operation of washing machine are as given in figure 15. 15

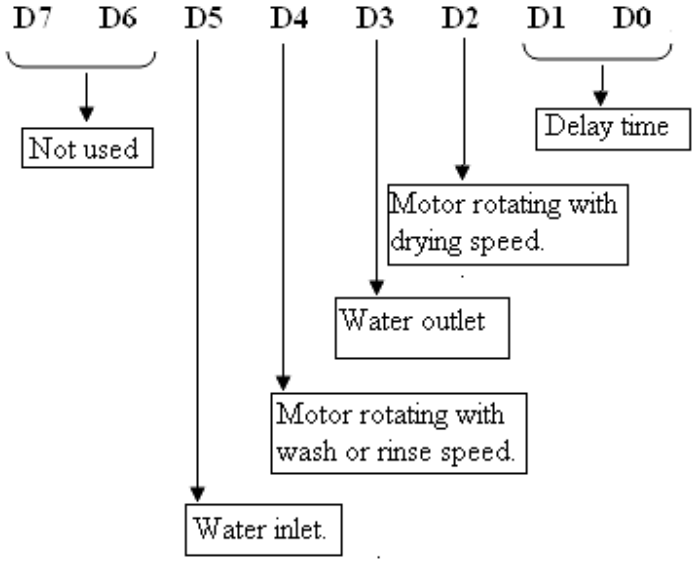


Fig. 15.15

Consider a water level sensor connected with a D1 bit of port B of 8255-I. The port B is used as the input port which will indicate if the water is filled to the specified level.

The sequence of events to be undertaken during the operation of washing machine is as given below:

1. The tub is filled with water up to the specified level with the water inlet. The level of the water will be sensed by the water level sensor.
2. The tub is rotated at wash or rinse speed for a fixed time t1.
3. The tub is allowed to empty for a fixed time t2.
4. The tub is filled again with water up to a specified level whose level will be sensed with water level sensor.
5. The tub is rotated again with wash or rinse speed for the fixed time t1.

6. The tub is allowed to empty again for a fixed time t2.
7. Switch on dryer for time t1.

Figure 15.16 shows the flow chart for the sequence of events to be carried out during the washing machine operation.

The assembly language program for the above mentioned sequence of events is given below:

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	;Initialize the stack pointer.
	MVI A,	82 H	;Control word for 8255-I.
	OUT	03 H	;Works port A of 8255-I as output ports and port B as input port.
	LXI H,	2500 H	; Initialize H-L register pair which indicate address of the look-up table.
NEXT 1	MOV A,	M	; Get the data from the address of the look-up table.
	OUT	00 H	; The accumulator contents go the port A of 8255-I.
	ANI	03 H	; Check if delay required.
	JZ	NEXT	; If the delay is not required jump to NEXT.
	ANI	02 H	; Check if delay-II is required.
	JZ	SB	; If yes jump to SB.
	CALL	DELAY I	; Call delay program of t1 seconds.
	JMP	PT1	; Jump to PT1.
SB	CALL	DELAY II	; Call delay program of t2.
	JMP	PT1	; Jump to PT1.
NEXT	INX	H	; Get the address of next data from the look-up table.
LOOP	IN	01 H	; Get the input from the water level sensor.
	SUB	M	; Check if the water level is equal to the required level.
	JNZ	LOOP	; If no jump to LOOP.
PT1	INX	H	; Get the address of next data from the look-up table.
	MOV A,	M	; Data is moved to Acc.
	CPI	00 H	; Data is compared with 00.

```
        JNZ         NEXT 1      ; If not zero jump to
                                  NEXT1.
        HLT                     ; Else Halt.
```

Subroutine program for DELAY I:

Label	**Mnemonics**	**Operand**	**Comments**
DELAY I	LXI D,	FFFF H	
LOOP 1	DCX	D	;Decrement D-E register pair.
	MOV A,	D	;Moves the contents stored in M_{D-E} to Acc.
	ORA	E	;The contents of A and E registers are ORed bit by bit.
	JNZ	LOOP 1	;If the result is not zero then jump to DELAY else next instruction.
	RET		;Return to main program.

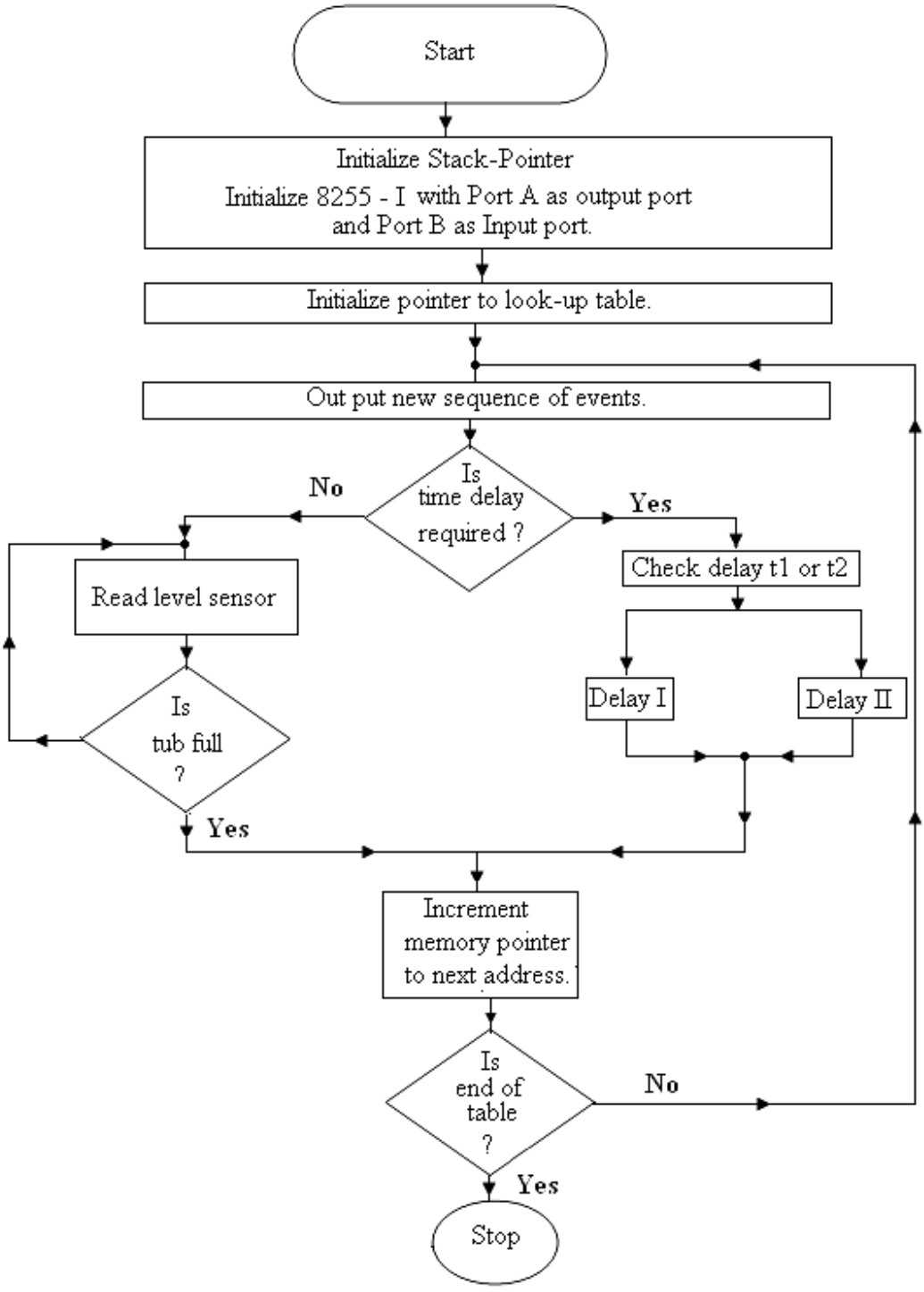

Start
Initialize Stack-Pointer
Initialize 8255 - I with Port A as output port
and Port B as Input port.
Initialize pointer to look-up table.
Out put new sequence of events.
Is time delay required ?
No
Yes
Read level sensor
Check delay t1 or t2
Delay I
Delay II
Is tub full ?
Yes
Increment memory pointer to next address.
Is end of table ?
No
Yes
Stop

Fig. 15.16

Subroutine program for DELAY II

Label	Mnemonics	Operand	Comments
DELAY II	LXI D,	DEEE H	
LOOP 2	DCX	D	; Decrement D-E register pair.
	MOV A,	D	; Moves the contents stored in M_{D-E} to Acc.
	ORA	E	; The contents of A and E registers are ORed bit by bit.
	JNZ	LOOP 2	; If the result is not zero then jump to DELAY else next instruction.
	RET		; Return to main program.

15.8 MICROPROCESSOR BASED WATER LEVEL CONTROLLER

Water level controller means water is to be pump out from the underground water tank to the anther water tank situated on the top of the building for the distribution of water to the other parts of the building. When the upper tank is empty or the level goes below to a certain level, the pump should be automatically switched on. Further, if the tank gets filled up to a certain level (top level) the pump should be switched off. The level of the water inside the upper water tank should be displayed on the seven segment units connected separately with the microprocessor kit. For this the tank to be filled is divided in to 8 equal regions which are say 10 cm apart. So at every 10 cm, a metallic probe is situated.

The arrangement of 8 probes which are to be immersed in the water tank is shown in figure 15.17. For the controller 8 metallic probes connected to +5 V d.c. supply through resistance are immersed in the upper tank. The difference in heights between the probes is equal to 10 cm (say). Besides the probe, another metallic strip is also immersed in the tank, which is grounded. The eight probes are also connected to the inputs of 8 inverters as shown in figure 15.18. The outputs of the inverters are connected to 8 bits of port A of 8255-I, which is used as input port. When any of the probes is immersed in the water (or water level is below the probe), the output of the corresponding inverter will provide a logic 0 to its bit of port A. However, if the water level touches the probe or above the probe, then it provides logic 1 to the corresponding bit of port A. The status of water level in the upper tank can directly be read by the microprocessor through the input statement. For example, if only one probe is immersed in the water, then port A of 8255-I will read it 01, similarly for the others. Table 15.4 shows the data to be read out by 8255-I, when different probes are immersed in the water.

When the water level of the tank is below 10 cm, the pump is automatically switched on; however, when its level reaches to 80 cm, the pump will be switched off.

Figure 15.19 shows the connection to the two segment display units connected to the port B of 8255-I for displaying the level of water in the upper tank in centimeters i.e. if the water level is between 1 and 2 probe, then the display unit will display 10, similarly for the other levels (ref. table 15.4).

Table 15.4

Sr.No.	Probe number/ numbers immersed in water	Data to be read out by the microprocessor		Level in cms.
		in Binary	in Hexadecimal	
1.	1	0000 0001	01 H	10
2.	1 and 2	0000 0011	03 H	20
3.	1 to 3	0000 0111	07 H	30
4.	1 to 4	0000 1111	0F H	40
5.	1 to 5	0001 1111	1F H	50
6.	1 to 6	0011 1111	3F H	60
7.	1 to 7	0111 1111	7F H	70
8.	1 to 8	1111 1111	FF H	80

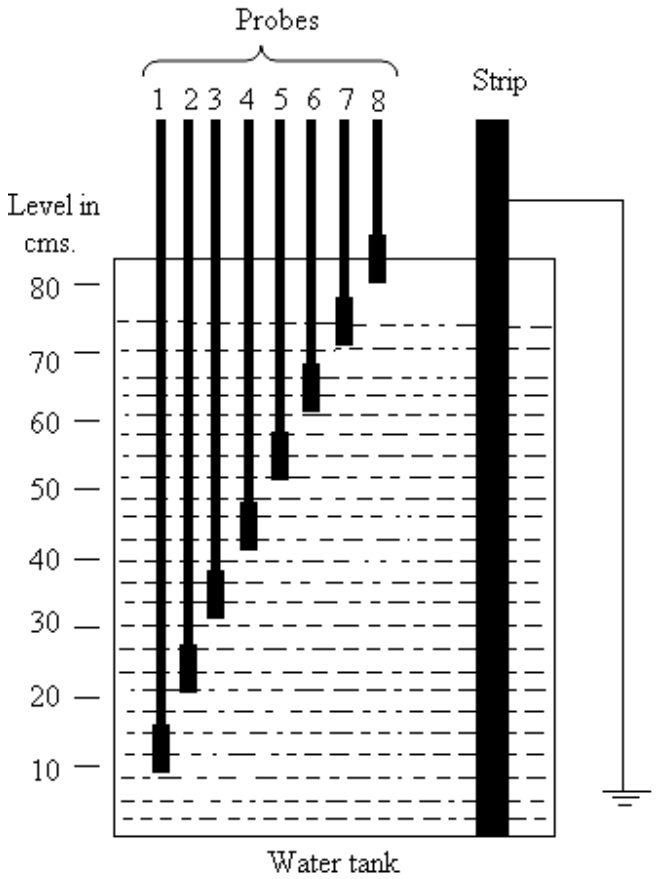


Fig. 15.17

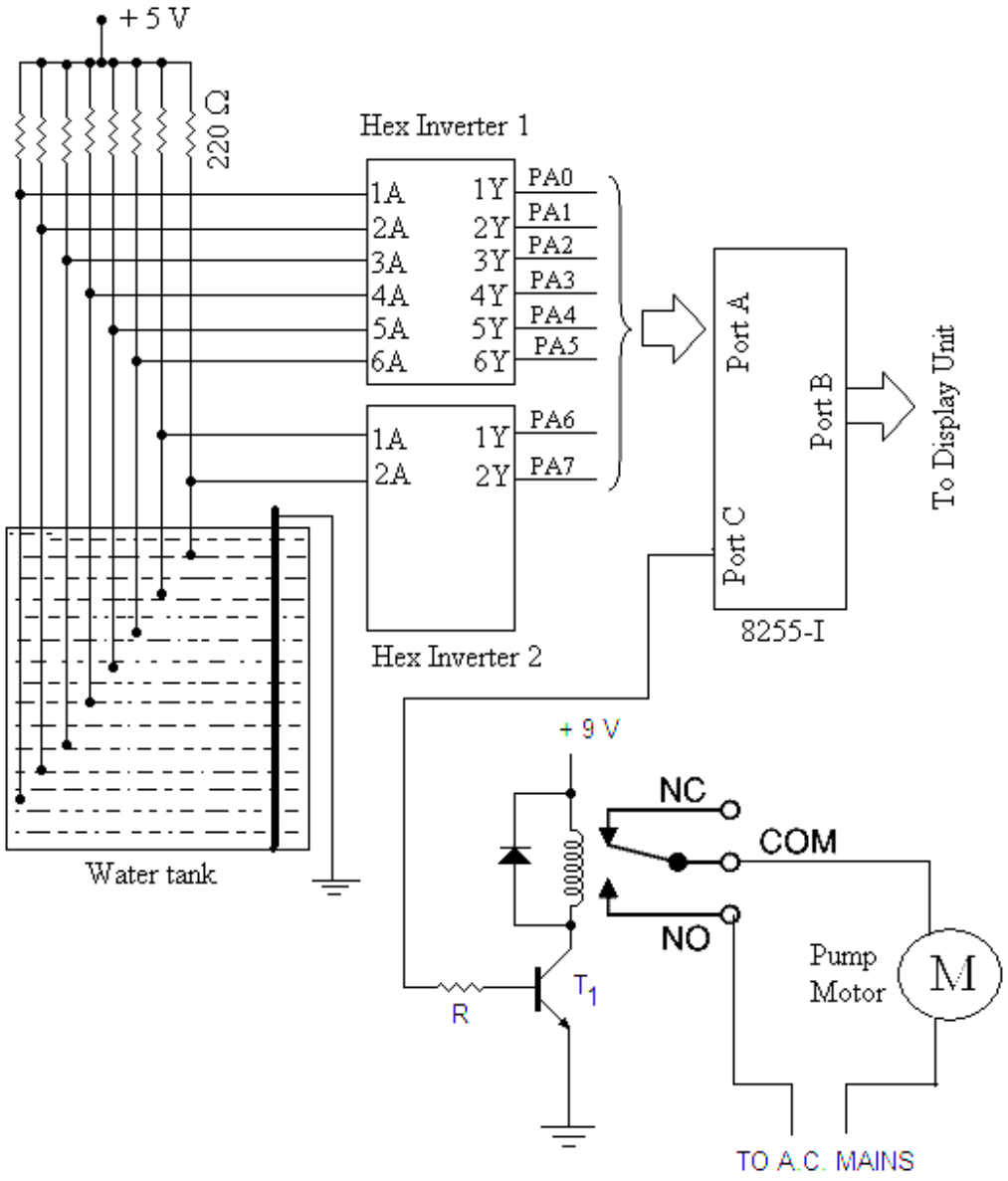

+ 5 V
220 Ω
Hex Inverter 1
1A
2A
3A
4A
5A
6A
1Y
2Y
3Y
4Y
5Y
6Y
PA0
PA1
PA2
PA3
PA4
PA5
1A
2A
1Y
2Y
PA6
PA7
Hex Inverter 2
Port A
Port B
Port C
8255-I
To Display Unit
Water tank
+ 9 V
NC
COM
NO
R
T1
Pump
Motor
M
TO A.C. MAINS

Fig. 15.18

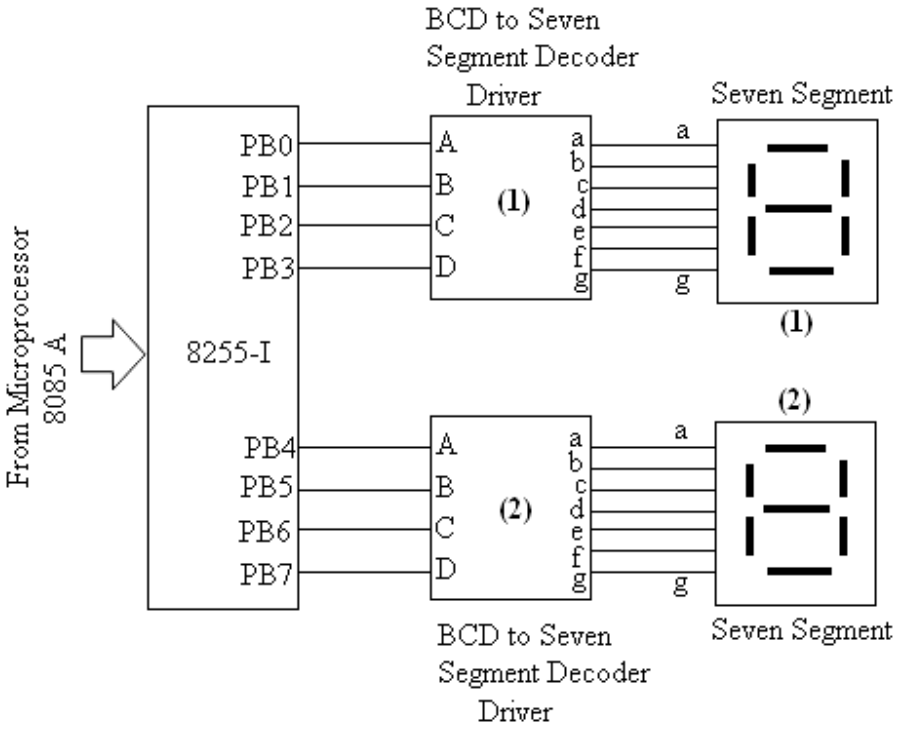


Fig. 15.19

The assembly language program for the design of water level controller with the above mentioned conditions is given below, which is self explanatory.

Label	**Mnemonics**	**Operand**	**Comments**
	LXI SP,	XXXX H	;Initialize the stack pointer.
	MVI A,	90H	;Control word for 8255-I.
	OUT	03H	;Works port A of 8255-I as input port and other ports as output ports.
START	IN	00 H	; Read the water level through input port A.
	LXI H,	25FF H	; Intilise the H-L register pair (starting address of the look up table).
	MOV C,	M	; Data is moved to C-register which is used as counter.
AGAIN	INX	H	; Increment the H-L register pair.
	CMP	M	; Compare the two levels.
	JZ	PT1	; If two levels are equal, then jump to PT1.
	DCR	C	; Decrement the counts.

```
        JNZ     AGAIN   ; If counts are not zero then
                          jump to AGAIN.
PT1     INR     H       ; Increment the content of
                          H-register.
        MOV A,  M       ; Get the water level in
                          Accumulator.
        OUT     01 H    ; Move it to Port B of 8255-I
                          for the display.
        CPI     10 H    ; Compare this data with 10
                          cm.
        JNC     NXT     ; If the water is more than 10
                          cm, then move it to NXT.
        PUSH    PSW     ; Else save the Acc contents
                          in the stack.
        MVI A,  01 H    ; 01 H is loaded to the
                          accumulator.
        OUT     02 H    ; PC0 is high (Switch on the
                          motor for lifting the water).
        POP     PSW     ; Get back the accumulator
                          contents from the stack.
NXT     CPI     80 H    ; Compared the data with 80
                          cm.
        JC      START   ; If the data is less than 60,
                          then jump to START.
        MVI A,  00 H    ; 00 H is loaded to the
                          accumulator.
        OUT     02 H    ; PC0 is low (Switch off the
                          motor).
        CALL    DELAY   ; Calls delay program.
        JMP     START   ; Jump to start.
```

Delay Program may be written as discussed in earlier program as per the requirement.

LOOK UP TABLE

25FF H	09 H (Counts)
2600 H	00 H
2601 H	01 H
2602 H	03 H
2603 H	07 H
2604 H	0F H
2605 H	1F H
2606 H	3F H
2607 H	7F H
2608 H	FF H

Water level in Cms.

2700 H	00 H	(00 Cm.)
2701 H	10 H	(10 Cm.)
2702 H	20 H	(20 Cm.)
2703 H	30 H	(30 Cm.)
2704 H	40 H	(40 Cm.)
2705 H	50 H	(50 Cm.)
2706 H	60 H	(60 Cm.)
2707 H	70 H	(70 Cm.)
2708 H	80 H	(80 Cm.)

15.9 MICROPROCESSOR BASED TEMPERATURE CONTROLLER

In this section a very simple design of temperature controller is being discussed. The temperature of water bath or of any other device is to be controlled or maintained constant. This is possible if the a.c. mains supplied to the device is switched off if the temperature of the device is more than the required temperature; and it is switched on if the temperature is less than the required temperature. A temperature sensor may be used to sense the temperature of the device. The sensor (say a thermocouple) may provide the thermo e.m.f. (d.c. signal) corresponding to the temperature. The d.c. signal may be converted to the digital signal with the help of A/D converter IC 0809. The output of the A/D converter can be read by the microprocessor through the 8255 PPI.

The arrangement for the control is shown in figure 15.20. Channel 0 of A/D converter is used to convert the sensor voltage to equivalent digital signal. This digital signal can be read out by the microprocessor 8085A through port A (used as input port) of 8255-I provided on the microprocessor kit. Port C_{Lower} of 8255 is used as output port to send the SOC (Start of Conversion) signal and other bits for the channel selection of A/D converter 0809. For the selection of channel 0 ALE (Address Latch Enable) and SOC are made high by the output port C_{Lower} of 8255. Port C_{Upper} of 8255 is uses as output port. Before reading the input data (digital value) by the microprocessor, EOC (End of Conversion) signal PC7 is checked for logic 1.The bit PB0 of port B (used as output port) of 8255 may be used to control the heating arrangement of the device through the relay system.

Before running the assembly language program, it is to be calibrated as per the sensor used in this arrangement. Note down the temperature and digital value (in Hexadecimal) which may be stored in some memory locations as a look-up table. This look-up table may be used for the display of the temperature on the address/data field of the microprocessor kit using its monitor program. The digital data of the temperature of the device to be maintained constant, may be saved in a memory location say 2500 H. By comparing the current data (of temperature) with the saved (or required) data, the electrical heating system may be switched on/off.

Figure 15.21 shows the flow chart for the controller.

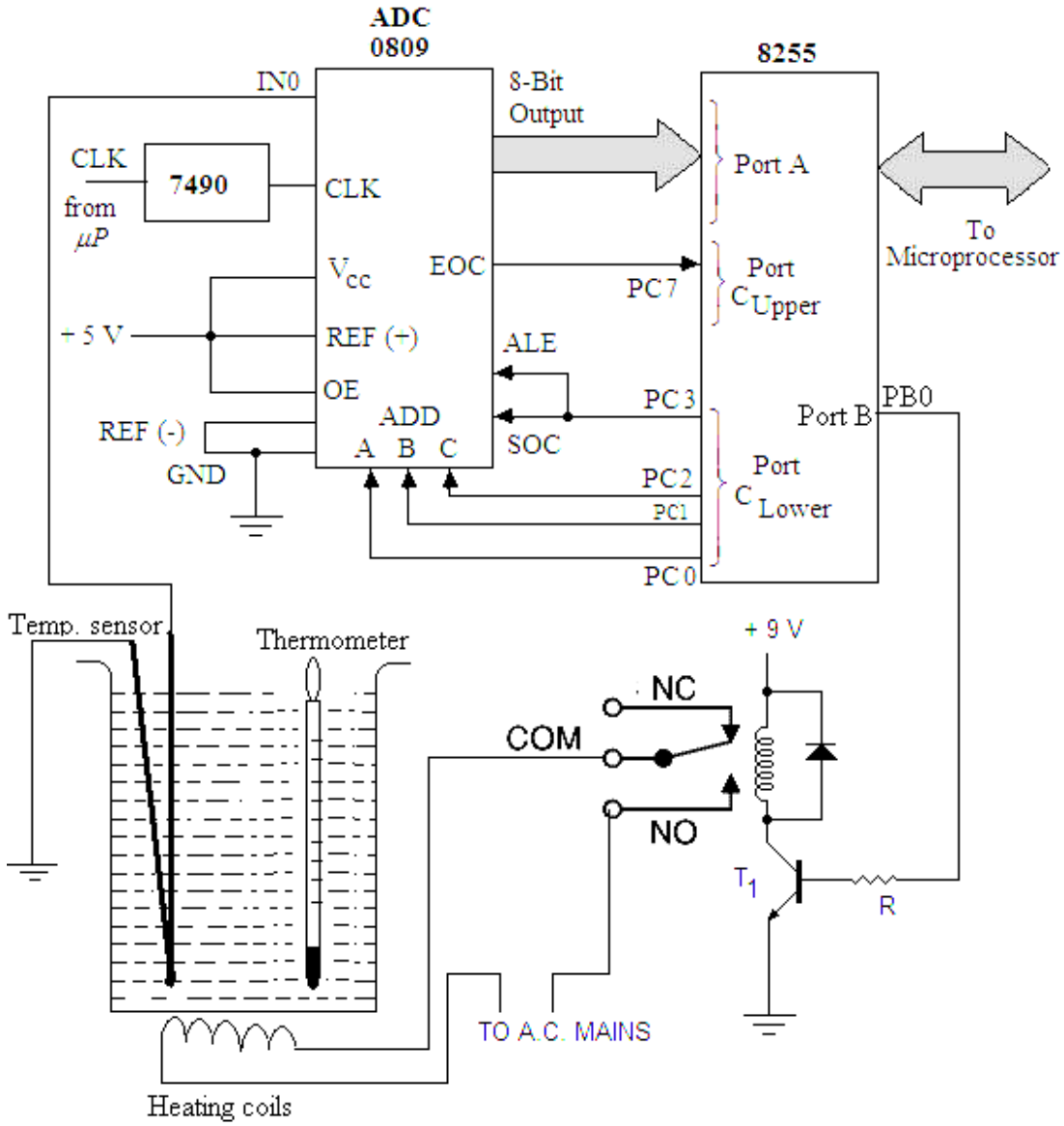


Fig. 15.20

PROGRAM

Label	**Mnemonics**	**operand**	**Comments**
	MVI	A, 98 H	; Initialize 8255-I to work Port A and Port C_{Upper} as input ports, and port C_{Lower} as output port.
	OUT	03 H	; Write the control word in the control word register of 8255-I.
	LXI H,	2500 H	; Initialize H-L register pair for address where required temp. T is saved.

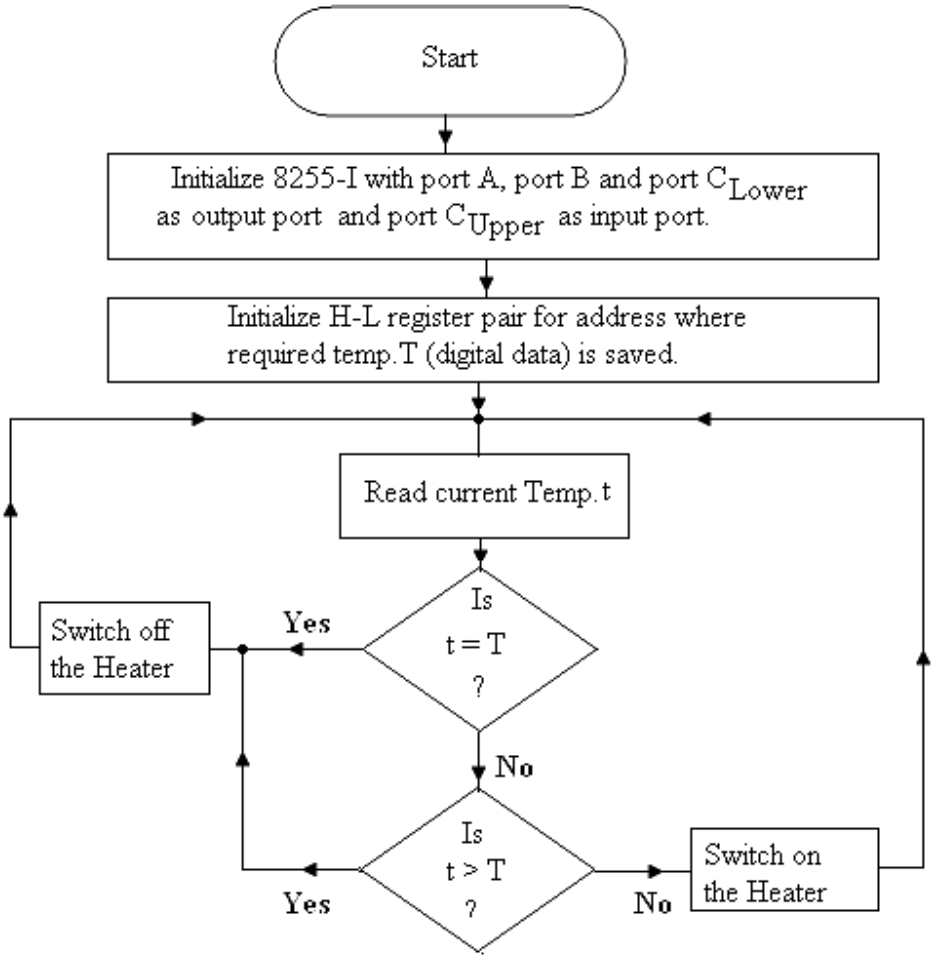


Fig. 15.21

START	MVI	A, 00 H	; Load accumulator with 00H.
	OUT	02 H	; 00H is sent to Port C_{Lower} to select channel 0.
	MVI	A, 08 H	; Load 08 H to accumulator.
	OUT	02 H	; ALE and SOC are enabled (high).
	MVI	A, 00 H	; Load 00 H to accumulator.
	OUT	02 H	; ALE and SOC will be low. A pulse is applied from high to low for the conversion through PC_3 without affecting the channel selected.
READ	IN	02 H	; Read End of conversion (PC_7).
	RAL		; Rotate left to check if PC_7 is one.
	JNC	READ	; If not 1 READ again.
	IN	00 H	; Read digital output of the A/D converter (Temp. t).

```
        STA     2501 H      ; Store the result in 2501 H memory
                            location.
        CMP     M           ; Compare the two temperatures for
                            equality.
        JNZ     NEXT        ; If not equal then jump to NEXT.
LOOP1   MVI A,  00 H        ; Switch off the heater.
        OUT     01 H        ; PB0 is low.
        JMP     START       ; Jump to START.
NEXT    SUB     M           ; Check if t>T ?
        JNC     LOOP1       ; If yes then jump to LOOP1.
        MVI A,  01 H        ; Switch on the heater.
        OUT     01 H        ; PB0 is high.
        JMP     START       ; Jump to START.
```

PROBLEMS

1. How will you design the digital clock on the microprocessor kit? Hours and minutes should be displayed on the address field and seconds should be displayed on the data field. Monitor programs of the kit may be used for the display of time. Write program in assembly language of 8085.
2. Design the digital clock which can display the time in 24 hours form on the microprocessor kit. Write program in assembly language of 8085.
3. In problem 1, on time should also be introduced. Write program in assembly language of 8085.
4. Discuss the design of microprocessor based LED dial clock. Write program in assembly language of 8085.
5. Discuss the design of microprocessor based running light in which some LEDs can run in a sequence. Write program in assembly language of 8085.
6. Discuss the design of microprocessor based school bell system. Write program in assembly language of 8085.
7. Discuss the design of microprocessor based Traffic light (simple arrangement). Write program in assembly language of 8085.
8. Discuss the design of microprocessor based Traffic light (complicated arrangement) in which traffic is allowed in left-right direction also. Write program in assembly language of 8085.
9. In the above design of microprocessor based traffic light, the delay time should also be displayed on the address and data field of the microprocessor kit. Write program in assembly language of 8085.
10. Discuss the design of microprocessor based Stepper Motor controller. Write program in assembly language of 8085.
11. Write program in assembly language of 8085 to control the different function of washing machine.

12. Discuss the water level controller which can pump out the water from the underground tank to the tank placed on the roof of the building. The motor of the pump should be switched on if the water level in the upper tank is less than certain level. The pump should be switched off when the water level in the upper tank is completely filled. Program for same should be written in assembly language of the 8085 microprocessor.
13. Discuss the design of microprocessor based Temperature controller which can maintain the temperature of a device constant. Write program in assembly language of 8085.

Appendix - I

Instruction codes of 8085 Microprocessor in Alphabetical Order

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags Affected
ACI	Data	CE	2	7	x	x	x	x	x	All
ADC	A	8F	1	4	x	x	x	x	x	All
ADC	B	88	1	4	x	x	x	x	x	All
ADC	C	89	1	4	x	x	x	x	x	All
ADC	D	8A	1	4	x	x	x	x	x	All
ADC	E	8B	1	4	x	x	x	x	x	All
ADC	H	8C	1	4	x	x	x	x	x	All
ADC	L	8D	1	4	x	x	x	x	x	All
ADC	M	8E	1	7	x	x	x	x	x	All
ADD	A	87	1	4	x	x	x	x	x	All
ADD	B	80	1	4	x	x	x	x	x	All
ADD	C	81	1	4	x	x	x	x	x	All
ADD	D	82	1	4	x	x	x	x	x	All
ADD	E	83	1	4	x	x	x	x	x	All
ADD	H	84	1	4	x	x	x	x	x	All
ADD	L	85	1	4	x	x	x	x	x	All
ADD	M	86	1	7	x	x	x	x	x	All
ADI	Data	C6	2	7	x	x	x	x	x	All

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
ANA	A	A7	1	4	0	1	x	x	x	CY is 0 AC is 1
ANA	B	A0	1	4	0	1	x	x	x	CY is 0 AC is 1
ANA	C	A1	1	4	0	1	x	x	x	CY is 0 AC is 1
ANA	D	A2	1	4	0	1	x	x	x	CY is 0 AC is 1
ANA	E	A3	1	4	0	1	x	x	x	CY is 0 AC is 1
ANA	H	A4	1	4	0	1	x	x	x	CY is 0 AC is 1
ANA	L	A5	1	4	0	1	x	x	x	CY is 0 AC is 1
ANA	M	A6	1	7	0	1	x	x	x	CY is 0 AC is 1
ANI	Data	E6	2	7	0	1	x	x	x	CY is 0 AC is 1
CALL	Label	CD	3	18	–	–	–	–	–	None
CC	Label	DC	3	18/9	–	–	–	–	–	None
CM	Label	FC	3	18/9	–	–	–	–	–	None
CMA		2F	1	4	–	–	–	–	–	None
CMC		3F	1	4	x	–	–	–	–	Only CY
CMP	A	BF	1	4	x	x	x	x	x	All
CMP	B	B8	1	4	x	x	x	x	x	All
CMP	C	B9	1	4	x	x	x	x	x	All
CMP	D	BA	1	4	x	x	x	x	x	All
CMP	E	BB	1	4	x	x	x	x	x	All
CMP	H	BC	1	4	x	x	x	x	x	All
CMP	L	BD	1	4	x	x	x	x	x	All
CMP	M	BE	1	7	x	x	x	x	x	All

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags Affected
CNC	Label	D4	3	18/9	–	–	–	–	–	None
CNZ	Label	C4	3	18/9	–	–	–	–	–	None
CP	Label	F4	3	18/9	–	–	–	–	–	None
CPE	Label	EC	3	18/9	–	–	–	–	–	None
CPI	Data	FE	2	7	x	x	x	x	x	All
CPO	Label	E4	3	18/9	–	–	–	–	–	None
CZ	Label	CC	3	18/9	–	–	–	–	–	None
DAA		27	1	4	x	x	x	x	x	All
DAD	B	09	1	10	x	–	–	–	–	Only CY flag
DAD	D	19	1	10	x	–	–	–	–	Only CY flag
DAD	H	29	1	10	x	–	–	–	–	Only CY flag
DAD	SP	39	1	10	x	–	–	–	–	Only CY flag
DCR	A	3D	1	4	–	x	x	x	x	All Expt. CY
DCR	B	05	1	4	–	x	x	x	x	All Expt. CY
DCR	C	0D	1	4	–	x	x	x	x	All Expt. CY
DCR	D	15	1	4	–	x	x	x	x	All Expt. CY
DCR	E	1D	1	4	–	x	x	x	x	All Expt. CY
DCR	H	25	1	4	–	x	x	x	x	All Expt. CY
DCR	L	2D	1	4	–	x	x	x	x	All Expt. CY
DCR	M	35	1	10	–	x	x	x	x	All Expt. CY
DCX	B	0B	1	6	–	–	–	–	–	None
DCX	D	1B	1	6	–	–	–	–	–	None

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
DCX	H	2B	1	6	–	–	–	–	–	None
DCX	SP	3B	1	6	–	–	–	–	–	None
DI		F3	1	4	–	–	–	–	–	None
EI		FB	1	4	–	–	–	–	–	None
HLT		76	1	5	–	–	–	–	–	None
IN	Port Addr.	DB	2	10	–	–	–	–	–	None
INR	A	3C	1	4	–	x	x	x	x	All Expt. CY
INR	B	04	1	4	–	x	x	x	x	All Expt. CY
INR	C	0C	1	4	–	x	x	x	x	All Expt. CY
INR	D	14	1	4	–	x	x	x	x	All Expt. CY
INR	E	1C	1	4	–	x	x	x	x	All Expt. CY
INR	H	24	1	4	–	x	x	x	x	All Expt. CY
INR	L	2C	1	4	–	x	x	x	x	All Expt. CY
INR	M	34	1	10	–	x	x	x	x	All Expt. CY
INX	B	03	1	6	–	–	–	–	–	None
INX	D	13	1	6	–	–	–	–	–	None
INX	H	23	1	6	–	–	–	–	–	None
INX	SP	33	1	6	–	–	–	–	–	None
JC	Label	DA	3	10/7	–	–	–	–	–	None
JM	Label	FA	3	10/7	–	–	–	–	–	None
JMP	Label	C3	3	10	–	–	–	–	–	None
JNC	Label	D2	3	10/7	–	–	–	–	–	None

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
JNZ	Label	C2	3	10/7	–	–	–	–	–	None
JP	Label	F2	3	10/7	–	–	–	–	–	None
JPE	Label	EA	3	10/7	–	–	–	–	–	None
JPO	Label	E2	3	10/7	–	–	–	–	–	None
JZ	Label	CA	3	10/7	–	–	–	–	–	None
LDA	Addr.	3A	3	13	–	–	–	–	–	None
LDAX	B	0A	1	7	–	–	–	–	–	None
LDAX	D	1A	1	7	–	–	–	–	–	None
LHLD	Addr.	2A	3	16	–	–	–	–	–	None
LXI	B	01	3	10	–	–	–	–	–	None
LXI	D	11	3	10	–	–	–	–	–	None
LXI	H	21	3	10	–	–	–	–	–	None
LXI	SP	31	3	10	–	–	–	–	–	None
MOV	A, A	7F	1	4	–	–	–	–	–	None
MOV	A, B	78	1	4	–	–	–	–	–	None
MOV	A, C	79	1	4	–	–	–	–	–	None
MOV	A, D	7A	1	4	–	–	–	–	–	None
MOV	A, E	7B	1	4	–	–	–	–	–	None
MOV	A, H	7C	1	4	–	–	–	–	–	None
MOV	A, L	7D	1	4	–	–	–	–	–	None
MOV	A, M	7E	1	7	–	–	–	–	–	None
MOV	B, A	47	1	4	–	–	–	–	–	None

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
MOV	B, B	40	1	4	–	–	–	–	–	None
MOV	B, C	41	1	4	–	–	–	–	–	None
MOV	B, D	42	1	4	–	–	–	–	–	None
MOV	B, E	43	1	4	–	–	–	–	–	None
MOV	B, H	44	1	4	–	–	–	–	–	None
MOV	B, L	45	1	4	–	–	–	–	–	None
MOV	B, M	46	1	7	–	–	–	–	–	None
MOV	C, A	4F	1	4	–	–	–	–	–	None
MOV	C, B	48	1	4	–	–	–	–	–	None
MOV	C, C	49	1	4	–	–	–	–	–	None
MOV	C, D	4A	1	4	–	–	–	–	–	None
MOV	C, E	4B	1	4	–	–	–	–	–	None
MOV	C, H	4C	1	4	–	–	–	–	–	None
MOV	C, L	4D	1	4	–	–	–	–	–	None
MOV	C, M	4E	1	7	–	–	–	–	–	None
MOV	D, A	57	1	4	–	–	–	–	–	None
MOV	D, B	50	1	4	–	–	–	–	–	None
MOV	D, C	51	1	4	–	–	–	–	–	None
MOV	D, D	52	1	4	–	–	–	–	–	None
MOV	D, E	53	1	4	–	–	–	–	–	None
MOV	D, H	54	1	4	–	–	–	–	–	None
MOV	D, L	55	1	4	–	–	–	–	–	None

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
MOV	D, M	56	1	7	–	–	–	–	–	None
MOV	E, A	5F	1	4	–	–	–	–	–	None
MOV	E, B	58	1	4	–	–	–	–	–	None
MOV	E, C	59	1	4	–	–	–	–	–	None
MOV	E, D	5A	1	4	–	–	–	–	–	None
MOV	E, E	5B	1	4	–	–	–	–	–	None
MOV	E, H	5C	1	7	–	–	–	–	–	None
MOV	E, L	5D	1	4	–	–	–	–	–	None
MOV	E, M	5E	1	7	–	–	–	–	–	None
MOV	H, A	67	1	4	–	–	–	–	–	None
MOV	H, B	60	1	4	–	–	–	–	–	None
MOV	H, C	61	1	4	–	–	–	–	–	None
MOV	H, D	62	1	4	–	–	–	–	–	None
MOV	H, E	63	1	4	–	–	–	–	–	None
MOV	H, H	64	1	4	–	–	–	–	–	None
MOV	H, L	65	1	4	–	–	–	–	–	None
MOV	H, M	66	1	7	–	–	–	–	–	None
MOV	L, A	6F	1	4	–	–	–	–	–	None
MOV	L, B	68	1	4	–	–	–	–	–	None
MOV	L, C	69	1	4	–	–	–	–	–	None
MOV	L, D	6A	1	4	–	–	–	–	–	None
MOV	L, E	6B	1	4	–	–	–	–	–	None

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
MOV	L, H	6C	1	4	–	–	–	–	–	None
MOV	L, L	6D	1	4	–	–	–	–	–	None
MOV	L, M	6E	1	7	–	–	–	–	–	None
MOV	M, A	77	1	7	–	–	–	–	–	None
MOV	M, B	70	1	7	–	–	–	–	–	None
MOV	M, C	71	1	7	–	–	–	–	–	None
MOV	M, D	72	1	7	–	–	–	–	–	None
MOV	M, E	73	1	7	–	–	–	–	–	None
MOV	M, H	74	1	7	–	–	–	–	–	None
MOV	M, L	75	1	7	–	–	–	–	–	None
MVI	A, data	3E	2	7	–	–	–	–	–	None
MVI	B, data	06	2	7	–	–	–	–	–	None
MVI	C, data	0E	2	7	–	–	–	–	–	None
MVI	D, data	16	2	7	–	–	–	–	–	None
MVI	E, data	1E	2	7	–	–	–	–	–	None
MVI	H, data	26	2	7	–	–	–	–	–	None
MVI	L, data	2E	2	7	–	–	–	–	–	None
MVI	M, data	36	2	10	–	–	–	–	–	None
NOP		00	1	4	–	–	–	–	–	None
ORA	A	B7	1	4	0	0	x	x	x	CY is 0 AC is 0
ORA	B	B0	1	4	0	0	x	x	x	CY is 0 AC is 0
ORA	C	B1	1	4	0	0	x	x	x	CY is 0 AC is 0

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
ORA	D	B2	1	4	0	0	x	x	x	CY is 0 AC is 0
ORA	E	B3	1	4	0	0	x	x	x	CY is 0 AC is 0
ORA	H	B4	1	4	0	0	x	x	x	CY is 0 AC is 0
ORA	L	B5	1	4	0	0	x	x	x	CY is 0 AC is 0
ORA	M	B6	1	7	0	0	x	x	x	CY is 0 AC is 0
ORI	Data	F6	2	7	0	0	x	x	x	CY is 0 AC is 0
OUT	Port Addr.	D3	2	10	–	–	–	–	–	None
PCHL		E9	1	6	–	–	–	–	–	None
POP	B	C1	1	10	–	–	–	–	–	None
POP	D	D1	1	10	–	–	–	–	–	None
POP	H	E1	1	10	–	–	–	–	–	None
POP	PSW	F1	1	10	x	x	x	x	x	All
PUSH	B	C5	1	12	–	–	–	–	–	None
PUSH	D	D5	1	12	–	–	–	–	–	None
PUSH	H	E5	1	12	–	–	–	–	–	None
PUSH	PSW	F5	1	12	x	x	x	x	x	All
RAL		17	1	4	x	–	–	–	–	Only CY
RAR		1F	1	4	x	–	–	–	–	Only CY
RC		D8	1	12/6	–	–	–	–	–	None
RET		C9	1	10	–	–	–	–	–	None
RIM		20	1	4	–	–	–	–	–	None
RLC		07	1	4	x	–	–	–	–	Only CY

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
RM		F8	1	12/6	–	–	–	–	–	None
RNC		D0	1	12/6	–	–	–	–	–	None
RNZ		C0	1	12/6	–	–	–	–	–	None
RP		F0	1	12/6	–	–	–	–	–	None
RPE		E8	1	12/6	–	–	–	–	–	None
RPO		E0	1	12/6	–	–	–	–	–	None
RRC		0F	1	4	x	–	–	–	–	Only CY
RST	0	C7	1	12	–	–	–	–	–	None
RST	1	CF	1	12	–	–	–	–	–	None
RST	2	D7	1	12	–	–	–	–	–	None
RST	3	DF	1	12	–	–	–	–	–	None
RST	4	E7	1	12	–	–	–	–	–	None
RST	5	EF	1	12	–	–	–	–	–	None
RST	6	F7	1	12	–	–	–	–	–	None
RST	7	FF	1	12	–	–	–	–	–	None
RZ		C8	1	12/6	–	–	–	–	–	None
SBB	A	9F	1	4	x	x	x	x	x	All
SBB	B	98	1	4	x	x	x	x	x	All
SBB	C	99	1	4	x	x	x	x	x	All
SBB	D	9A	1	4	x	x	x	x	x	All
SBB	E	9B	1	4	x	x	x	x	x	All
SBB	H	9C	1	4	x	x	x	x	x	All

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
SBB	L	9D	1	4	x	x	x	x	x	All
SBB	M	9E	1	7	x	x	x	x	x	All
SBI	Data	DE	2	7	x	x	x	x	x	All
SHLD	Addr.	22	3	16	–	–	–	–	–	None
SIM		30	1	4	–	–	–	–	–	None
SPHL		F9	1	6	–	–	–	–	–	None
STA	Addr.	32	3	13	–	–	–	–	–	None
STAX	B	02	1	7	–	–	–	–	–	None
STAX	D	12	1	7	–	–	–	–	–	None
STC		37	1	4	x	–	–	–	–	Only CY
SUB	A	97	1	4	x	x	x	x	x	All
SUB	B	90	1	4	x	x	x	x	x	All
SUB	C	91	1	4	x	x	x	x	x	All
SUB	D	92	1	4	x	x	x	x	x	All
SUB	E	93	1	4	x	x	x	x	x	All
SUB	H	94	1	4	x	x	x	x	x	All
SUB	L	95	1	4	x	x	x	x	x	All
SUB	M	96	1	7	x	x	x	x	x	All
SUI	Data	D6	2	7	x	x	x	x	x	All
XCHG		EB	1	4	–	–	–	–	–	None
XRA	A	AF	1	4	0	0	x	x	x	CY & AC Zero
XRA	B	A8	1	4	0	0	x	x	x	CY & AC Zero

Mne-monics	Ope-rand	Op code	Bytes	T-States	Status CY	AC	Z	S	P	Flags affected
XRA	C	A9	1	4	0	0	x	x	x	CY & AC Zero
XRA	D	AA	1	4	0	0	x	x	x	CY & AC Zero
XRA	E	AB	1	4	0	0	x	x	x	CY & AC Zero
XRA	H	AC	1	4	0	0	x	x	x	CY & AC Zero
XRA	L	AD	1	4	0	0	x	x	x	CY & AC Zero
XRA	M	AE	1	7	0	0	x	x	x	CY & AC Zero
XRI	Data	EE	2	7	0	0	x	x	x	CY & AC Zero
XTHL		E3	1	16	–	–	–	–	–	None

Appendix - II

Instruction codes of 8085 Microprocessor in Hexadecimal Order

Hex Code	Mnemonics	Hex Code	Mnemonics
00	NOP	27	DAA
01	LXI B	28	---
02	STAX B	29	DAD H
03	INX B	2A	LHLD
04	INR B	2B	DCX H
05	DCR B	2C	INR L
06	MVI B	2D	DCR L
07	RLC	2E	MVI L
08	---	2F	CMA
09	DAD B	30	SIM
0A	LDAX B	31	LXI SP
0B	DCX B	32	STA
0C	INR C	33	INX SP
0D	DCR C	34	INR M
0E	MVI C	35	DCR M
0F	RRC	36	MVI M
10	---	37	STC
11	LXI D	38	---
12	STAX D	39	DAD SP
13	INX D	3A	LDA
14	INR D	3B	DCX SP
15	DCR D	3C	INR A
16	MVI D	3D	DCR A
17	RAL	3E	MVI A
18	---	3F	CMC
19	DAD D	40	MOV B, B
1A	LDAX D	41	MOV B, C
1B	DCX D	42	MOV B, D
1C	INR E	43	MOV B, E
1D	DCR E	44	MOV B, H
1E	MVI E	45	MOV B, L
1F	RAR	46	MOV B, M
20	RIM	47	MOV B, A
21	LXI H	48	MOV C, B
22	SHLD	49	MOV C, C
23	INX H	4A	MOV C, D
24	INR H	4B	MOV C, E
25	DCR H	4C	MOV C, H

26	MVI H	4D	MOV C, L

Hex Code	Mnemonics	Hex Code	Mnemonics
4E	MOV C, M	75	MOV M, L
4F	MOV C, A	76	HLT
50	MOV D, B	77	MOV M, A
51	MOV D, C	78	MOV A, B
52	MOV D, D	79	MOV A, C
53	MOV D, E	7A	MOV A, D
54	MOV D, H	7B	MOV A, E
55	MOV D, L	7C	MOV A, H
56	MOV D, M	7D	MOV A, L
57	MOV D, A	7E	MOV A, M
58	MOV E, B	7F	MOV A, A
59	MOV E, C	80	ADD B
5A	MOV E, D	81	ADD C
5B	MOV E, E	82	ADD D
5C	MOV E, H	83	ADD E
5D	MOV E, L	84	ADD H
5E	MOV E, M	85	ADD L
5F	MOV E, A	86	ADD M
60	MOV H, B	87	ADD A
61	MOV H, C	88	ADC B
62	MOV H, D	89	ADC C
63	MOV H, E	8A	ADC D
64	MOV H, H	8B	ADC E
65	MOV H, L	8C	ADC H
66	MOV H, M	8D	ADC L
67	MOV H, A	8E	ADC M
68	MOV L, B	8F	ADC A
69	MOV L, C	90	SUB B
6A	MOV L, D	91	SUB C
6B	MOV L, E	92	SUB D
6C	MOV L, H	93	SUB E
6D	MOV L, L	94	SUB H
6E	MOV L, M	95	SUB L
6F	MOV L, A	96	SUB M
70	MOV M, B	97	SUB A
71	MOV M, C	98	SBB B
72	MOV M, D	99	SBB C
73	MOV M, E	9A	SBB D
74	MOV M, H	9B	SBB E

Hex Code	Mnemonics	Hex Code	Mnemonics
9C	SBB H	C3	JMP
9D	SBB L	C4	CNZ
9E	SBB M	C5	PUSH B
9F	SBB A	C6	ADI
A0	ANA B	C7	RST 0
A1	ANA C	C8	RZ
A2	ANA D	C9	RET
A3	ADA E	CA	JZ
A4	ANA H	CB	---
A5	ANA L	CC	CZ
A6	ANA M	CD	CALL
A7	ANA A	CE	ACI
A8	XRA B	CF	RST 1
A9	XRA C	D0	RNC
AA	XRA D	D1	POP D
AB	XRA E	D2	JNC
AC	XRA H	D3	OUT
AD	XRA L	D4	CNC
AE	XRA M	D5	PUSH D
AF	XRA A	D6	SUI
B0	ORA B	D7	RST 2
B1	ORA C	D8	RC
B2	ORA D	D9	---
B3	ORA E	DA	JC
B4	ORA H	DB	IN
B5	ORA L	DC	CC
B6	ORA M	DD	---
B7	ORA A	DE	SBI
B8	CMP B	DF	RST 3
B9	CMP C	E0	RPO
BA	CMP D	E1	POP H
BB	CMP E	E2	JPO
BC	CMP H	E3	XTHL
BD	CMP L	E4	CPO
BE	CMP M	E5	PUSH H
BF	CMP A	E6	ANI
C0	RNZ	E7	RST 4
C1	POP B	E8	RPE
C2	JNZ	E9	PCHL

Hex Code	Mnemonics	Hex Code	Mnemonics
EA	JPE	F5	PUSH PSW
EB	XCHG	F6	ORI
EC	CPE	F7	RST 6
ED	---	F8	RM
EE	XRI	F9	SPHL
EF	RST 5	FA	JM
F0	RP	FB	EI
F1	POP PSW	FC	CM
F2	JP	FD	---
F3	DI	FE	CPI
F4	CP	FF	RST 7

Total: 246 Instructions

Appendix - III

Instruction Set of 8085

Lower order Nibble →

Higher order Nibble ↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LXI B	STAX B	INX B	INR B	DCR B	MVI B	RLC	-	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C	RRC
1	-	LXI D	STAX D	INX D	INR D	DCR D	MVI D	RAL	-	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E	RAR
2	RIM	LXI H	SHLD	INX H	INR H	DCR H	MVI H	DAA	-	DAD H	LHLD	DCX H	INR L	DCR L	MVI L	CMA
3	SIM	LXI SP	STA	INX SP	INR M	DCR M	MVI M	STC	-	DAD SP	LDA	DCX SP	INR A	DCR A	MVI A	CMC
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,H	MOV A,L	MOV A,M	MOV A,A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC D	ADC E	ADC H	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ	JMP	CNZ	PUSH B	ADI	RST 0	RZ	RET	JZ	-	CZ	CALL	ACI	RST 1
D	RNC	POP D	JNC	OUT	CNC	PUSH D	SUI	RST 2	RC	-	JC	IN	CC	-	SBI	RST 3
E	RPO	POP H	JPO	XTHL	CPO	PUSH H	ANI	RST 4	RPE	PCHL	JPE	XCHG	CPE	-	XRI	RST 5
F	RP	POP PSW	JP	DI	CP	PUSH PSW	ORI	RST 6	RM	SPHL	JM	EI	CM	-	CPI	RST 7

CONCEPTS OF PROGRAMMING LANGUAGES

ROBERT W. SEBESTA

12/E

CONCEPTS OF PROGRAMMING LANGUAGES

ROBERT W. SEBESTA

12/E

CONCEPTS OF PROGRAMMING LANGUAGES

TWELFTH EDITION

CONCEPTS OF PROGRAMMING LANGUAGES

TWELFTH EDITION

ROBERT W. SEBESTA

University of Colorado at Colorado Springs

330 Hudson Street, NY NY 10013

Senior Vice President, Courseware Portfolio Management: *Marcia Horton*

Director, Portfolio Management: Engineering, Computer Science & Global Editions: *Julian Partridge*

Specialist, Higher Ed Portfolio Management: *Matt Goldstein*

Portfolio Management Assistant: *Meghan Jacoby*

Managing Content Producer: *Scott Disanno*

Content Producer: *Carole Snyder*

Web Developer: *Steve Wright*

Rights and Permissions Manager: *Ben Ferrini*

Manufacturing Buyer, Higher Ed, Lake Side Communications Inc (LSC): *Maura Zaldivar-Garcia*

Inventory Manager: *Ann Lam*

Product Marketing Manager: *Yvonne Vannatta*

Field Marketing Manager: *Demetrius Hall*

Marketing Assistant: *Jon Bryant*

Cover Designer: *Joyce Wells, jWellsDesign*

Full-Service Project Manager: *Prathiba Rajagopal, SPi Global*

Composition: *SPi Global*

Library of Congress Cataloging-in-Publication Data

Names: Sebesta, Robert W., author.

Title: Concepts of programming languages / Robert W. Sebesta, University of Colorado at Colorado Springs.

Description: Twelfth edition. | Pearson, [2019]

Identifiers: LCCN 2017059077| ISBN 9780134997186 (alk. paper) | ISBN 0134997182 (alk. paper)

Subjects: LCSH: Programming languages (Electronic computers)

Classification: LCC QA76.7 .S43 2019 | DDC 005.13--dc23 LC record available at https://lccn.loc.gov/2017059077

1 18

ISBN 10: 0-13-499718-2

ISBN 13: 978-0-13-499718-6

Changes for the Twelfth Edition of Concepts of Programming Languages

- **Chapter 2**: Added Section 2.16.4 A Replacement for Objective-C: Swift
    - Added Section 2.16.5 Another Related Language: Delphi
    - Deleted Section 2.18.6 Origins and Characteristics of Lua
- **Chapter 5**: Rewrote several paragraphs in Section 5.5.3 to correct and clarify
- **Chapter 6**: Added a paragraph to Section 6.3.2 to describe support for strings in Swift
    - Added a paragraph to Section 6.4.2 to describe support the enumeration types in Swift
    - Added a paragraph to Section 6.5.3 to describe support for arrays in Swift
    - Added a paragraph to Section 6.6.1 to describe support for associative arrays in Swift
    - Deleted the interview in Section 6.6.1
    - Added Section 6.12 Optional Types
- **Chapter 8**: Added a design issue and a brief discussion of it to Section 8.3.1.1
    - Added several paragraphs to Section 8.3.4 that describe iterators in

Python

- **Chapter 9**: Added a paragraph to Section 9.5.4 on Swift parameters
- **Chapter 11**: Deleted Section 11.4.2 (Abstract Data Types in Objective-C)
- **Chapter 12**: Deleted Section 12.4.5 (Objective-C)
    - Deleted Objective-C column from Table 12.1
    - Added a paragraph in the Summary on reflection

Preface

Changes for the Twelfth Edition

The goals, overall structure, and approach of this twelfth edition of *Concepts of Programming Languages* remains the same as those of the eleven previous editions. The principal goals are to introduce the fundamental constructs of contemporary programming languages and to provide the reader with the tools necessary for the critical evaluation of existing and future programming languages. A secondary goal is to prepare the reader for the study of compiler design, by providing an in-depth discussion of programming language structures, presenting a formal method of describing syntax, and introducing approaches to lexical and syntax analysis.

The twelfth edition evolved from the eleventh through several different kinds of changes. To maintain the currency of the material, nearly all discussion of some programming languages, specifically Lua and Objective-C, has been removed. Material on the newer language, Swift, was added to several chapters.

In addition, a new section on optional types was added to Chapter 6. Material was added to Section 8.3.4 to describe iterators in Python. In numerous places in the manuscript small changes were made to correct and/or clarify the discussion.

The Vision

This book describes the fundamental concepts of programming languages by discussing the design issues of the various language constructs, examining the design choices for these constructs in some of the most common languages, and critically comparing design alternatives.

Any serious study of programming languages requires an examination of some related topics, among which are formal methods of describing the syntax and semantics of programming languages, which are covered in Chapter 3. Also, implementation techniques for various language constructs must be considered: Lexical and syntax analysis are discussed in Chapter 4, and implementation of subprogram linkage is covered in Chapter 10. Implementation of some other language constructs is discussed in various other parts of the book.

The following paragraphs outline the contents of the twelfth edition.

Chapter Outlines

Chapter 1 begins with a rationale for studying programming languages. It then discusses the criteria used for evaluating programming languages and language constructs. The primary influences on language design, common design trade-offs, and the basic approaches to implementation are also examined.

Chapter 2 outlines the evolution of the languages that are discussed in this book. Although no attempt is made to describe any language completely, the origins, purposes, and contributions of each are discussed. This historical overview is valuable, because it provides the background necessary to understanding the practical and theoretical basis for contemporary language design. It also motivates further study of language design and evaluation. Because none of the remainder of the book depends on Chapter 2, it can be read on its own, independent of the other chapters.

Chapter 3 describes the primary formal method for describing the syntax of programming language—BNF. This is followed by a description of attribute grammars, which describe both the syntax and static semantics of languages. The difficult task of semantic description is then explored, including brief introductions to the three most common methods: operational, denotational, and axiomatic semantics.

Chapter 4 introduces lexical and syntax analysis. This chapter is targeted to those Computer Science departments that no longer require a compiler design course in their curricula. Similar to Chapter 2, this chapter stands alone and can be studied independently of the rest of the book, except for Chapter 3, on which it depends.

Chapters 5 through 14 describe in detail the design issues for the primary constructs of programming languages. In each case, the design choices for several example languages are presented and evaluated. Specifically, Chapter 5 covers the many characteristics of variables, Chapter 6 covers data types, and Chapter 7 explains expressions and assignment statements. Chapter 8

describes control statements, and Chapters 9 and 10 discuss subprograms and their implementation. Chapter 11 examines data abstraction facilities. Chapter 12 provides an in-depth discussion of language features that support object-oriented programming (inheritance and dynamic method binding), Chapter 13 discusses concurrent program units, and Chapter 14 is about exception handling, along with a brief discussion of event handling.

Chapters 15 and 16 describe two of the most important alternative programming paradigms: functional programming and logic programming. However, some of the data structures and control constructs of functional programming languages are discussed in Chapters 6 and 8. Chapter 15 presents an introduction to Scheme, including descriptions of some of its primitive functions, special forms, and functional forms, as well as some examples of simple functions written in Scheme. Brief introductions to ML, Haskell, and F# are given to illustrate some different directions in functional language design. Chapter 16 introduces logic programming and the logic programming language, Prolog.

To the Instructor

Chapters 1 and 3 are typically covered in detail, and though students find it interesting and beneficial reading, Chapter 2 receives little lecture time due to its lack of hard technical content. Because no material in subsequent chapters depends on Chapter 2, as noted earlier, it can be skipped entirely. If a course in compiler design is required, Chapter 4 is not covered.

Chapters 5 through 9 should be relatively easy for students with extensive programming experience in C++, Java, or C#. Chapters 10 through 14 are more challenging and require more detailed lectures.

Chapters 15 and 16 are entirely new to most students at the junior level. Ideally, language processors for Scheme and Prolog should be available for students required to learn the material in these chapters. Sufficient material is included to allow students to dabble with some simple programs.

Undergraduate courses will probably not be able to cover all of the material

in the last two chapters. Graduate courses, however, should be able to completely discuss the material in those chapters by skipping over some parts of the early chapters on imperative languages.

Supplemental Materials

The following supplements are available to all readers of this book at *www.pearson.com/cs-resources*.

- A set of lecture note slides. PowerPoint slides are available for each chapter in the book.
- All of the figures from the book.

A companion Web site to the book is available at *www.pearson.com/cs-resources*. This site contains mini-manuals (approximately 100-page tutorials) on a handful of languages.

Solutions to many of the problem sets are available to qualified instructors in our Instructor Resource Center at *www.pearson.com*. Please contact your school's Pearson representative or visit *www.pearson.com* to register.

Language Processor Availability

Processors for and information about some of the programming languages discussed in this book can be found at the following Web sites:

C, C++, Fortran, and Ada	*gcc.gnu.org*
C# and F#	*microsoft.com*
Java	*java.sun.com*
Haskell	*haskell.org*
Scheme	*www.plt-scheme.org/software/drscheme*
Perl	*www.perl.com*
Python	*www.python.org*
Ruby	*www.ruby-lang.org*

JavaScript is included in virtually all browsers; PHP is included in virtually all Web servers.

All this information is also included on the companion Web site.

Acknowledgments

The suggestions from outstanding reviewers contributed greatly to this book’s present form and contents. In alphabetical order, they are:

Aaron Rababaah	*University of Maryland at Eastern Shore*
Amar Raheja	*California State Polytechnic University–Pomona*
Amer Diwan	*University of Colorado*
Bob Neufeld	*Wichita State University*
Bruce R. Maxim	*University of Michigan–Dearborn*
Charles Nicholas	*University of Maryland–Baltimore County*
Cristian Videira Lopes	*University of California–Irvine*
Curtis Meadow	*University of Maine*
David E. Goldschmidt	
Donald Kraft	*Louisiana State University*
Duane J. Jarc	*University of Maryland, University College*
Euripides Montagne	*University of Central Florida*
Frank J. Mitropoulos	*Nova Southeastern University*
Gloria Melara	*California State University–Northbridge*
Hossein Saiedian	*University of Kansas*
I-ping Chu	*DePaul University*
Ian Barland	*Radford University*
K. N. King	*Georgia State University*
Karina Assiter	*Wentworth Institute of Technology*
Mark Llewellyn	*University of Central Florida*
Matthew Michael Burke	
Michael Prentice	*SUNY Buffalo*
Nancy Tinkham	*Rowan University*
Neelam Soundarajan	*Ohio State University*
Nigel Gwee	*Southern University–Baton Rouge*
Pamela Cutter	*Kalamazoo College*
Paul M. Jackowitz	*University of Scranton*
Paul Tymann	*Rochester Institute of Technology*
Richard M. Osborne	*University of Colorado–Denver*
Richard Min	*University of Texas at Dallas*
Robert McCloskey	*University of Scranton*
Ryan Stansifer	*Florida Institute of Technology*
Salih Yurttas	*Texas A&M University*
Saverio Perugini	*University of Dayton*
Serita Nelesen	*Calvin College*
Simon H. Lin	*California State University–Northbridge*
Stephen Edwards	*Virginia Tech*
Stuart C. Shapiro	*SUNY Buffalo*
Sumanth Yenduri	*University of Southern Mississippi*
Teresa Cole	*Boise State University*
Thomas Turner	*University of Central Oklahoma*
Tim R. Norton	*University of Colorado–Colorado Springs*
Timothy Henry	*University of Rhode Island*
Walter Pharr	*College of Charleston*
Xiangyan Zeng	*Fort Valley State University*

Numerous other people provided input for the previous editions of *Concepts of Programming Languages* at various stages of its development. All of their comments were useful and greatly appreciated. In alphabetical order, they are: Vicki Allan, Henry Bauer, Carter Bays, Manuel E. Bermudez, Peter Brouwer, Margaret Burnett, Paosheng Chang, Liang Cheng, John Crenshaw, Charles Dana, Barbara Ann Griem, Mary Lou Haag, John V. Harrison, Eileen Head, Ralph C. Hilzer, Eric Joanis, Leon Jololian, Hikyoo Koh, Jiang B. Liu, Meiliu Lu, Jon Mauney, Robert McCoard, Dennis L. Mumaugh, Michael G. Murphy, Andrew Oldroyd, Young Park, Rebecca Parsons, Steve J. Phelps, Jeffery Popyack, Steven Rapkin, Hamilton Richard, Tom Sager, Raghvinder - Sangwan, Joseph Schell, Sibylle Schupp, Mary Louise Soffa, Neelam Soundarajan, Ryan Stansifer, Steve Stevenson, Virginia Teller, Yang Wang, John M. Weiss, Franck Xia, and Salih Yurnas.

Matt Goldstein, Portfolio Management Specialist; Meghan Jacoby, Portfolio Management Assistant; Managing Content Producer, Scott Disanno; and Prathiba Rajagopal, all deserve my gratitude for their efforts to produce the twelfth edition both quickly and carefully.

About the Author

Robert Sebesta is an Associate Professor Emeritus in the Computer Science Department at the University of Colorado–Colorado Springs. Professor Sebesta received a BS in applied mathematics from the University of Colorado in Boulder and MS and PhD degrees in computer science from Pennsylvania State University. He has taught computer science for more than 40 years. His professional interests are the design and evaluation of programming languages and Web programming.

Contents

CONCEPTS OF PROGRAMMING LANGUAGES

TWELFTH EDITION

1 Preliminaries

Before we begin discussing the concepts of programming languages, we must consider a few preliminaries. First, we explain some reasons why computer science students and professional software developers should study general approaches to language design and evaluation. This discussion is especially valuable for those who believe that a working knowledge of one or two programming languages is sufficient for computer scientists. Then, we briefly describe the major programming domains. Next, because the book evaluates language constructs and features, we present a list of criteria that can serve as a basis for such judgments. Then, we discuss the two major influences on language design: machine architecture and program design methodologies. After that, we introduce the major categories of programming languages. Next, we describe a few of the most important trade-offs that must be - considered during language design.

Because this book is also about the implementation of programming - languages, this chapter includes an overview of the most common general approaches to implementation. Finally, we briefly describe a few examples of programming environments and discuss their impact on software production.

1.1 Reasons for Studying Concepts of Programming Languages

It is natural for students to wonder how they will benefit from the study of programming language concepts. After all, many other topics in computer - science are worthy of serious study. In fact, many now believe that there are more important areas of computing for study than can be covered in a four-year college curriculum. The following is what we believe to be a compelling list of potential benefits of studying concepts of programming languages:

- Increased capacity to express ideas. It is widely believed that the depth at which people can think is influenced by the expressive power of the language in which they communicate their thoughts. Those with only a weak understanding of natural language are limited in the complexity of their thoughts, particularly in depth of abstraction. In other words, it is difficult for people to conceptualize structures they cannot describe, verbally or in writing.

  Programmers, in the process of developing software, are similarly constrained. The language in which they develop software places limits on the kinds of control structures, data structures, and abstractions they can use; thus, the forms of algorithms they can construct are likewise limited. Awareness of a wider variety of programming language features can reduce such limitations in software development. Programmers can increase the range of their software development thought processes by learning new language constructs.

  It might be argued that learning the capabilities of other languages does not help a programmer who is forced to use a language that lacks those capabilities. That argument does not hold up, however, because often, - language constructs can be simulated in other languages that do not support those constructs directly. For example, a C (Harbison and Steele, 2002) programmer who had learned the structure and uses of associative arrays in Perl (Christianson et al., 2013) might design

structures that simulate associative arrays in that language. In other words, the study of programming language concepts builds an appreciation for valuable language features and constructs and encourages programmers to use them, even when the language they are using does not directly support such features and constructs.

- Improved background for choosing appropriate languages. Some professional programmers have had little formal education in computer science; rather, they have developed their programming skills independently or through in-house training programs. Such training programs often limit instruction to one or two languages that are directly relevant to the current projects of the organization. Other programmers received their formal training years ago. The languages they learned then are no longer widely used, and many features now available in programming languages were not commonly known at the time. The result is that many programmers, when given a choice of languages for a new project, use the language with which they are most familiar, even if it is poorly suited for the project at hand. If these programmers were familiar with a wider range of languages and language constructs, they would be better able to choose the language with the features that best address the problem.

  Some of the features of one language often can be simulated in another language. However, it is preferable to use a feature whose design has been integrated into a language than to use a simulation of that feature, which is often less elegant, more cumbersome, and less safe.

- Increased ability to learn new languages. Computer programming is still a relatively young discipline, and design methodologies, software development tools, and programming languages are still in a state of continuous evolution. This makes software development an exciting profession, but it also means that continuous learning is essential. The process of learning a new programming language can be lengthy and difficult, especially for someone who is comfortable with only one or two languages and has never examined programming language concepts in general. Once a thorough understanding of the fundamental concepts of languages is acquired, it becomes far easier to see how these concepts

are incorporated into the design of the language being learned. For example, programmers who understand the concepts of object-oriented programming will have a much easier time learning Ruby (Thomas et al., 2013) than those who have never used those concepts.

The same phenomenon occurs in natural languages. The better you know the grammar of your native language, the easier it is to learn a second language. Furthermore, learning a second language has the benefit of teaching you more about your first language.

The TIOBE Programming Community issues an index (`http://www.tiobe.com/index.php/content/paperinfo/tpci/index.htm`) that is an indicator of the relative popularity of programming languages. For example, according to the index, Java, C, C++ (Lippman et al., 2012), and C# (Albahari and Abrahari, 2012) were the four most popular languages in use in February 2017.1 However, dozens of other languages were widely used at the time. The index data also show that the distribution of usage of programming languages is always changing. The number of languages in use and the dynamic nature of the statistics imply that every software developer must be prepared to learn different languages.

1. Note that this index is only one measure of the popularity of programming languages, and its accuracy is not universally accepted.

Finally, it is essential that practicing programmers know the vocabulary and fundamental concepts of programming languages so they can read and understand programming language descriptions and evaluations, as well as promotional literature for languages and compilers. These are the sources of information needed in order to choose and learn a language.

- Better understanding of the significance of implementation. In learning the concepts of programming languages, it is both interesting and necessary to touch on the implementation issues that affect those concepts. In some cases, an understanding of implementation issues leads to an understanding of why languages are designed the way they are. In turn, this knowledge leads to the ability to use a language more intelligently, as it was designed to be used. We can become better

programmers by understanding the choices among programming language constructs and the consequences of those choices.

Certain kinds of program bugs can be found and fixed only by a programmer who knows some related implementation details. Another benefit of understanding implementation issues is that it allows us to visualize how a computer executes various language constructs. In some cases, some knowledge of implementation issues provides hints about the relative efficiency of alternative constructs that may be chosen for a program. For example, programmers who know little about the complexity of the implementation of subprogram calls often do not realize that a small subprogram that is frequently called can be a highly inefficient design choice.

Because this book touches on only a few of the issues of implementation, the previous two paragraphs also serve well as rationale for studying compiler design.

- Better use of languages that are already known. Most contemporary programming languages are large and complex. Accordingly, it is uncommon for a programmer to be familiar with and use all of the features of a language he or she uses. By studying the concepts of programming languages, programmers can learn about previously unknown and unused parts of the languages they already use and begin to use those features.

- Overall advancement of computing. Finally, there is a global view of computing that can justify the study of programming language concepts. Although it is usually possible to determine why a particular programming language became popular, many believe, at least in retrospect, that the most popular languages are not always the best available. In some cases, it might be concluded that a language became widely used, at least in part, because those in positions to choose languages were not sufficiently familiar with programming language concepts.

  For example, many people believe it would have been better if ALGOL 60 (Backus et al., 1963) had displaced Fortran (ISO/IEC 1539-1, 2010)

in the early 1960s, because it was more elegant and had much better control statements, among other reasons. That it did not, is due partly to the programmers and software development managers of that time, many of whom did not clearly understand the conceptual design of ALGOL 60. They found its description difficult to read (which it was) and even more difficult to understand. They did not appreciate the benefits of block structure, recursion, and well-structured control statements, so they failed to see the benefits of ALGOL 60 over Fortran.

Of course, many other factors contributed to the lack of acceptance of ALGOL 60, as we will see in Chapter 2. However, the fact that computer users were generally unaware of the benefits of the language played a significant role.

In general, if those who choose languages were well informed, perhaps better languages would eventually squeeze out poorer ones.

1.2 Programming Domains

Computers have been applied to a myriad of different areas, from controlling nuclear power plants to providing video games in mobile phones. Because of this great diversity in computer use, programming languages with very different goals have been developed. In this section, we briefly discuss a few of the most common areas of computer applications and their associated languages.

1.2.1 Scientific Applications

The first digital computers, which appeared in the late 1940s and early 1950s, were invented and used for scientific applications. Typically, the scientific applications of that time used relatively simple data structures, but required large numbers of floating-point arithmetic computations. The most common data structures were arrays and matrices; the most common control structures were counting loops and selections. The early high-level programming languages invented for scientific applications were designed to provide for those needs. Their competition was assembly language, so efficiency was a primary concern. The first language for scientific applications was Fortran. ALGOL 60 and most of its descendants were also intended to be used in this area, although they were designed to be used in related areas as well. For some scientific applications where efficiency is the primary concern, such as those that were common in the 1950s and 1960s, no subsequent language is significantly better than Fortran, which explains why Fortran is still used.

1.2.2 Business Applications

The use of computers for business applications began in the 1950s. Special computers were developed for this purpose, along with special languages. The first successful high-level language for business was COBOL (ISO/IEC, 2002), the initial version of which appeared in 1960. It probably still is the

most commonly used language for these applications. Business languages are characterized by facilities for producing elaborate reports, precise ways of describing and storing decimal numbers and character data, and the ability to specify decimal arithmetic operations.

There have been few developments in business application languages outside the development and evolution of COBOL. Therefore, this book includes only limited discussions of the structures in COBOL.

1.2.3 Artificial Intelligence

Artificial intelligence (AI) is a broad area of computer applications characterized by the use of symbolic rather than numeric computations. Symbolic computation means that symbols, consisting of names rather than numbers, are manipulated. Also, symbolic computation is more conveniently done with linked lists of data rather than arrays. This kind of programming sometimes requires more flexibility than other programming domains. For example, in some AI applications the ability to create and execute code segments during execution is convenient.

The first widely used programming language developed for AI applications was the functional language Lisp (McCarthy et al., 1965), which appeared in 1959. Most AI applications developed prior to 1990 were written in Lisp or one of its close relatives. During the early 1970s, however, an alternative approach to some of these applications appeared—logic programming using the Prolog (Clocksin and Mellish, 2013) language. More recently, some AI applications have been written in systems languages such as Python (Lutz, 2013). Scheme (Dybvig, 2011), a dialect of Lisp, and Prolog are introduced in Chapters 15 and 16, respectively.

1.2.4 Web Software

The World Wide Web is supported by an eclectic collection of languages, ranging from markup languages, such as HTML, which is not a programming

language, to general-purpose programming languages, such as Java. Because of the pervasive need for dynamic Web content, some computation capability is often included in the technology of content presentation. This functionality can be provided by embedding programming code in an HTML document. Such code is often in the form of a scripting language, such as JavaScript (Flanagan, 2011) or PHP (Tatroe et al., 2013). There are also some markup-like languages that have been extended to include constructs that control document processing, which are discussed in Section 1.5 and in Chapter 2.

1.3 Language Evaluation Criteria

As noted previously, the purpose of this book is to examine carefully the underlying concepts of the various constructs and capabilities of programming languages. We will also evaluate these features, focusing on their impact on the software development process, including maintenance. To do this, we need a set of evaluation criteria. Such a list of criteria is necessarily controversial, because it is difficult to get even two computer scientists to agree on the value of some given language characteristic relative to others. In spite of these differences, most would agree that the criteria discussed in the following subsections are important.

Some of the characteristics that influence three of the four most important of these criteria are shown in Table 1.1, and the criteria themselves are discussed in the following sections.2 Note that only the most important characteristics are included in the table, mirroring the discussion in the following subsections. One could probably make the case that if one considered less important characteristics, virtually all table positions could include “bullets.”

2. The fourth primary criterion is cost, which is not included in the table because it is only slightly related to the other criteria and the characteristics that influence them.

Table 1.1 Language evaluation criteria and the characteristics that affect them

	CRITERIA		
Characteristic	READABILITY	WRITABILITY	RELIABILITY
Simplicity	•	•	•
Orthogonality	•	•	•
Data types	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

Note that some of these characteristics are broad and somewhat vague, such as writability, whereas others are specific language constructs, such as exception handling. Furthermore, although the discussion might seem to imply that the criteria have equal importance, that implication is not intended, and it is clearly not the case.

1.3.1 Readability

One of the most important criteria for judging a programming language is the ease with which programs can be read and understood. Before 1970, software development was largely thought of in terms of writing code. The primary positive characteristic of programming languages was efficiency. Language constructs were designed more from the point of view of the computer than of the computer users. In the 1970s, however, the software life-cycle concept (Booch, 1987) was developed; coding was relegated to a much smaller role, and maintenance was recognized as a major part of the cycle, particularly in terms of cost. Because ease of maintenance is determined in large part by the readability of programs, readability became an important measure of the quality of programs and programming languages. This was an important juncture in the evolution of programming languages. There was a distinct crossover from a focus on machine orientation to a focus on human orientation.

Readability must be considered in the context of the problem domain. For example, if a program that describes a computation is written in a language not designed for such use, the program may be unnatural and convoluted, making it unusually difficult to read.

The following subsections describe characteristics that contribute to the readability of a programming language.

1.3.1.1 Overall Simplicity

The overall simplicity of a programming language strongly affects its readability. A language with a large number of basic constructs is more difficult to learn than one with a smaller number. Programmers who must use a large language often learn a subset of the language and ignore its other features. This learning pattern is sometimes used to excuse the large number of language constructs, but that argument is not valid. Readability problems occur whenever the program's author has learned a different subset from that subset with which the reader is familiar.

A second complicating characteristic of a programming language is **feature multiplicity**—that is, having more than one way to accomplish a particular operation. For example, in Java, a user can increment a simple integer variable in four different ways:

```
count = count + 1
count += 1
count++
++count
```

Although the last two statements have slightly different meanings from each other and from the others in some contexts, all of them have the same meaning when used as stand-alone expressions. These variations are discussed in Chapter 7.

A third potential problem is **operator overloading**, in which a single operator symbol has more than one meaning. Although this is often useful, it can lead to reduced readability if users are allowed to create their own

overloading and do not do it sensibly. For example, it is clearly acceptable to overload + to use it for both integer and floating-point addition. In fact, this overloading simplifies a language by reducing the number of operators. However, suppose the programmer defined + used between single-dimensioned array operands to mean the sum of all elements of both arrays. Because the usual meaning of vector addition is quite different from this, this unusual meaning could confuse both the author and the program's readers. An even more extreme example of program confusion would be a user defining + between two vector operands to mean the difference between their respective first elements. Operator overloading is further discussed in Chapter 7.

Simplicity in languages can, of course, be carried too far. For example, the form and meaning of most assembly language statements are models of simplicity, as you can see when you consider the statements that appear in the next section. This very simplicity, however, makes assembly language programs less readable. Because they lack more complex control statements, program structure is less obvious; because the statements are simple, far more of them are required than in equivalent programs in a high-level language. These same arguments apply to the less extreme case of high-level languages with inadequate control and data-structuring constructs.

1.3.1.2 Orthogonality

Orthogonality in a programming language means that a relatively small set of primitive constructs can be combined in a relatively small number of ways to build the control and data structures of the language. Furthermore, every possible combination of primitives is legal and meaningful. For example, consider data types. Suppose a language has four primitive data types (integer, float, double, and character) and two type operators (array and pointer). If the two type operators can be applied to themselves and the four primitive data types, a large number of data structures can be defined.

The meaning of an orthogonal language feature is independent of the context of its appearance in a program. (The word *orthogonal* comes from the mathematical concept of orthogonal vectors, which are independent of each

other.) Orthogonality follows from a symmetry of relationships among primitives. A lack of orthogonality leads to exceptions to the rules of the language. For example, in a programming language that supports pointers, it should be possible to define a pointer to point to any specific type defined in the language. However, if pointers are not allowed to point to arrays, many potentially useful user-defined data structures cannot be defined.

We can illustrate the use of orthogonality as a design concept by comparing one aspect of the assembly languages of the IBM mainframe computers and the VAX series of minicomputers. We consider a single simple situation: adding two 32-bit integer values that reside in either memory or registers and replacing one of the two values with the sum. The IBM mainframes have two instructions for this purpose, which have the forms

```
A Reg1, memory_cell
AR Reg1, Reg2
```

where `Reg1` and `Reg2` represent registers. The semantics of these are

```
Reg1 ← contents(Reg1) + contents(memory_cell)
Reg1 ← contents(Reg1) + contents(Reg2)
```

The VAX addition instruction for 32-bit integer values is

```
ADDL operand_1, operand_2
```

whose semantics is

```
operand_2 ← contents(operand_1) + contents(operand_2)
```

In this case, either operand can be a register or a memory cell.

The VAX instruction design is orthogonal in that a single instruction can use either registers or memory cells as its operands. There are two ways to specify operands, which can be combined in all possible ways. The IBM design is not orthogonal. Only two out of four operand combinations possibilities are legal, and the two require different instructions, `A` and `AR`. The IBM design is more restricted and therefore less writable. For example, you cannot add two values and store the sum in a memory location. Furthermore, the IBM design is more difficult to learn because of the

restrictions and the additional instruction.

Orthogonality is closely related to simplicity: The more orthogonal the design of a language, the fewer exceptions the language rules require. Fewer exceptions mean a higher degree of regularity in the design, which makes the language easier to learn, read, and understand. Anyone who has learned a significant part of the English language can testify to the difficulty of learning its many rule exceptions (for example, *i* before *e* except after *c*).

As examples of the lack of orthogonality in a high-level language, consider the following rules and exceptions in C. Although C has two kinds of structured data types, arrays and records (**`struct`**s), records can be returned from functions but arrays cannot. A member of a structure can be any data type except **`void`** or a structure of the same type. An array element can be any data type except **`void`** or a function. Parameters are passed by value, unless they are arrays, in which case they are, in effect, passed by reference (because the appearance of an array name without a subscript in a C program is interpreted to be the address of the array's first element).

As an example of context dependence, consider the C expression

```
a + b
```

This expression often means that the values of `a` and `b` are fetched and added together. However, if `a` happens to be a pointer and `b` is an integer, it affects the value of `b`. For example, if `a` points to a float value that occupies four bytes, then the value of `b` must be scaled—in this case multiplied by 4—before it is added to `a`. Therefore, the type of `a` affects the treatment of the value of `b`. The context of `b` affects its meaning.

Too much orthogonality can also cause problems. Perhaps the most orthogonal programming language is ALGOL 68 (van Wijngaarden et al., 1969). Every language construct in ALGOL 68 has a type, and there are no restrictions on those types. In addition, most constructs produce values. This combinational freedom allows extremely complex constructs. For example, a conditional can appear as the left side of an assignment, along with declarations and other assorted statements, as long as the result is an address. This extreme form of orthogonality leads to unnecessary complexity.

Furthermore, because languages require a large number of primitives, a high degree of orthogonality results in an explosion of combinations. So, even if the combinations are simple, their sheer numbers lead to complexity.

Simplicity in a language, therefore, is at least in part the result of a combination of a relatively small number of primitive constructs and a limited use of the concept of orthogonality.

Some believe that functional languages offer a good combination of simplicity and orthogonality. A functional language, such as Lisp, is one in which computations are made primarily by applying functions to given parameters. In contrast, in imperative languages such as C, C++, and Java, computations are usually specified with variables and assignment statements. Functional languages offer potentially the greatest overall simplicity, because they can accomplish everything with a single construct, the function call, which can be combined simply with other function calls. This simple elegance is the reason why some language researchers are attracted to functional languages as the primary alternative to complex nonfunctional languages such as Java. Other factors, the most important of which is probably efficiency, however, have prevented functional languages from becoming more widely used.

1.3.1.3 Data Types

The presence of adequate facilities for defining data types and data structures in a language is another significant aid to readability. For example, suppose a numeric type is used for an indicator flag because there is no Boolean type in the language. In such a language, for example, in the original version of C, we might have an assignment such as the following:

```
timeout = 1
```

The meaning of this statement is unclear, whereas in a language that includes Boolean types, we would have the following:

```
timeout = true
```

The meaning of this statement is perfectly clear.

1.3.1.4 Syntax Design

The syntax, or form, of the elements of a language has a significant effect on the readability of programs. Following are some examples of syntactic design choices that affect readability:

- Special words. Program appearance and thus program readability are strongly influenced by the forms of a language's special words (for example, **`while`**, **`class`**, and **`for`**). Especially important is the method of forming compound statements, or statement groups, primarily in control constructs. Some languages have used matching pairs of special words or symbols to form groups. C and its descendants use braces to specify compound statements. All of these languages have diminished readability because statement groups are always terminated in the same way, which makes it difficult to determine which group is being ended when an **`end`** or a right brace appears. Fortran 95 and Ada (ISO/IEC, 2014) make this clearer by using a distinct closing syntax for each type of statement group. For example, Ada uses **`end if`** to terminate a selection construct and **`end loop`** to terminate a loop construct. This is an example of the conflict between simplicity that results in fewer reserved words, as in Java, and the greater readability that can result from using more reserved words, as in Ada.

  Another important issue is whether the special words of a language can be used as names for program variables. If so, then the resulting programs can be very confusing. For example, in Fortran 95, special words, such as `Do` and `End`, are legal variable names, so the appearance of these words in a program may or may not connote something special.

- Form and meaning. Designing statements so that their appearance at least partially indicates their purpose is an obvious aid to readability. Semantics, or meaning, should follow directly from syntax, or form. In some cases, this principle is violated by two language constructs that are identical or similar in appearance but have different meanings,

depending perhaps on context. In C, for example, the meaning of the reserved word **`static`** depends on the context of its appearance. If used on the definition of a variable inside a function, it means the variable is created at compile time. If used on the definition of a variable that is outside all functions, then it means the variable is visible only in the file in which its definition appears; that is, it is not exported from that file.

One of the primary complaints about the shell commands of UNIX (Robbins, 2005) is that their appearance does no t always suggest their function. For example, the meaning of the UNIX command `grep` can be deciphered only through prior knowledge, or perhaps cleverness and familiarity with the UNIX editor, `ed`. The appearance of `grep` connotes nothing to UNIX beginners. (In `ed`, the command */regular_expression*/ searches for a substring that matches the regular expression. Preceding this with `g` makes it a global command, specifying that the scope of the search is the whole file being edited. Following the command with `p` specifies that lines with the matching substring are to be printed. So `g`*/regular_expression*/`p`, which can obviously be abbreviated as `grep`, prints all lines in a file that contain substrings that match its operand, which is a regular expression.)

1.3.2 Writability

Writability is a measure of how easily a language can be used to create programs for a chosen problem domain. Most of the language characteristics that affect readability also affect writability. This follows directly from the fact that the process of writing a program requires the programmer frequently to reread the part of the program that is already written.

As is the case with readability, writability must be considered in the context of the target problem domain of a language. It simply is not fair to compare the writability of two languages in the realm of a particular application when one was designed for that application and the other was not. For example, the writabilities of Visual BASIC (VB) (Halvorson, 2013) and C are dramatically different for creating a program that has a graphical user interface (GUI), for which VB is ideal. Their writabilities are also quite different for writing

systems programs, such as an operating system, for which C was designed.

The following subsections describe the most important characteristics influencing the writability of a language.

1.3.2.1 Simplicity and Orthogonality

If a language has a large number of different constructs, some programmers who use the language might not be familiar with all of them. This situation can lead to a misuse of some features and a disuse of others that may be either more elegant or more efficient, or both, than those that are used. It may even be possible, as noted by Hoare (1973), to use unknown features accidentally, with bizarre results. Therefore, a smaller number of primitive constructs and a consistent set of rules for combining them (that is, orthogonality) is much better than simply having a large number of primitives. A programmer can design a solution to a complex problem after learning only a simple set of primitive constructs.

On the other hand, too much orthogonality can be a detriment to writability. Errors in programs can go undetected when nearly any combination of primitives is legal. This can lead to code absurdities that cannot be discovered by the compiler.

1.3.2.2 Expressivity

Expressivity in a language can refer to several different characteristics. In a language such as APL (Gilman and Rose, 1983), it means that there are very powerful operators that allow a great deal of computation to be accomplished with a very small program. More commonly, it means that a language has relatively convenient, rather than cumbersome, ways of specifying computations. For example, in C, the notation `count++` is more convenient and shorter than `count = count + 1`. Also, the **`and then`** Boolean operator in Ada is a convenient way of specifying short-circuit evaluation of a Boolean expression. The inclusion of the **`for`** statement in Java makes writing

counting loops easier than with the use of **`while`**, which is also possible. All of these increase the writability of a language.

1.3.3 Reliability

A program is said to be reliable if it performs to its specifications under all conditions. The following subsections describe several language features that have a significant effect on the reliability of programs in a given language.

1.3.3.1 Type Checking

Type checking is simply testing for type errors in a given program, either by the compiler or during program execution. Type checking is an important factor in language reliability. Because run-time type checking is expensive, compile-time type checking is more desirable. Furthermore, the earlier errors in programs are detected, the less expensive it is to make the required repairs. The design of Java requires checks of the types of nearly all variables and expressions at compile time. This virtually eliminates type errors at run time in Java programs. Types and type checking are discussed in depth in Chapter 6.

One example of how failure to type check, at either compile time or run time, has led to countless program errors is the use of subprogram parameters in the original C language (Kernighan and Ritchie, 1978). In this language, the type of an actual parameter in a function call was not checked to determine whether its type matched that of the corresponding formal parameter in the function. An **`int`** type variable could be used as an actual parameter in a call to a function that expected a **`float`** type as its formal parameter, and neither the compiler nor the run-time system would detect the inconsistency. For example, because the bit string that represents the integer 23 is essentially unrelated to the bit string that represents a floating-point 23, if an integer 23 is sent to a function that expects a floating-point parameter, any uses of the parameter in the function will produce nonsense. Furthermore, such problems are often difficult to diagnose.3 The current version of C has eliminated this

problem by requiring all parameters to be type checked. Subprograms and parameter-passing techniques are discussed in Chapter 9.

3. In response to this and other similar problems, UNIX systems include a utility program named `lint` that checks C programs for such problems.

1.3.3.2 Exception Handling

The ability of a program to intercept run-time errors (as well as other unusual conditions detectable by the program), take corrective measures, and then continue is an obvious aid to reliability. This language facility is called **exception handling**. Ada, C++, Java, and C# include extensive capabilities for exception handling, but such facilities are practically nonexistent in some widely used languages, for example, C. Exception handling is discussed in Chapter 14.

1.3.3.3 Aliasing

Loosely defined, **aliasing** is having two or more distinct names in a program that can be used to access the same memory cell. It is now generally accepted that aliasing is a dangerous feature in a programming language. Most programming languages allow some kind of aliasing—for example, two pointers (or references) set to point to the same variable, which is possible in most languages. In such a program, the programmer must always remember that changing the value pointed to by one of the two changes the value referenced by the other. Some kinds of aliasing, as described in Chapters 5 and 9, can be prohibited by the design of a language.

In some languages, aliasing is used to overcome deficiencies in the language's data abstraction facilities. Other languages greatly restrict aliasing to increase their reliability.

1.3.3.4 Readability and Writability

Both readability and writability influence reliability. A program written in a language that does not support natural ways to express the required algorithms will necessarily use unnatural approaches. Unnatural approaches are less likely to be correct for all possible situations. The easier a program is to write, the more likely it is to be correct.

Readability affects reliability in both the writing and maintenance phases of the life cycle. Programs that are difficult to read are difficult both to write and to modify.

1.3.4 Cost

The total cost of a programming language is a function of many of its characteristics.

First, there is the cost of training programmers to use the language, which is a function of the simplicity and orthogonality of the language and the experience of the programmers. Although more powerful languages are not necessarily more difficult to learn, they often are.

Second, there is the cost of writing programs in the language. This is a function of the writability of the language, which depends in part on its closeness in purpose to the particular application. The original efforts to design and implement high-level languages were driven by the desire to lower the costs of creating software.

Both the cost of training programmers and the cost of writing programs in a language can be significantly reduced in a good programming environment. Programming environments are discussed in Section 1.8.

Third, the cost of executing programs written in a language is greatly influenced by that language's design. A language that requires many run-time type checks will prohibit fast code execution, regardless of the quality of the compiler. Although execution efficiency was the foremost concern in the design of early languages, it is now considered to be less important.

A simple trade-off can be made between compilation cost and execution speed of the compiled code. **Optimization** is the name given to the collection of techniques that compilers may use to decrease the size and/or increase the execution speed of the code they produce. If little or no optimization is done, compilation can be done much faster than if a significant effort is made to produce optimized code. The choice between the two alternatives is influenced by the environment in which the compiler will be used. In a laboratory for beginning programming students, who often compile their programs many times during development but use little execution time (their programs are small and they must execute correctly only once), little or no optimization should be done. In a production environment, where compiled programs are executed many times after development, it is better to pay the extra cost to optimize the code.

Fourth, there is the cost of poor reliability. If the software fails in a critical system, such as a nuclear power plant or an X-ray machine for medical use, the cost could be very high. The failures of noncritical systems can also be very expensive in terms of lost future business or lawsuits over defective software systems.

The final consideration is the cost of maintaining programs, which includes both corrections and modifications to add new functionality. The cost of software maintenance depends on a number of language characteristics, primarily readability. Because maintenance is often done by individuals other than the original author of the software, poor readability can make the task extremely challenging.

The importance of software maintainability cannot be overstated. It has been estimated that for large software systems with relatively long lifetimes, maintenance costs can be as high as two to four times as much as development costs (Sommerville, 2010).

Of all the contributors to language costs, three are most important: program development, maintenance, and reliability. Because these are functions of writability and readability, these two evaluation criteria are, in turn, the most important.

Of course, a number of other criteria could be used for evaluating

programming languages. One example is **portability**, or the ease with which programs can be moved from one implementation to another. Portability is most strongly influenced by the degree of standardization of the language. Some languages are not standardized at all, making programs in these languages very difficult to move from one implementation to another. This problem is alleviated in some cases by the fact that implementations for some languages now have single sources. Standardization is a time-consuming and difficult process. A committee began work on producing a standard version of C++ in 1989. It was approved in 1998.

Generality (the applicability to a wide range of applications) and **well--definedness** (the completeness and precision of the language's official defining document) are two other criteria.

Most criteria, particularly readability, writability, and reliability, are neither precisely defined nor accurately measurable. Nevertheless, they are useful concepts and they provide valuable insight into the design and evaluation of programming languages.

A final note on evaluation criteria: language design criteria are weighed differently from different perspectives. Language implementors are concerned primarily with the difficulty of implementing the constructs and features of the language. Language users are worried about writability first and readability later. Language designers are likely to emphasize elegance and the ability to attract widespread use. These characteristics often conflict with one another.

1.4 Influences on Language Design

In addition to those factors described in Section 1.3, several other factors influence the basic design of programming languages. The most important of these are computer architecture and programming design methodologies.

1.4.1 Computer Architecture

The basic architecture of computers has had a profound effect on language design. Most of the popular languages of the past 60 years have been designed around the prevalent computer architecture, called the **von Neumann architecture**, after one of its originators, John von Neumann (pronounced “von Noyman”). These languages are called **imperative** languages. In a von Neumann computer, both data and programs are stored in the same memory. The central processing unit (CPU), which executes instructions, is separate from the memory. Therefore, instructions and data must be transmitted, or piped, from memory to the CPU. Results of operations in the CPU must be moved back to memory. Nearly all digital computers built since the 1940s have been based on the von Neumann architecture. The overall structure of a von Neumann computer is shown in Figure 1.1.

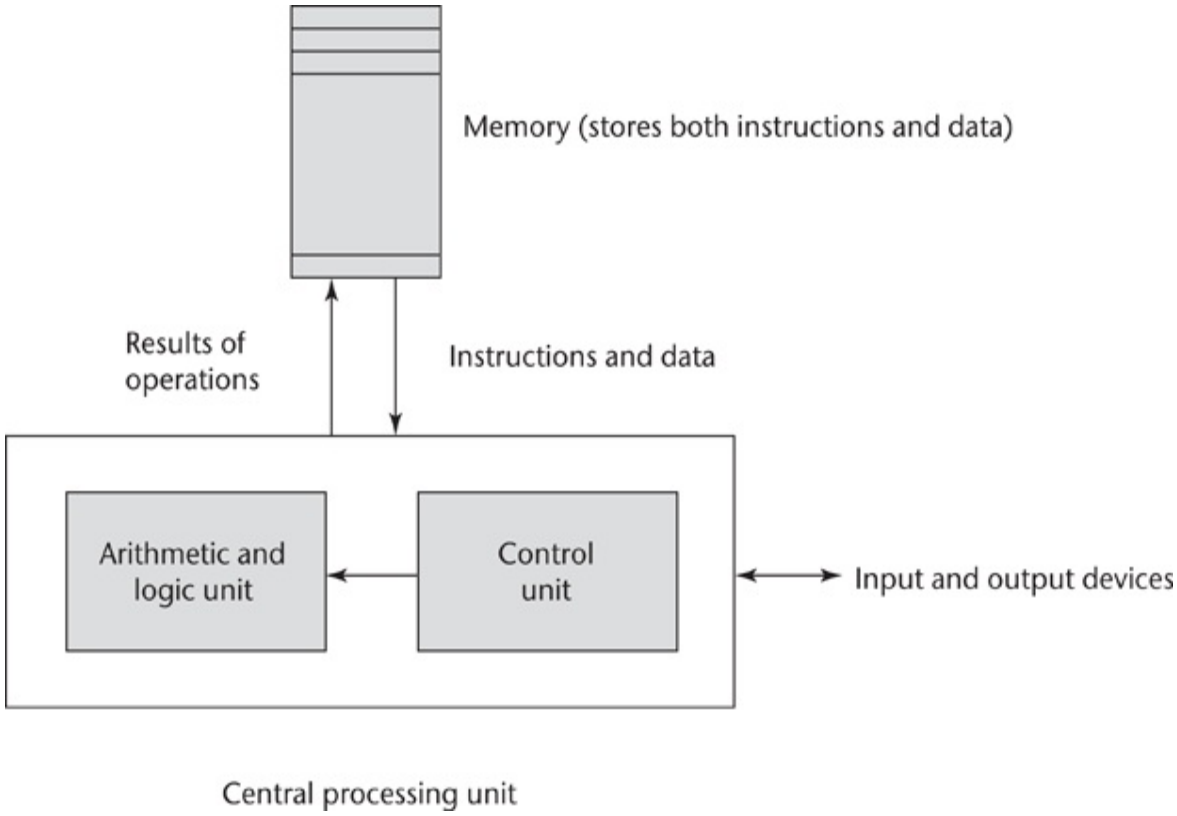


Figure 1.1 The von Neumann computer architecture

Figure 1.1 Full Alternative Text

Because of the von Neumann architecture, the central features of imperative languages are variables, which model the memory cells; assignment statements, which are based on the piping operation; and the iterative form of repetition, which is the most efficient way to implement repetition on this architecture. Operands in expressions are piped from memory to the CPU, and the result of evaluating the expression is piped back to the memory cell represented by the left side of the assignment. Iteration is fast on von Neumann computers because instructions are stored in adjacent cells of memory and repeating the execution of a section of code requires only a branch instruction. This efficiency discourages the use of recursion for repetition, although recursion is sometimes more natural.

The execution of a machine code program on a von Neumann architecture computer occurs in a process called the **fetch-execute cycle**. As stated earlier, programs reside in memory but are executed in the CPU. Each instruction to be executed must be moved from memory to the processor. The address of the next instruction to be executed is maintained in a register called the **program counter**. The fetch-execute cycle can be simply described by the following algorithm:

- initialize the program counter
- **`repeat`** forever
- fetch the instruction pointed to by the program counter
- increment the program counter to point at the next instruction
- decode the instruction
- execute the instruction
- **`end repeat`**

The “decode the instruction” step in the algorithm means the instruction is examined to determine what action it specifies. Program execution terminates when a stop instruction is encountered, although on an actual computer a stop instruction is rarely executed. Rather, control transfers from the operating system to a user program for its execution and then back to the operating system when the user program execution is complete. In a computer system in which more than one user program may be in memory at a given time, this process is far more complex.

As stated earlier, a functional, or applicative, language is one in which the primary means of computation is applying functions to given parameters. Programming can be done in a functional language without the kind of variables that are used in imperative languages, without assignment statements, and without iteration. Although many computer scientists have expounded on the myriad benefits of functional languages, such as Scheme, it is unlikely that they will displace the imperative languages until a non–von

Neumann computer is designed that allows efficient execution of programs in functional languages. Among those who have bemoaned this fact, perhaps the most eloquent was John Backus (1978), the principal designer of the original version of Fortran.

In spite of the fact that the structure of imperative programming languages is modeled on a machine architecture, rather than on the abilities and inclinations of the users of programming languages, some believe that using imperative languages is somehow more natural than using a functional language. So, these people believe that even if functional programs were as efficient as imperative programs, the use of imperative programming languages would still dominate.

1.4.2 Programming Design Methodologies

The late 1960s and early 1970s brought an intense analysis, begun in large part by the structured-programming movement, of both the software development process and programming language design.

An important reason for this research was the shift in the major cost of computing from hardware to software, as hardware costs decreased and programmer costs increased. Increases in programmer productivity were relatively small. In addition, progressively larger and more complex problems were being solved by computers. Rather than simply solving sets of equations to simulate satellite tracks, as in the early 1960s, programs were being written for large and complex tasks, such as controlling large petroleum-refining facilities and providing worldwide airline reservation systems.

The new software development methodologies that emerged as a result of the research of the 1970s were called top-down design and stepwise refinement. The primary programming language deficiencies that were discovered were incompleteness of type checking and inadequacy of control statements (requiring the extensive use of gotos).

In the late 1970s, a shift from procedure-oriented to data-oriented program design methodologies began. Simply put, data-oriented methods emphasize data design, focusing on the use of abstract data types to solve problems.

For data abstraction to be used effectively in software system design, it must be supported by the languages used for implementation. The first language to provide even limited support for data abstraction was SIMULA 67 (Birtwistle et al., 1973), although that language certainly was not propelled to popularity because of it. The benefits of data abstraction were not widely recognized until the early 1970s. However, most languages designed since the late 1970s support data abstraction, which is discussed in detail in Chapter 11.

The latest step in the evolution of data-oriented software development, which began in the early 1980s, is object-oriented design. Object-oriented methodology begins with data abstraction, which encapsulates processing with data objects and controls access to data, and adds inheritance and dynamic method binding. Inheritance is a powerful concept that greatly enhances the potential reuse of existing software, thereby providing the possibility of significant increases in software development productivity. This is an important factor in the increase in popularity of object-oriented languages. Dynamic (run-time) method binding allows more flexible use of inheritance.

Object-oriented programming developed along with a language that supported its concepts: Smalltalk (Goldberg and Robson, 1989). Although Smalltalk never became as widely used as many other languages, support for object-oriented programming is now part of most popular imperative languages, including Java, C++, and C#. Object-oriented concepts have also found their way into functional programming in CLOS (Bobrow et al., 1988) and F# (Syme et al., 2010), as well as logic programming in Prolog++ (Moss, 1994). Language support for object-oriented programming is discussed in detail in Chapter 12.

Procedure-oriented programming is, in a sense, the opposite of data-oriented programming. Although data-oriented methods now dominate software development, procedure-oriented methods have not been abandoned. On the contrary, in recent years, a good deal of research has occurred in procedure-oriented programming, especially in the area of concurrency. These research

efforts brought with them the need for language facilities for creating and controlling concurrent program units. Java and C# include such capabilities. Concurrency is discussed in detail in Chapter 13.

All of these evolutionary steps in software development methodologies led to new language constructs to support them.

1.5 Language Categories

Programming languages are often categorized into four bins: imperative, functional, logic, and object oriented. However, we do not consider languages that support object-oriented programming to form a separate category of languages. We have described how the most popular languages that support object-oriented programming grew out of imperative languages. Although the object-oriented software development paradigm differs significantly from the procedure-oriented paradigm usually used with imperative languages, the extensions to an imperative language required to support object-oriented programming are not intensive. For example, the expressions, assignment statements, and control statements of C and Java are nearly identical. (On the other hand, the arrays, subprograms, and semantics of Java are very different from those of C.) Similar statements can be made for functional languages that support object-oriented programming.

Some authors refer to scripting languages as a separate category of programming languages. However, languages in this category are bound together more by their implementation method, partial or full interpretation, than by a common language design. The languages that are typically called scripting languages, among them Perl, JavaScript, and Ruby (Flanagan and Matsumoto, 2008), are imperative languages in every sense.

A logic programming language is an example of a rule-based language. In an imperative language, an algorithm is specified in great detail, and the specific order of execution of the instructions or statements must be included. In a rule-based language, however, rules are specified in no particular order, and the language implementation system must choose an order in which the rules are used to produce the desired result. This approach to software development is radically different from those used with the other two categories of languages and clearly requires a completely different kind of language. Prolog, the most commonly used logic programming language, and logic programming are discussed in Chapter 16.

In recent years, a new category of languages has emerged, the

markup/programming hybrid languages. Markup languages are not programming languages. For instance, HTML, the most widely used markup language, is used to specify the layout of information in Web documents. However, some programming capability has crept into some extensions to HTML and XML. Among these are the Java Server Pages Standard Tag Library (JSTL) and eXtensible Stylesheet Language Transformations (XSLT). Both of these are briefly introduced in Chapter 2. Those languages cannot be compared to any of the complete programming languages and therefore will not be discussed after Chapter 2.

1.6 Language Design Trade-Offs

The programming language evaluation criteria described in Section 1.3 - provide a framework for language design. Unfortunately, that framework is self-contradictory. In his insightful paper on language design, Hoare (1973) stated that "there are so many important but conflicting criteria, that their reconciliation and satisfaction is a major engineering task."

Two criteria that conflict are reliability and cost of execution. For example, the Java language definition demands that all references to array elements be checked to ensure that the index or indices are in their legal ranges. This step adds a great deal to the cost of execution of Java programs that contain large numbers of references to array elements. C does not require index range checking, so C programs execute faster than semantically equivalent Java programs, although Java programs are more reliable. The designers of Java traded execution efficiency for reliability.

As another example of conflicting criteria that leads directly to design trade-offs, consider the case of APL. APL includes a powerful set of operators for array operands. Because of the large number of operators, a significant number of new symbols had to be included in APL to represent the operators. Also, many APL operators can be used in a single, long, complex expression. One result of this high degree of expressivity is that, for applications involving many array operations, APL is very writable. Indeed, a huge amount of computation can be specified in a very small program. Another result is that APL programs have very poor readability. A compact and concise expression has a certain mathematical beauty but it is difficult for anyone other than the programmer to understand. Well-known author Daniel McCracken (1970) once noted that it took him four hours to read and understand a four-line APL program. The designer of APL traded readability for writability.

The conflict between writability and reliability is a common one in language design. The pointers of C++ can be manipulated in a variety of ways, which supports highly flexible addressing of data. Because of the potential

reliability problems with pointers, they are not included in Java.

Examples of conflicts among language design (and evaluation) criteria abound; some are subtle, others are obvious. It is therefore clear that the task of choosing constructs and features when designing a programming language requires many compromises and trade-offs.

1.7 Implementation Methods

As described in Section 1.4.1, two of the primary components of a computer are its internal memory and its processor. The internal memory is used to store programs and data. The processor is a collection of circuits that provides a realization of a set of primitive operations, or machine instructions, such as those for arithmetic and logic operations. In most computers, some of these instructions, which are sometimes called macroinstructions, are actually implemented with a set of instructions called microinstructions, which are defined at an even lower level. Because microinstructions are never seen by software, they will not be discussed further here.

The machine language of the computer is its set of instructions. In the absence of other supporting software, its own machine language is the only language that most hardware computers "understand." Theoretically, a computer could be designed and built with a particular high-level language as its machine language, but it would be very complex and expensive. Furthermore, it would be highly inflexible, because it would be difficult (but not impossible) to use it with other high-level languages. The more practical machine design choice implements in hardware a very low-level language that provides the most commonly needed primitive operations and requires system software to create an interface to programs in higher-level languages.

A language implementation system cannot be the only software on a computer. Also required is a large collection of programs, called the operating system, which supplies higher-level primitives than those of the machine language. These primitives provide system resource management, input and output operations, a file management system, text and/or program editors, and a variety of other commonly needed functions. Because language implementation systems need many of the operating system facilities, they interface with the operating system rather than directly with the processor (in machine language).

The operating system and language implementations are layered over the

machine language interface of a computer. These layers can be thought of as virtual computers, providing interfaces to the user at higher levels. For example, an operating system and a C compiler provide a virtual C computer. With other compilers, a machine can become other kinds of virtual computers. Most computer systems provide several different virtual computers. User programs form another layer over the top of the layer of virtual computers. The layered view of a computer is shown in Figure 1.2.

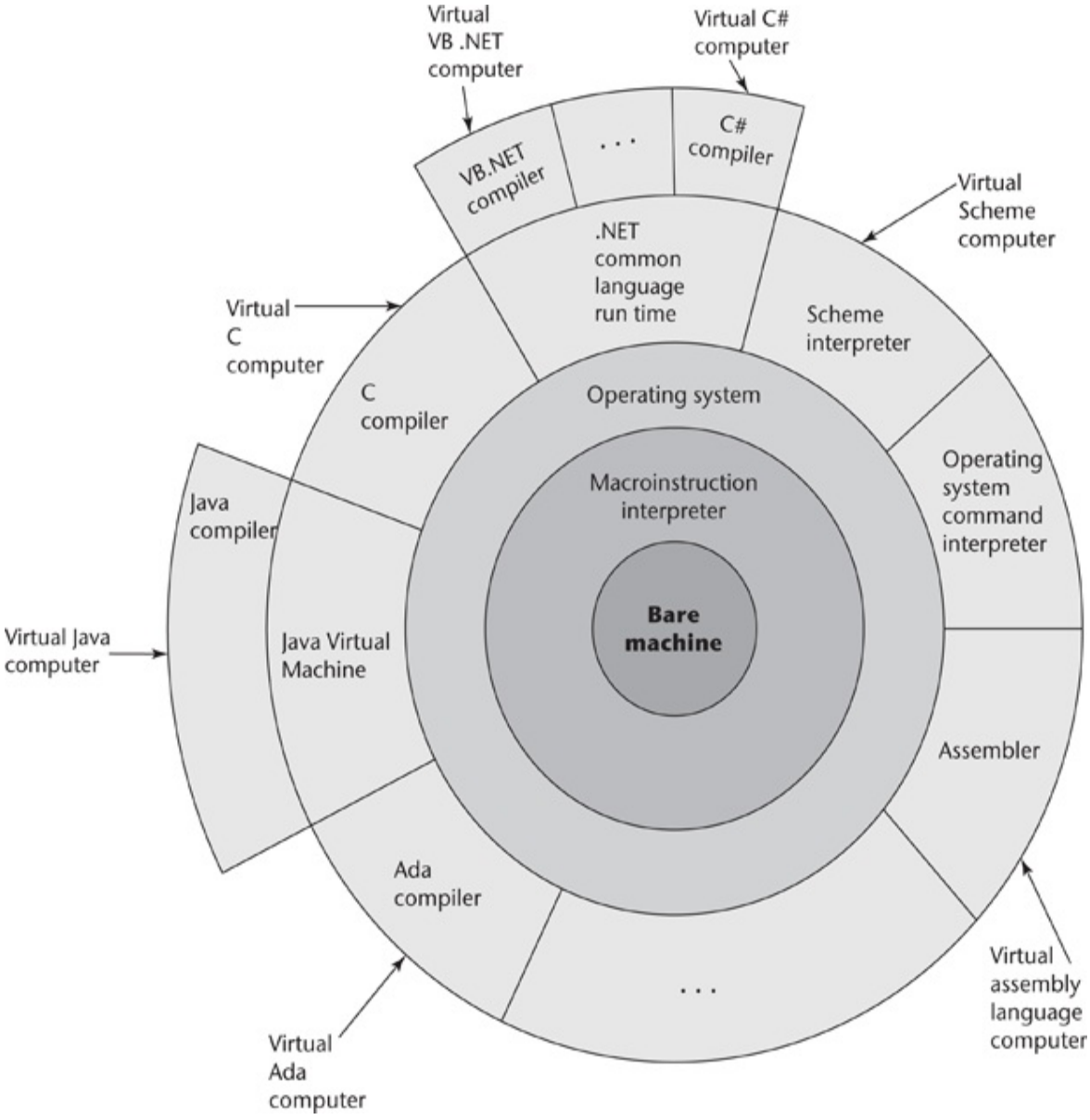


Figure 1.2 Layered interface of

virtual computers, provided by a typical computer system

Figure 1.2 Full Alternative Text

The implementation systems of the first high-level programming languages, constructed in the late 1950s, were among the most complex software systems of that time. In the 1960s, intensive research efforts were made to understand and formalize the process of constructing these high-level language implementations. The greatest success of those efforts was in the area of syntax analysis, primarily because that part of the implementation process is an application of parts of automata theory and formal language theory that were then well understood.

1.7.1 Compilation

Programming languages can be implemented by any of three general methods. At one extreme, programs can be translated into machine language, which can be executed directly on the computer. This method is called a **compiler implementation** and has the advantage of very fast program execution, once the translation process is complete. Most production implementations of languages, such as C, COBOL, and C++, are by compilers.

The language that a compiler translates is called the **source language**. The process of compilation and program execution takes place in several phases, the most important of which are shown in Figure 1.3.

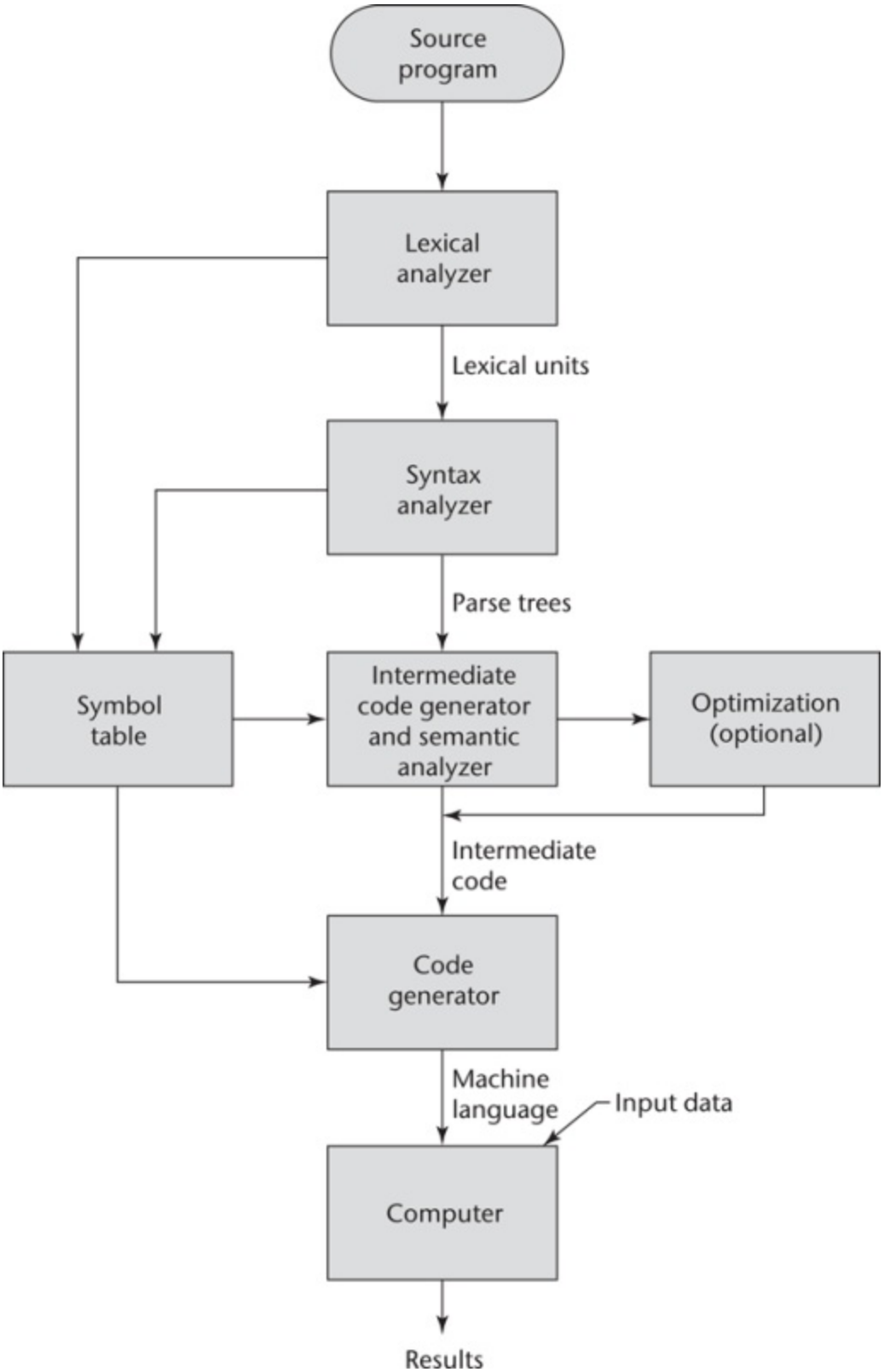

Source program
Lexical analyzer
Lexical units
Syntax analyzer
Parse trees
Symbol table
Intermediate code generator and semantic analyzer
Optimization (optional)
Intermediate code
Code generator
Machine language
Input data
Computer
Results

Figure 1.3 The compilation process

Figure 1.3 Full Alternative Text

The lexical analyzer gathers the characters of the source program into lexical units. The lexical units of a program are identifiers, special words, operators, and punctuation symbols. The lexical analyzer ignores comments in the source program because the compiler has no use for them.

The syntax analyzer takes the lexical units from the lexical analyzer and uses them to construct hierarchical structures called **parse trees**. These parse trees represent the syntactic structure of the program. In many cases, no actual parse tree structure is constructed; rather, the information that would be required to build a tree is generated and used directly. Both lexical units and parse trees are discussed further in Chapter 3. Lexical analysis and syntax analysis, or parsing, are discussed in Chapter 4.

The intermediate code generator produces a program in a different language, at an intermediate level between the source program and the final output of the compiler: the machine language program.4 Intermediate languages sometimes look very much like assembly languages, and in fact, sometimes are actual assembly languages. In other cases, the intermediate code is at a level somewhat higher than an assembly language. The semantic analyzer is an integral part of the intermediate code generator. The semantic analyzer checks for errors, such as type errors, that are difficult, if not impossible, to detect during syntax analysis.

4. Note that the words *program* and *code* are often used interchangeably.

Optimization, which improves programs (usually in their intermediate code version) by making them smaller or faster or both. Because many kinds of optimization are difficult to do on machine language, most optimization is done on the intermediate code.

The code generator translates the optimized intermediate code version of the

program into an equivalent machine language program.

The symbol table serves as a database for the compilation process. The primary contents of the symbol table are the type and attribute information of each user-defined name in the program. This information is placed in the symbol table by the lexical and syntax analyzers and is used by the semantic analyzer and the code generator.

As stated previously, although the machine language generated by a compiler can be executed directly on the hardware, it must nearly always be run along with some other code. Most user programs also require programs from the operating system. Among the most common of these are programs for input and output. The compiler builds calls to required system programs when they are needed by the user program. Before the machine language programs produced by a compiler can be executed, the required programs from the operating system must be found and linked to the user program. The linking operation connects the user program to the system programs by placing the addresses of the entry points of the system programs in the calls to them in the user program. The user and system code together are sometimes called a **load module**, or **executable image**. The process of collecting system programs and linking them to user programs is called **linking and loading**, or sometimes just **linking**. It is accomplished by a systems program called a **linker**.

In addition to systems programs, user programs must often be linked to previously compiled programs that reside in libraries. So the linker not only links a given program to system programs, but also it may link it to other user or system-supplied programs.

The speed of the connection between a computer’s memory and its processor often determines the speed of the computer, because instructions often can be executed faster than they can be moved to the processor for execution. This connection is called the **von Neumann bottleneck**; it is the primary limiting factor in the speed of von Neumann architecture computers. The von Neumann bottleneck has been one of the primary motivations for the research and development of parallel computers.

1.7.2 Pure Interpretation

Pure interpretation lies at the opposite end (from compilation) among implementation methods. With this approach, programs are interpreted by another program called an interpreter, with no translation whatever. The interpreter program acts as a software simulation of a machine whose fetch-execute cycle deals with high-level language program statements rather than machine instructions. This software simulation obviously provides a virtual machine for the language.

Pure interpretation has the advantage of allowing easy implementation of many source-level debugging operations, because all run-time error messages can refer to source-level units. For example, if an array index is found to be out of range, the error message can easily indicate the source line of the error and the name of the array. On the other hand, this method has the serious disadvantage that execution is 10 to 100 times slower than in compiled systems. The primary source of this slowness is the decoding of the high-level language statements, which are far more complex than machine language instructions (although there may be fewer statements than instructions in equivalent machine code). Furthermore, regardless of how many times a statement is executed, it must be decoded every time. Therefore, statement decoding, rather than the connection between the processor and memory, is the bottleneck of a pure interpreter.

Another disadvantage of pure interpretation is that it often requires more space. In addition to the source program, the symbol table must be present during interpretation. Furthermore, the source program may be stored in a form designed for easy access and modification rather than one that provides for minimal size.

Although some simple early languages of the 1960s (APL, SNOBOL (Griswold et al., 1971), and Lisp) were purely interpreted, by the 1980s, the approach was rarely used on high-level languages. However, in recent years, pure interpretation has made a significant comeback with some Web scripting languages, such as JavaScript and PHP, which are now widely used. The process of pure interpretation is shown in Figure 1.4.

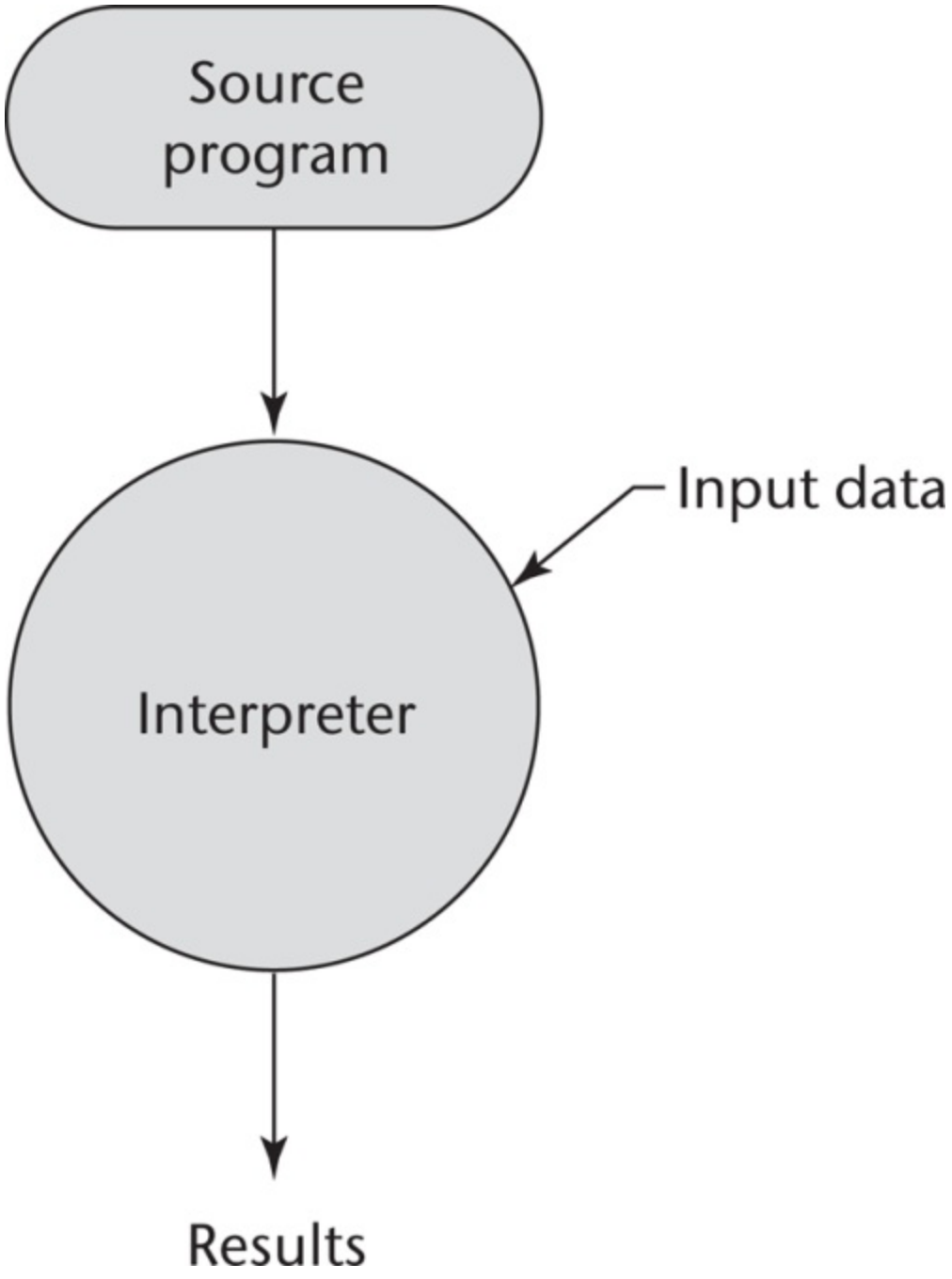


Figure 1.4 Pure interpretation

1.7.3 Hybrid Implementation

Systems

Some language implementation systems are a compromise between compilers and pure interpreters; they translate high-level language programs to an intermediate language designed to allow easy interpretation. This method is faster than pure interpretation because the source language statements are decoded only once. Such implementations are called **hybrid implementation systems**.

The process used in a hybrid implementation system is shown in Figure 1.5. Instead of translating intermediate language code to machine code, it simply interprets the intermediate code.

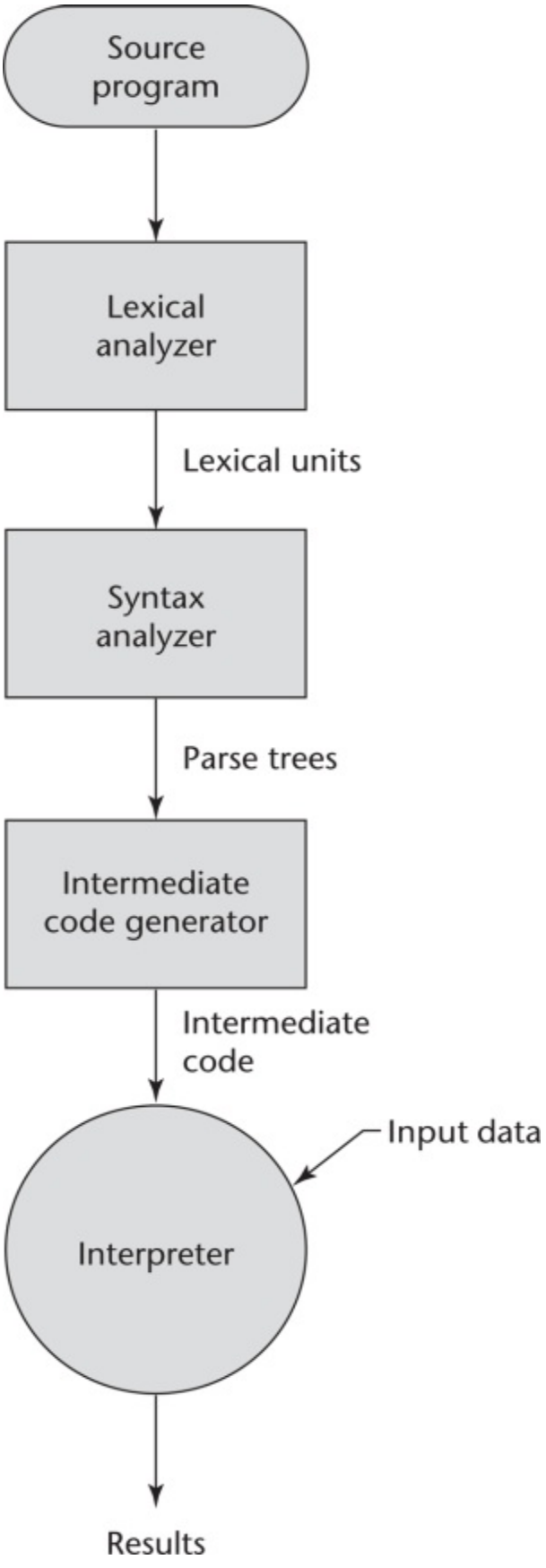

Source program
Lexical analyzer
Lexical units
Syntax analyzer
Parse trees
Intermediate code generator
Intermediate code
Input data
Interpreter
Results

Figure 1.5 Hybrid implementation system

Figure 1.5 Full Alternative Text

Perl is implemented with a hybrid system. Perl programs are partially - compiled to detect errors before interpretation and to simplify the interpreter.

Initial implementations of Java were all hybrid. Its intermediate form, called **byte code**, provides portability to any machine that has a byte code interpreter and an associated run-time system. Together, these are called the Java Virtual Machine. There are now systems that translate Java byte code into machine code for faster execution.

A Just-in-Time (JIT) implementation system initially translates programs to an intermediate language. Then, during execution, it compiles intermediate language methods into machine code when they are called. The machine code version is kept for subsequent calls. JIT systems now are widely used for Java programs. Also, the .NET languages are all implemented with a JIT system.

Sometimes an implementor may provide both compiled and interpreted implementations for a language. In these cases, the interpreter is used to develop and debug programs. Then, after a (relatively) bug-free state is reached, the programs are compiled to increase their execution speed.

1.7.4 Preprocessors

A **preprocessor** is a program that processes a program just before the program is compiled. Preprocessor instructions are embedded in programs. The preprocessor is essentially a macro expander. Preprocessor instructions are commonly used to specify that the code from another file is to be included. For example, the C preprocessor instruction

```
#include "myLib.h"
```

causes the preprocessor to copy the contents of `myLib.h` into the program at the position of the `#include`.

Other preprocessor instructions are used to define symbols to represent expressions. For example, one could use

```
#define max(A, B) ((A) > (B) ? (A) : (B))
```

to determine the largest of two given expressions. For example, the expression

```
x = max(2 * y, z / 1.73);
```

would be expanded by the preprocessor to

```
x = ((2 * y) > (z / 1.73) ? (2 * y) : (z / 1.73);
```

Notice that this is one of those cases where expression side effects can cause trouble. For example, if either of the expressions given to the `max` macro have side effects—such as `z++`—it could cause a problem. Because one of the two expression parameters is evaluated twice, this could result in `z` being -incremented twice by the code produced by the macro expansion.

1.8 Programming Environments

A programming environment is the collection of tools used in the development of software. This collection may consist of only a file system, a text editor, a linker, and a compiler. Or it may include a large collection of integrated tools, each accessed through a uniform user interface. In the latter case, the process of the development and maintenance of software is greatly enhanced. Therefore, the characteristics of a programming language are not the only measure of the software development capability of a system. We now briefly describe several programming environments.

UNIX is an older programming environment, first distributed in the middle 1970s, built around a portable multiprogramming operating system. It provides a wide array of powerful support tools for software production and maintenance in a variety of languages. In the past, the most important feature absent from UNIX was a uniform interface among its tools. This made it more difficult to learn and to use. However, UNIX is now often used through a GUI that runs on top of UNIX. Examples of UNIX GUIs are the Solaris Common Desktop Environment (CDE), GNOME, and KDE. These GUIs make the interface to UNIX appear similar to that of Windows and Macintosh systems.

Borland JBuilder is a programming environment that provides an integrated compiler, editor, debugger, and file system for Java development, where all four are accessed through a graphical interface. JBuilder is a complex and powerful system for creating Java software.

Microsoft Visual Studio .NET is a relatively recent step in the evolution of software development environments. It is a large and elaborate collection of software development tools, all used through a windowed interface. This system can be used to develop software in any one of the five .NET languages: C#, Visual Basic.NET, JScript (Microsoft’s version of JavaScript), F# (a functional language), and C++/CLI.

NetBeans is a development environment that is primarily used for Java

application development but also supports JavaScript, Ruby, and PHP. Both Visual Studio and NetBeans are more than development environments—they are also frameworks, which means they actually provide common parts of the code of the application.

SUMMARY

The study of programming languages is valuable for some important reasons: It increases our capacity to use different constructs in writing programs, enables us to choose languages for projects more intelligently, and makes learning new languages easier.

Computers are used in a wide variety of problem-solving domains. The design and evaluation of a particular programming language is highly - dependent on the domain in which it is to be used.

Among the most important criteria for evaluating languages are readability, writability, reliability, and overall cost. These will be the basis on which we examine and judge the various language features discussed in the remainder of the book.

The major influences on language design have been machine architecture and software design methodologies.

Designing a programming language is primarily an engineering feat, in which a long list of trade-offs must be made among features, constructs, and capabilities.

The major methods of implementing programming languages are compilation, pure interpretation, and hybrid implementation.

Programming environments have become important parts of software development systems, in which the language is just one of the components.

REVIEW QUESTIONS

1. Why is it useful for a programmer to have some background in language design, even though he or she may never actually design a programming language?

2. How can knowledge of programming language characteristics benefit the whole computing community?

3. What programming language has dominated scientific computing over the past 60 years?

4. What programming language has dominated business applications over the past 60 years?

5. What programming language has dominated artificial intelligence over the past 60 years?

6. In what language is most of UNIX written?

7. What is the disadvantage of having too many features in a language?

8. How can user-defined operator overloading harm the readability of a program?

9. What is one example of a lack of orthogonality in the design of C?

10. What language used orthogonality as a primary design criterion?

11. What primitive control statement is used to build more complicated control statements in languages that lack them?

12. What does it mean for a program to be reliable?

13. Why is type checking the parameters of a subprogram important?

14. What is aliasing?

15. What is exception handling?

16. Why is readability important to writability?

17. How is the cost of compilers for a given language related to the design of that language?

18. What have been the strongest influences on programming language design over the past 60 years?

19. What is the name of the category of programming languages whose structure is dictated by the von Neumann computer architecture?

20. What two programming language deficiencies were discovered as a result of the research in software development in the 1970s?

21. What are the three fundamental features of an object-oriented programming language?

22. What language was the first to support the three fundamental features of object-oriented programming?

23. What is an example of two language design criteria that are in direct conflict with each other?

24. What are the three general methods of implementing a programming language?

25. Which produces faster program execution, a compiler or a pure interpreter?

26. What role does the symbol table play in a compiler?

27. What does a linker do?

28. Why is the von Neumann bottleneck important?

29. What are the advantages in implementing a language with a pure interpreter?

PROBLEM SET

1. Do you believe our capacity for abstract thought is influenced by our language skills? Support your opinion.

2. What are some features of specific programming languages you know whose rationales are a mystery to you?

3. What arguments can you make for the idea of a single language for all programming domains?

4. What arguments can you make against the idea of a single language for all programming domains?

5. Name and explain another criterion by which languages can be judged (in addition to those discussed in this chapter).

6. What common programming language statement, in your opinion, is most detrimental to readability?

7. Java uses a right brace to mark the end of all compound statements. What are the arguments for and against this design?

8. Many languages distinguish between uppercase and lowercase letters in user-defined names. What are the pros and cons of this design decision?

9. Explain the different aspects of the cost of a programming language.

10. What are the arguments for writing efficient programs even though hardware is relatively inexpensive?

11. Describe some design trade-offs between efficiency and safety in some language you know.

12. In your opinion, what major features would a perfect programming language include?

13. Was the first high-level programming language you learned implemented with a pure interpreter, a hybrid implementation system, or a compiler? (You may have to research this.)

14. Describe the advantages and disadvantages of some programming environment you have used.

15. How do type declaration statements for simple variables affect the readability of a language, considering that some languages do not require them?

16. Write an evaluation of some programming language you know, using the criteria described in this chapter.

17. Some programming languages—for example, Pascal—have used the semicolon to separate statements, while Java uses it to terminate statements. Which of these, in your opinion, is most natural and least likely to result in syntax errors? Support your answer.

18. Many contemporary languages allow two kinds of comments: one in which delimiters are used on both ends (multiple-line comments), and one in which a delimiter marks only the beginning of the comment (one-line comments). Discuss the advantages and disadvantages of each of these with respect to our criteria.

2 Evolution of the Major Programming Languages

This chapter describes the development of a collection of programming - languages. It explores the environment in which each was designed and focuses on the contributions of the language and the motivation for its development. Overall language descriptions are not included; rather, we discuss only some of the new features introduced by each language. Of particular interest are the features that most influenced subsequent languages or the field of computer science.

This chapter does not include an in-depth discussion of any language feature or concept; that is left for later chapters. Brief, informal explanations of features will suffice for our trek through the development of these languages.

We discuss a wide variety of languages and language concepts that will not be familiar to many readers. These topics are described in detail only in later chapters. Those who find this unsettling may prefer to delay reading this chapter until the rest of the book has been studied.

The choice as to which languages to discuss here was subjective, and some - readers will unhappily note the absence of one or more of their favorites. However, to keep this historical coverage to a reasonable size, it was necessary to leave out some languages that some regard highly. The choices were based on our estimate of each language’s importance to language development and the computing world as a whole. We also include brief discussions of some other languages that are referenced later in the book.

The organization of this chapter is as follows: The initial versions of languages generally are discussed in chronological order. However, subsequent versions of languages appear with their initial version, rather than in later sections. For example, Fortran 2003 is discussed in the section with Fortran I (1956). Also, in some cases, languages of secondary importance that are related to a language that has its own section appear in that section.

This chapter includes listings of 14 complete example programs, each in a different language. These programs are not described in this chapter; they are meant to illustrate the appearance of programs in these languages. Readers familiar with any of the common imperative languages should be able to read and understand most of the code in these programs, except those in Lisp, COBOL, and Smalltalk. (A Scheme function similar to the Lisp example is discussed in Chapter 15.) The same problem is solved by the Fortran, ALGOL 60, PL/I, Basic, Pascal, C, Perl, Ada, Java, JavaScript, and C# programs. Note that most of the contemporary languages in this list support dynamic arrays, but because of the simplicity of the example problem, we did not use them in the example programs. Also, in the Fortran 95 program, we avoided using the features that could have avoided the use of loops altogether, in part to keep the program simple and readable and in part just to illustrate the basic loop structure of the language.

Figure 2.1 is a chart of the genealogy of the high-level languages discussed in this chapter.

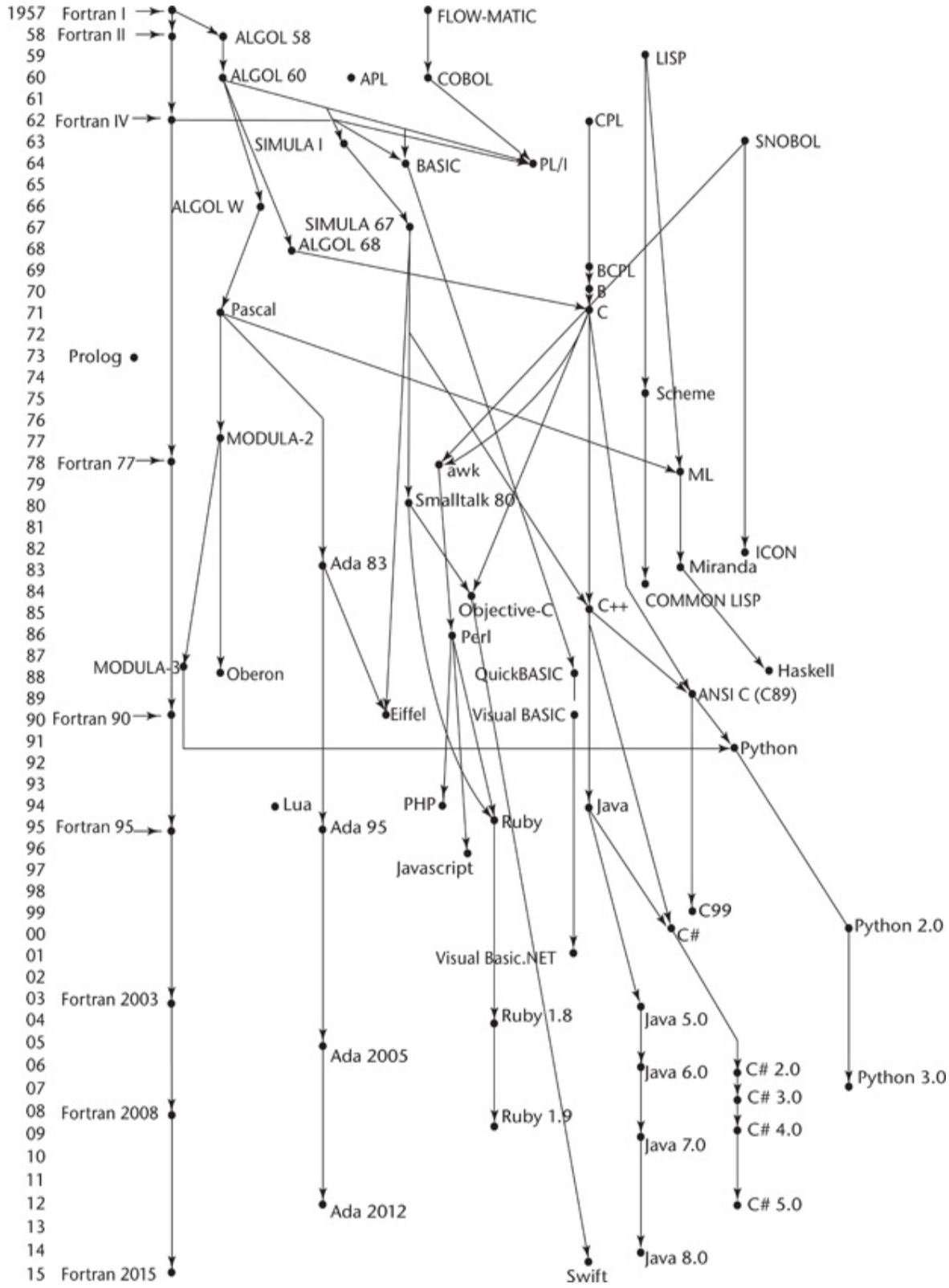

1957 Fortran I
58 Fortran II
59
60
61
62 Fortran IV
63
64
65
66
67
68
69
70
71
72
73 Prolog
74
75
76
77
78 Fortran 77
79
80
81
82
83
84
85
86
87
88
89
90 Fortran 90
91
92
93
94
95 Fortran 95
96
97
98
99
00
01
02
03 Fortran 2003
04
05
06
07
08 Fortran 2008
09
10
11
12
13
14
15 Fortran 2015
ALGOL 58
ALGOL 60
FLOW-MATIC
APL
COBOL
LISP
CPL
SNOBOL
SIMULA I
BASIC
PL/I
ALGOL W
SIMULA 67
ALGOL 68
BCPL
B
C
Pascal
Scheme
MODULA-2
awk
ML
Fortran 77
Smalltalk 80
ICON
Ada 83
Miranda
COMMON LISP
Objective-C
C++
Perl
Haskell
MODULA-3
Oberon
QuickBASIC
ANSI C (C89)
Eiffel
Visual BASIC
Python
Lua
PHP
Java
Ada 95
Ruby
Javascript
C99
Python 2.0
C#
Visual Basic.NET
Ruby 1.8
Java 5.0
Ada 2005
Java 6.0
C# 2.0
Python 3.0
C# 3.0
Ruby 1.9
C# 4.0
Java 7.0
Ada 2012
C# 5.0
Java 8.0
Swift

Figure 2.1 Genealogy of common high-level programming languages

Figure 2.1 Full Alternative Text

2.1 Zuse’s Plankalkül

The first programming language discussed in this chapter is highly unusual in several respects. For one thing, it was never implemented. Furthermore, although developed in 1945, its description was not published until 1972. Because so few people were familiar with the language, some of its capabilities did not appear in other languages until 15 years after its development.

2.1.1 Historical Background

Between 1936 and 1945, German scientist Konrad Zuse (pronounced “Tsoo-zuh”) built a series of complex and sophisticated computers from electromechanical relays. By early 1945, Allied bombing had destroyed all but one of his latest models, the Z4, so he moved to a remote Bavarian village, Hinterstein, and his research group members went their separate ways.

Working alone, Zuse embarked on an effort to develop a language for expressing computations for the Z4, a project he had begun in 1943 as a proposal for his Ph.D. dissertation. He named this language Plankalkül, which means *program calculus*. In a lengthy manuscript dated 1945 but not published until 1972 (Zuse, 1972), Zuse defined Plankalkül and wrote algorithms in the language to solve a wide variety of problems.

2.1.2 Language Overview

Plankalkül was remarkably complete, with some of its most advanced features in the area of data structures. The simplest data type in Plankalkül was the single bit. Integer and floating-point numeric types were built from the bit type. The floating-point type used twos-complement notation and the “hidden bit” scheme currently used to avoid storing the most significant bit of

the normalized fraction part of a floating-point value.

In addition to the usual scalar types, Plankalkül included arrays and records (called *structs* in the C-based languages). The records could include nested records.

Although the language had no explicit goto, it did include an iterative statement similar to the Ada `for`. It also had the command `Fin` with a superscript that specified an exit out of a given number of iteration loop nestings or to the beginning of a new iteration cycle. Plankalkül included a selection statement, but it did not allow an `else` clause.

One of the most interesting features of Zuse's programs was the inclusion of mathematical expressions showing the current relationships between program variables. These expressions stated what would be true during execution at the points in the code where they appeared. These are very similar to the assertions of Java and in those in axiomatic semantics, which is discussed in Chapter 3.

Zuse's manuscript contained programs of far greater complexity than any written prior to 1945. Included were programs to sort arrays of numbers; test the connectivity of a given graph; carry out integer and floating-point operations, including square root; and perform syntax analysis on logic formulas that had parentheses and operators in six different levels of precedence. Perhaps most remarkable were his 49 pages of algorithms for playing chess, a game in which he was not an expert.

If a computer scientist had found Zuse's description of Plankalkül in the early 1950s, the single aspect of the language that would have hindered its implementation as defined would have been the notation. Each statement consisted of either two or three lines of code. The first line was most like the statements of current languages. The second line, which was optional, contained the subscripts of the array references in the first line. The same method of indicating subscripts was used by Charles Babbage in programs for his Analytical Engine in the middle of the nineteenth century. The last line of each Plankalkül statement contained the type names for the variables mentioned in the first line. This notation is quite intimidating when first seen.

The following example assignment statement, which assigns the value of the expression `A[4} + 1` to `A[5]`, illustrates this notation. The row labeled `V` is for subscripts, and the row labeled `S` is for the data types. In this example, `1.n` means an integer of *n* bits:

```
  | A + 1 =>  A
V | 4         5
S | 1.n       1.n
```

We can only speculate on the direction that programming language design might have taken if Zuse's work had been widely known in 1945 or even 1950. It is also interesting to consider how his work might have been different had he done it in a peaceful environment surrounded by other scientists, rather than in Germany in 1945 in virtual isolation.

2.2 Pseudocodes

First, note that the word *pseudocode* is used here in a different sense than its contemporary meaning. We call the languages discussed in this section pseudocodes because that's what they were named at the time they were developed and used (the late 1940s and early 1950s). However, they are clearly not pseudocodes in the contemporary sense.

The computers that became available in the late 1940s and early 1950s were far less usable than those of today. In addition to being slow, unreliable, expensive, and having extremely small memories, the machines of that time were difficult to program because of the lack of supporting software.

There were no high-level programming languages or even assembly languages, so programming was done in machine code, which is both tedious and error prone. Among its problems is the use of numeric codes for specifying instructions. For example, an ADD instruction might be specified by the code 14 rather than a connotative textual name, even if only a single letter. This makes programs very difficult to read. A more serious problem is absolute addressing, which makes program modification tedious and error prone. For example, suppose we have a machine language program stored in memory. Many of the instructions in such a program refer to other locations within the program, usually to reference data or to indicate the targets of branch instructions. Inserting an instruction at any position in the program other than at the end invalidates the correctness of all instructions that refer to addresses beyond the insertion point, because those addresses must be increased to make room for the new instruction. To make the addition correctly, all instructions that refer to addresses that follow the addition must be found and modified. A similar problem occurs with deletion of an instruction. In this case, however, machine languages often include a "no operation" instruction that can replace deleted instructions, thereby avoiding the problem.

These are standard problems with all machine languages and were the primary motivations for inventing assemblers and assembly languages. In

addition, most programming problems of that time were numerical and required floating-point arithmetic operations and indexing of some sort to allow the convenient use of arrays. Neither of these capabilities, however, was included in the architecture of the computers of the late 1940s and early 1950s. These deficiencies naturally led to the development of somewhat higher-level languages.

2.2.1 Short Code

The first of these new languages, named Short Code, was developed by John Mauchly in 1949 for the BINAC computer, which was one of the first successful stored-program electronic computers. Short Code was later transferred to a UNIVAC I computer (the first commercial electronic computer sold in the United States) and, for several years, was one of the primary means of programming those machines. Although little is known of the original Short Code because its complete description was never published, a programming manual for the UNIVAC I version did survive (Remington-Rand, 1952). It is safe to assume that the two versions were very similar.

The words of the UNIVAC I's memory had 72 bits, grouped as 12 six-bit bytes. Short Code consisted of coded versions of mathematical expressions that were to be evaluated. The codes were byte-pair values, and many equations could be coded in a word. The following operation codes were included:

```
01 -    06 abs value      1n (n+2)nd power
02 )    07 +              2n (n+2)nd root
03 =    08 pause          4n if <= n
04 /    09 (              58 print and tab
```

Variables were named with byte-pair codes, as were locations to be used as constants. For example, `X0` and `Y0` could be variables. The statement

```
X0 = SQRT(ABS(Y0))
```

would be coded in a word as `00 X0 03 20 06 Y0`. The initial `00` was used as

padding to fill the word. Interestingly, there was no multiplication code; multiplication was indicated by simply placing the two operands next to each other, as in algebra.

Short Code was not translated to machine code; rather, it was implemented with a pure interpreter. At the time, this process was called *automatic programming*. It clearly simplified the programming process, but at the expense of execution time. Short Code interpretation was approximately 50 times slower than machine code.

2.2.2 Speedcoding

In other places, interpretive systems were being developed that extended machine languages to include floating-point operations. The Speedcoding system developed by John Backus for the IBM 701 is an example of such a system (Backus, 1954). The Speedcoding interpreter effectively converted the 701 to a virtual three-address floating-point calculator. The system included pseudoinstructions for the four arithmetic operations on floating-point data, as well as operations such as square root, sine, arc tangent, exponent, and logarithm. Conditional and unconditional branches and input/output conversions were also part of the virtual architecture. To get an idea of the limitations of such systems, consider that the remaining usable memory after loading the interpreter was only 700 words and that the add instruction took 4.2 milliseconds to execute. On the other hand, Speedcoding included the novel facility of automatically incrementing address registers. This facility did not appear in hardware until the UNIVAC 1107 computers of 1962. Because of such features, matrix multiplication could be done in 12 Speedcoding instructions. Backus claimed that problems that could take two weeks to program in machine code could be programmed in a few hours using Speedcoding.

2.2.3 The UNIVAC “Compiling” System

Between 1951 and 1953, a team led by Grace Hopper at UNIVAC developed a series of “compiling” systems named A-0, A-1, and A-2 that expanded a pseudocode into machine code subprograms in the same way as macros are expanded into assembly language. The pseudocode source for these “compilers” was still quite primitive, although even this was a great improvement over machine code because it made source programs much shorter. Wilkes (1952) independently suggested a similar process.

2.2.4 Related Work

Other means of easing the task of programming were being developed at about the same time. At Cambridge University, David J. Wheeler (1950) developed a method of using blocks of relocatable addresses to solve, at least partially, the problem of absolute addressing, and later, Maurice V. Wilkes (also at Cambridge) extended the idea to design an assembly program that could combine chosen subroutines and allocate storage (Wilkes et al., 1951, 1957). This was indeed an important and fundamental advance.

We should also mention that assembly languages, which are quite different from the pseudocodes discussed, evolved during the early 1950s. However, they had little impact on the design of high-level languages.

2.3 The IBM 704 and Fortran

Certainly one of the greatest single advances in computing came with the introduction of the IBM 704 in 1954, in large measure because its capabilities prompted the development of Fortran. One could argue that if it had not been IBM with the 704 and Fortran, it would soon thereafter have been some other organization with a similar computer and related high-level language. However, IBM was the first with both the foresight and the resources to undertake these developments.

2.3.1 Historical Background

One of the primary reasons why the slowness of interpretive systems was tolerated from the late 1940s to the mid-1950s was the lack of floating-point hardware in the available computers. All floating-point operations had to be simulated in software, a very time-consuming process. Because so much processor time was spent in software floating-point processing, the overhead of interpretation and the simulation of indexing were relatively insignificant. As long as floating-point had to be done by software, interpretation was an acceptable expense. However, many programmers of that time never used interpretive systems, preferring the efficiency of hand-coded machine (or assembly) language. The announcement of the IBM 704 system, with both indexing and floating-point instructions in hardware, heralded the end of the interpretive era, at least for scientific computation. The inclusion of floating-point hardware removed the hiding place for the cost of interpretation.

Although Fortran is often credited with being the first compiled high-level language, the question of who deserves credit for implementing the first such language is somewhat open. Knuth and Pardo (1977) give the credit to Alick E. Glennie for his Autocode compiler for the Manchester Mark I computer. Glennie developed the compiler at Fort Halstead, Royal Armaments Research Establishment, in England. The compiler was operational by September 1952. However, according to John Backus (Wexelblat, 1981, p. 26),

Glennie's Autocode was so low level and machine oriented that it should not be considered a compiled system. Backus gives the credit to Laning and Zierler at the Massachusetts Institute of Technology.

The Laning and Zierler system (Laning and Zierler, 1954) was the first algebraic translation system to be implemented. By algebraic, we mean that it translated arithmetic expressions, used separately coded subprograms to compute transcendental functions (e.g., sine and logarithm), and included arrays. The system was implemented on the MIT Whirlwind computer, in experimental prototype form, in the summer of 1952 and in a more usable form by May 1953. The translator generated a subroutine call to code each formula, or expression, in the program. The source language was easy to read, and the only actual machine instructions included were for branching. Although this work preceded the work on Fortran, it never escaped MIT.

In spite of these earlier works, the first widely accepted compiled high-level language was Fortran. The following subsections chronicle this important development.

2.3.2 Design Process

Even before the 704 system was announced in May 1954, plans were begun for Fortran. By November 1954, John Backus and his group at IBM had produced the report titled "The IBM Mathematical FORmula TRANslating System: FORTRAN" (IBM, 1954). This document described the first version of Fortran, which we refer to as Fortran 0, prior to its implementation. It also boldly stated that Fortran would provide the efficiency of hand-coded programs and the ease of programming of the interpretive pseudocode systems. In another burst of optimism, the document stated that Fortran would eliminate coding errors and the debugging process. Based on this premise, the first Fortran compiler included little syntax error checking.

The environment in which Fortran was developed was as follows: (1) Computers had small memories and were slow and relatively unreliable; (2) the primary use of computers was for scientific computations; (3) there were no existing efficient and effective ways to program computers; and (4)

because of the high cost of computers compared to the cost of programmers, speed of the generated object code was the primary goal of the first Fortran compilers. The characteristics of the early versions of Fortran follow directly from this environment.

2.3.3 Fortran I Overview

Fortran 0 was modified during the implementation period, which began in January 1955 and continued until the release of the compiler in April 1957. The implemented language, which we call Fortran I, is described in the first Fortran *Programmer's Reference Manual*, published in October 1956 (IBM, 1956). Fortran I included input/output formatting, variable names of up to six characters (it had been just two in Fortran 0), user-defined subroutines, although they could not be separately compiled, the `If` selection statement, and the `Do` loop statement.

All of Fortran I's control statements were based on 704 instructions. It is not clear whether the 704 designers dictated the control statement design of Fortran I or whether the designers of Fortran I suggested these instructions to the 704 designers.

There were no data-typing statements in the Fortran I language. Variables whose names began with `I`, `J`, `K`, `L`, `M`, and `N` were implicitly integer type, and all others were implicitly floating-point. The choice of the letters for this convention was based on the fact that at that time scientists and engineers used letters as variable subscripts, usually *i*, *j*, and *k*. In a gesture of generosity, Fortran's designers threw in the three additional letters.

The most audacious claim made by the Fortran development group during the design of the language was that the machine code produced by the compiler would be about half as efficient as what could be produced by hand.1 This, more than anything else, made skeptics of potential users and prevented a great deal of interest in Fortran before its actual release. To almost everyone's surprise, however, the Fortran development group nearly achieved its goal in efficiency. The largest part of the 18 worker-years of effort used to construct the first compiler had been spent on optimization, and

the results were remarkably effective.

1. In fact, the Fortran team believed that the code generated by their compiler could be no less than half as fast as handwritten machine code, or the language would not be adopted by users.

The early success of Fortran is shown by the results of a survey made in April 1958. At that time, roughly half of the code being written for 704s was being written in Fortran, in spite of the skepticism of most of the programming world only a year earlier.

2.3.4 Fortran II

The Fortran II compiler was distributed in the spring of 1958. It fixed many of the bugs in the Fortran I compilation system and added some significant features to the language, the most important being the independent compilation of subroutines. Without independent compilation, any change in a program required that the entire program be recompiled. Fortran I's lack of independent-compilation capability, coupled with the poor reliability of the 704, placed a practical restriction on the length of programs to about 300 to 400 lines (Wexelblat, 1981, p. 68). Longer programs had a poor chance of being compiled completely before a machine failure occurred. The capability of including precompiled machine language versions of subprograms shortened the compilation process considerably and made it practical to develop much larger programs.

2.3.5 Fortrans IV, 77, 90, 95, 2003, and 2008

A Fortran III was developed, but it was never widely distributed. Fortran IV, however, became one of the most widely used programming languages of its time. It evolved over the period 1960 to 1962 and was standardized as Fortran 66 (ANSI, 1966), although that name was rarely used. Fortran IV was

an improvement over Fortran II in many ways. Among its most important additions were explicit type declarations for variables, a logical `If` construct, and the capability of passing subprograms as parameters to other subprograms.

Fortran IV was replaced by Fortran 77, which became the new standard in 1978 (ANSI, 1978a). Fortran 77 retained most of the features of Fortran IV and added character string handling, logical loop control statements, and an `If` with an optional `Else` clause.

Fortran 90 (ANSI, 1992) was dramatically different from Fortran 77. The most significant additions were dynamic arrays, records, pointers, a multiple selection statement, and modules. In addition, Fortran 90 subprograms could be recursively called.

A new concept that was included in the Fortran 90 definition was that of removing some language features from earlier versions. While Fortran 90 included all of the features of Fortran 77, the language definition included a list of constructs that were recommended for removal in the next version of the language.

Fortran 90 included two simple syntactic changes that altered the appearance of both programs and the literature describing the language. First, the required fixed format of code, which required the use of specific character positions for specific parts of statements, was dropped. For example, statement labels could appear only in the first five positions and statements could not begin before the seventh position. This rigid formatting of code was designed around the use of punch cards. The second change was that the official spelling of FORTRAN became Fortran. This change was accompanied by the change in convention of using all uppercase letters for keywords and identifiers in Fortran programs. The new convention was that only the first letter of keywords and identifiers would be uppercase.

Fortran 95 (INCITS/ISO/IEC, 1997) continued the evolution of the language, but only a few changes were made. Among other things, a new iteration construct, `Forall`, was added to ease the task of parallelizing Fortran programs.

Fortran 2003 (Metcalf et al., 2004) added support for object-oriented programming, parameterized derived types, procedure pointers, and interoperability with the C programming language.

The latest version of Fortran, Fortran 2008 (ISO/IEC 1539-1, 2010), added support for blocks to define local scopes, co-arrays, which provide a parallel execution model, and the `DO CONCURRENT` construct, to specify loops without interdependencies. A new version, Fortran 2015, is under development and is scheduled for release in 2018.

2.3.6 Evaluation

The original Fortran design team thought of language design only as a necessary prelude to the critical task of designing the translator. Furthermore, it never occurred to them that Fortran would be used on computers not manufactured by IBM. However, they were forced to consider building Fortran compilers for other IBM machines only because the successor to the 704, the 709, was announced before the 704 Fortran compiler was released. The effect that Fortran has had on the use of computers, along with the fact that all subsequent programming languages owe a debt to Fortran, is indeed impressive in light of the modest goals of its designers.

One of the features of Fortran I, and all of its successors before 90, that allows highly optimizing compilers was that the types and storage for all - variables are fixed before run time. No new variables or space could be allocated during execution. This was a sacrifice of flexibility to simplicity and efficiency. It eliminated the possibility of recursive subprograms and made it difficult to implement data structures that grow or change shape dynamically. Of course, the kinds of programs that were being built at the time of the development of the early versions of Fortran were primarily numerical in nature and were simple in comparison with more recent software projects. Therefore, the sacrifice was not a great one.

The overall success of Fortran is difficult to overstate: It dramatically changed the way computers are used. This is, of course, in large part due to its being the first widely used high-level language. In comparison with

concepts and languages developed later, early versions of Fortran suffer in a variety of ways, as should be expected. After all, it would not be fair to compare the performance and comfort of a 1910 Model T Ford with the performance and comfort of a 2017 Ford Mustang. Nevertheless, in spite of the inadequacies of Fortran, the momentum of the huge investment in Fortran software, among other factors, has kept it in use for 60 years.

Alan Perlis, one of the designers of ALGOL 60, said of Fortran in 1978, “Fortran is the *lingua franca* of the computing world. It is the language of the streets in the best sense of the word, not in the prostitutional sense of the word. And it has survived and will survive because it has turned out to be a remarkably useful part of a very vital commerce” (Wexelblat, 1981, p. 161).

The following is an example of a Fortran 95 program:

```
! Fortran 95 Example program
!  Input:   An integer, List_Len, where List_Len is less
!           than 100, followed by List_Len-Integer values
!  Output:  The number of input values that are greater
!           than the average of all input values
Implicit none
Integer Dimension(99) :: Int_List
Integer :: List_Len, Counter, Sum, Average, Result
Result= 0
Sum = 0
Read *, List_Len
If ((List_Len > 0) .AND. (List_Len < 100)) Then
! Read input data into an array and compute its sum
   Do Counter = 1, List_Len
      Read *, Int_List(Counter)
      Sum = Sum + Int_List(Counter)
   End Do
! Compute the average
   Average = Sum / List_Len
! Count the values that are greater than the average
   Do Counter = 1, List_Len
      If (Int_List(Counter) > Average) Then
          Result = Result + 1
      End If
   End Do
! Print the result
   Print *, 'Number of values > Average is:', Result
Else
   Print *, 'Error - list length value is not legal'
```

```
End If
End Program Example
```

2.4 Functional Programming: Lisp

The first functional programming language was invented to provide language features for list processing, the need for which grew out of the first applications in the area of artificial intelligence (AI).

2.4.1 The Beginnings of Artificial Intelligence (AI) and List Processing

Interest in AI appeared in the mid-1950s in a number of places. Some of this interest grew out of linguistics, some from psychology, and some from mathematics. Linguists were concerned with natural language processing. - Psychologists were interested in modeling human information storage and retrieval, as well as other fundamental processes of the brain. Mathematicians were interested in mechanizing certain intelligent processes, such as theorem proving. All of these investigations arrived at the same conclusion: Some method must be developed to allow computers to process symbolic data in linked lists. At the time, nearly all computation was on numeric data in arrays.

The concept of list processing was developed by Allen Newell, J. C. Shaw, and Herbert Simon at the RAND Corporation. It was first published in a classic paper that describes one of the first AI programs, the Logic Theorist,[2] and a language in which it could be implemented (Newell and Simon, 1956). The language, named IPL-I (Information Processing Language I), was never implemented. The next version, IPL-II, was implemented on a RAND Johnniac computer. Development of IPL continued until 1960, when the description of IPL-V was published (Newell and Tonge, 1960). The low level of the IPL languages prevented their widespread use. They were actually assembly languages for a hypothetical computer, implemented with an interpreter, in which list-processing instructions were included. Another factor that kept the IPL languages from becoming popular was their

implementation on the obscure Johnniac machine.

2. Logic Theorist discovered proofs for theorems in propositional calculus.

The contributions of the IPL languages were in their list design and their demonstration that list processing was feasible and useful.

IBM became interested in AI in the mid-1950s and chose theorem proving as a demonstration area. At the time, the Fortran project was still underway. The high cost of the Fortran I compiler convinced IBM that their list processing should be attached to Fortran, rather than in the form of a new language. Thus, the Fortran list processing language (FLPL) was designed and implemented as an extension to Fortran. FLPL was used to construct a theorem prover for plane geometry, which was then considered the easiest area for mechanical theorem proving.

2.4.2 Lisp Design Process

John McCarthy of MIT took a summer position at the IBM Information Research Department in 1958. His goal for the summer was to investigate symbolic computations and to develop a set of requirements for doing such computations. As a pilot example problem area, he chose differentiation of algebraic expressions. From this study came a list of language requirements. Among them were the control flow methods of mathematical functions: recursion and conditional expressions. The only available high-level language of the time, Fortran I, had neither of these.

Another requirement that grew from the symbolic-differentiation investigation was the need for dynamically allocated linked lists and some kind of implicit deallocation of abandoned lists. McCarthy simply would not allow his elegant algorithm for differentiation to be cluttered with explicit deallocation statements.

Because FLPL did not support recursion, conditional expressions, dynamic storage allocation, or implicit deallocation, it was clear to McCarthy that a new language was needed.

When McCarthy returned to MIT in the fall of 1958, he and Marvin Minsky formed the MIT AI Project, with funding from the Research Laboratory for Electronics at MIT. The first important effort of the project was to produce a software system for list processing. It was to be used initially to implement a program proposed by McCarthy called the Advice Taker.3 This application became the impetus for the development of the list-processing language Lisp. The first version of Lisp is sometimes called “pure Lisp” because it is a purely functional language. In the following section, we describe the development of pure Lisp.

3. Advice Taker represented information with sentences written in a formal language and used a logical inferencing process to decide what to do.

2.4.3 Language Overview

2.4.3.1 Data Structures

Pure Lisp has only two kinds of data structures: atoms and lists. Atoms are either symbols, which have the form of identifiers, or numeric literals. The concept of storing symbolic information in linked lists is natural and was used in IPL-II. Such structures allow insertions and deletions at any point, operations that were then thought to be a necessary part of list processing. It was eventually determined, however, that Lisp programs rarely require these operations.

Lists are specified by delimiting their elements with parentheses. Simple lists, in which elements are restricted to atoms, have the form

```
(A B C D)
```

Nested list structures are also specified by parentheses. For example, the list

```
(A (B C) D (E (F G)))
```

is composed of four elements. The first is the atom `A`; the second is the sublist `(B C)`; the third is the atom `D`; the fourth is the sublist `(E (F G))`, which has

as its second element the sublist `(F G)`.

Internally, lists are stored as single-linked list structures, in which each node has two pointers and represents a list element. A node containing an atom has its first pointer pointing to some representation of the atom, such as its symbol or numeric value, or a pointer to a sublist. A node for a sublist element has its first pointer pointing to the first node of the sublist. In both cases, the second pointer of a node points to the next element of the list. A list is referenced by a pointer to its first element.

The internal representations of the two lists shown earlier are depicted in Figure 2.2. Note that the elements of a list are shown horizontally. The last element of a list has no successor, so its link is `NIL`, which is represented in Figure 2.2 as a diagonal line in the element. Sublists are shown with the same structure.

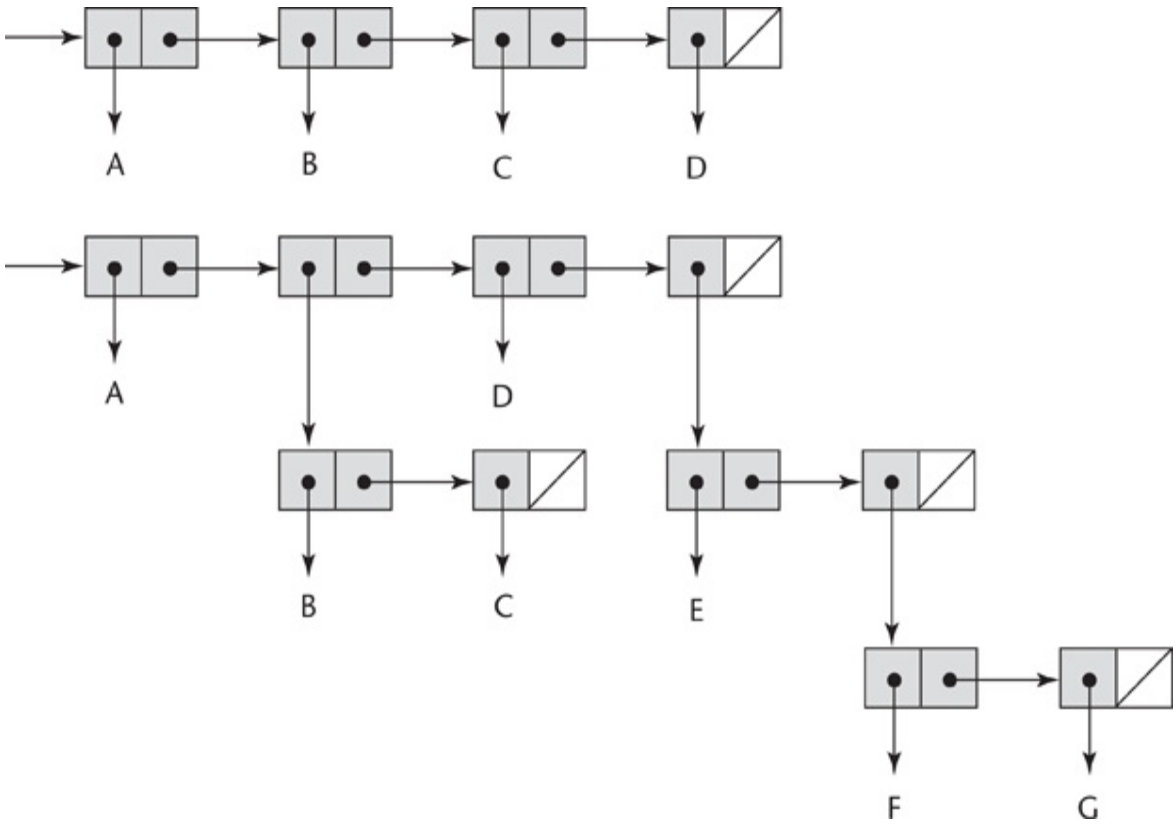


Figure 2.2 Internal

representation of two Lisp lists

Figure 2.2 Full Alternative Text

2.4.3.2 Processes in Functional Programming

Lisp was designed as a functional programming language. All computation in a purely functional program is accomplished by applying functions to arguments. Neither the assignment statements nor the variables that abound in imperative language programs are necessary in functional language programs. Furthermore, repetitive processes can be specified with recursive function calls, making iteration (loops) unnecessary. These basic concepts of functional programming make it significantly different from programming in an imperative language.

2.4.3.3 The Syntax of Lisp

Lisp is very different from the imperative languages, both because it is a functional programming language and because the appearance of Lisp programs is so different from those in languages like Java or C++. For example, the syntax of Java is a complicated mixture of English and algebra, while Lisp's syntax is a model of simplicity. Program code and data have exactly the same form: parenthesized lists. Consider again the list

```
(A B C D)
```

When interpreted as data, it is a list of four elements. When viewed as code, it is the application of the function named `A` to the three parameters `B`, `C`, and `D`.

2.4.4 Evaluation

Lisp completely dominated AI applications for a quarter century. Much of the cause of Lisp's reputation for being highly inefficient has been eliminated. Many contemporary implementations are compiled, and the resulting code is much faster than running the source code on an interpreter. In addition to its success in AI, Lisp pioneered functional programming, which has proven to be a lively area of research in programming languages. As stated in Chapter 1, many programming language researchers believe functional programming is a much better approach to software development than procedural programming using imperative languages.

The following is an example of a Lisp program:

```
; Lisp Example function
; The following code defines a Lisp predicate function
; that takes two lists as arguments and returns True
; if the two lists are equal, and NIL (false) otherwise
  (DEFUN equal_lists (lis1 lis2)
   (COND
     ((ATOM lis1) (EQ lis1 lis2))
     ((ATOM lis2) NIL)
     ((equal_lists (CAR lis1) (CAR lis2))
                 (equal_lists (CDR lis1) (CDR lis2)))
     (T NIL)
   )
)
```

2.4.5 Two Descendants of Lisp

Two dialects of Lisp are now widely used, Scheme and Common Lisp. These are briefly discussed in the following subsections.

2.4.5.1 Scheme

The Scheme language emerged from MIT in the mid-1970s. It is characterized by its small size, its exclusive use of static scoping (discussed in Chapter 5), and its treatment of functions as first-class entities. As first-class entities, Scheme functions can be assigned to variables, passed as

parameters, and returned as the values of function applications. They can also be the elements of lists. Early versions of Lisp did not provide all of these capabilities, nor did they use static scoping.

As a small language with simple syntax and semantics, Scheme is well suited to educational applications, such as courses in functional programming and general introductions to programming. Scheme is described in some detail in Chapter 15.

2.4.5.2 Common Lisp

During the 1970s and early 1980s, a large number of different dialects of Lisp were developed and used. This led to the familiar problem of lack of portability among programs written in the various dialects. Common Lisp - (Graham, 1996) was created in an effort to rectify this situation. Common Lisp was designed by combining the features of several dialects of Lisp developed in the early 1980s, including Scheme, into a single language. Being such an amalgam, Common Lisp is a relatively large and complex language. Its basis, however, is pure Lisp, so its syntax, primitive functions, and fundamental nature come from that language.

Recognizing the flexibility provided by dynamic scoping as well as the simplicity of static scoping, Common Lisp allows both. The default scoping for variables is static, but by declaring a variable to be **`special`**, that variable becomes dynamically scoped.

Common Lisp has a large number of data types and structures, including records, arrays, complex numbers, and character strings. It also has a form of packages for modularizing collections of functions and data providing access control.

Common Lisp is further described in Chapter 15.

2.4.6 Related Languages

ML (*Meta**L**anguage*; Ullman, 1998) was originally designed in the 1980s by Robin Milner at the University of Edinburgh as a metalanguage for a program verification system named Logic for Computable Functions (LCF; Milner et al., 1997). ML is primarily a functional language, but it also supports imperative programming. Unlike Lisp and Scheme, the type of every variable and expression in ML can be determined at compile time. Types are associated with objects rather than names. Types of names and expressions are inferred from their context.

Unlike Lisp and Scheme, ML does not use the parenthesized functional syntax that originated with lambda expressions. Rather, the syntax of ML resembles that of the imperative languages, such as Java and C++.

Miranda was developed by David Turner (1986) at the University of Kent in Canterbury, England, in the early 1980s. Miranda is based partly on the languages ML, SASL, and KRC. Haskell (Hudak and Fasel, 1992) is based in large part on Miranda. Like Miranda, it is a purely functional language, having no variables and no assignment statement. Another distinguishing characteristic of Haskell is its use of lazy evaluation. This means that no expression is evaluated until its value is required. This leads to some surprising capabilities in the language.

Caml (Cousineau et al., 1998) and its dialect that supports object-oriented programming, OCaml (Smith, 2006), descended from ML and Haskell. Finally, F# is a relatively new typed language based directly on OCaml. F# (Syme et al., 2010) is a .NET language with direct access to the whole .NET library. Being a .NET language also means it can smoothly interoperate with any other .NET language. F# supports both functional programming and procedural programming. It also fully supports object-oriented programming.

ML, Haskell, and F# are further discussed in Chapter 15.

2.5 The First Step Toward Sophistication: ALGOL 60

ALGOL 60 strongly influenced subsequent programming languages and is therefore of central importance in any historical study of languages.

2.5.1 Historical Background

ALGOL 60 was the result of efforts to design a universal programming language for scientific applications. By late 1954, the Laning and Zierler algebraic system had been in operation for over a year, and the first report on Fortran had been published. Fortran became a reality in 1957, and several other high-level languages were being developed. Most notable among them were IT, which was designed by Alan Perlis at Carnegie Tech, and two languages for the UNIVAC computers, MATH-MATIC and UNICODE. The proliferation of languages made program sharing among users difficult. Furthermore, the new languages were all growing up around single architectures, some for UNIVAC computers and some for IBM 700-series machines. In response to this blossoming of machine-dependent languages, several major computer user groups in the United States, including SHARE (the IBM scientific user group) and USE (UNIVAC Scientific Exchange, the large-scale UNIVAC scientific user group), submitted a petition to the Association for Computing Machinery (ACM) on May 10, 1957, to form a committee to study and recommend action to create a machine-independent scientific programming language. Although Fortran might have been a candidate, it could not become a universal language, because at the time it was solely owned by IBM.

Previously, in 1955, GAMM (a German acronym for Society for Applied Mathematics and Mechanics) had formed a committee to design one universal, machine-independent algorithmic language. The desire for this new language was in part due to the Europeans' fear of being dominated by IBM.

By late 1957, however, the appearance of several high-level languages in the United States convinced the GAMM subcommittee that their effort had to be widened to include the Americans, and a letter of invitation was sent to ACM. In April 1958, after Fritz Bauer of GAMM presented the formal proposal to ACM, the two groups officially agreed to a joint language design project.

2.5.2 Early Design Process

GAMM and ACM each sent four members to the first design meeting. The meeting, which was held in Zurich from May 27 to June 1, 1958, began with the following goals for the new language:

- The syntax of the language should be as close as possible to standard mathematical notation, and programs written in it should be readable with little further explanation.
- It should be possible to use the language for the description of algorithms in printed publications.
- Programs in the new language must be mechanically translatable into machine language.

The first goal indicated that the new language was to be used for scientific programming, which was the primary computer application area at that time. The second was something entirely new to the computing business. The last goal is an obvious necessity for any programming language.

The Zurich meeting succeeded in producing a language that met the stated goals, but the design process required innumerable compromises, both among individuals and between the two sides of the Atlantic. In some cases, the compromises were not so much over great issues as they were over spheres of influence. The question of whether to use a comma (the European method) or a period (the American method) for a decimal point is one example.

2.5.3 ALGOL 58 Overview

The language designed at the Zurich meeting was named the International Algorithmic Language (IAL). It was suggested during the design that the language be named ALGOL, for ALGOrithmic Language, but the name was rejected because it did not reflect the international scope of the committee. During the following year, however, the name was changed to ALGOL, and the language subsequently became known as ALGOL 58.

In many ways, ALGOL 58 was a descendant of Fortran, which is quite natural. It generalized many of Fortran's features and added several new constructs and concepts. Some of the generalizations had to do with the goal of not tying the language to any particular machine, and others were attempts to make the language more flexible and powerful. A rare combination of simplicity and elegance emerged from the effort.

ALGOL 58 formalized the concept of data type, although only variables that were not floating-point required explicit declaration. It added the idea of compound statements, which most subsequent languages incorporated. Some features of Fortran that were generalized were the following: Identifiers were allowed to have any length, as opposed to Fortran I's restriction to six or fewer characters; any number of array dimensions was allowed, unlike Fortran I's limitation to no more than three; the lower bound of arrays could be specified by the programmer, whereas in Fortran it was implicitly 1; nested selection statements were allowed, which was not the case in Fortran I.

ALGOL 58 acquired the assignment operator in a rather unusual way. Zuse used the form

- expression => variable

for the assignment statement in Plankalkül. Although Plankalkül had not yet been published, some of the European members of the ALGOL 58 committee were familiar with the language. The committee dabbled with the Plankalkül assignment form but, because of arguments about character set limitations,[4] the greater-than symbol was changed to a colon. Then, largely at the

insistence of the Americans, the whole statement was turned around to the Fortran form

4. The card punches of that time did not include the greater-than symbol.

- variable := expression

The Europeans preferred the opposite form, but that would be the reverse of Fortran.

2.5.4 Reception of the ALGOL 58 Report

In December 1958, publication of the ALGOL 58 report (Perlis and Samelson, 1958) was greeted with enthusiasm. In the United States, the new language was viewed more as a collection of ideas for programming language design than as a universal standard language. Actually, the ALGOL 58 report was not meant to be a finished product but rather a preliminary document for international discussion. Nevertheless, three major design and implementation efforts used the report as their basis. At the University of Michigan, the MAD language was born (Arden et al., 1961). The U.S. Naval Electronics Group produced the NELIAC language (Huskey et al., 1963). At System Development Corporation, JOVIAL was designed and implemented (Shaw, 1963). JOVIAL, an acronym for Jules' Own Version of the International Algebraic Language, represents the only language based on ALGOL 58 to achieve widespread use (Jules was Jules I. Schwartz, one of JOVIAL's designers). JOVIAL became widely used because it was the official scientific language for the U.S. Air Force for a quarter century.

The rest of the U.S. computing community was not so kind to the new language. At first, both IBM and its major scientific user group, SHARE, seemed to embrace ALGOL 58. IBM began an implementation shortly after the report was published, and SHARE formed a subcommittee, SHARE IAL, to study the language. The subcommittee subsequently recommended that ACM standardize ALGOL 58 and that IBM implement it for all of the 700-

series computers. The enthusiasm was short-lived, however. By the spring of 1959, both IBM and SHARE, through their Fortran experience, had had enough of the pain and expense of getting a new language started, both in terms of developing and using the first-generation compilers and in terms of training users in the new language and persuading them to use it. By the middle of 1959, both IBM and SHARE had developed such a vested interest in Fortran that they decided to retain it as *the* scientific language for the IBM 700-series machines, thereby abandoning ALGOL 58.

2.5.5 ALGOL 60 Design Process

During 1959, ALGOL 58 was furiously debated in both Europe and the United States. Large numbers of suggested modifications and additions were published in the European *ALGOL Bulletin* and in *Communications of the ACM*. One of the most important events of 1959 was the presentation of the work of the Zurich committee to the International Conference on Information Processing, for there Backus introduced his new notation for describing the syntax of programming languages, which later became known as BNF (Backus-Naur form). BNF is described in detail in Chapter 3.

In January 1960, the second ALGOL meeting was held, this time in Paris. The purpose of the meeting was to debate the 80 suggestions that had been formally submitted for consideration. Peter Naur of Denmark had become heavily involved in the development of ALGOL, even though he had not been a member of the Zurich group. It was Naur who created and published the *ALGOL Bulletin*. He spent a good deal of time studying Backus's paper that introduced BNF and decided that BNF should be used to describe formally the results of the 1960 meeting. After making a few relatively minor changes to BNF, he wrote a description of the new proposed language in BNF and handed it out to the members of the 1960 group at the beginning of the meeting.

2.5.6 ALGOL 60 Overview

Although the 1960 meeting lasted only six days, the modifications made to ALGOL 58 were dramatic. Among the most important new developments were the following:

- The concept of block structure was introduced. This allowed the programmer to localize parts of programs by introducing new data environments, or scopes.

- Two different means of passing parameters to subprograms were allowed: pass by value and pass by name.

- Procedures were allowed to be recursive. The ALGOL 58 description was unclear on this issue. Note that although recursion was new for the imperative languages, Lisp had already provided recursive functions in 1959.

- Stack-dynamic arrays were allowed. A stack-dynamic array is one for which the subscript range or ranges are specified by variables, so that the size of the array is set at the time storage is allocated to the array, which happens when the declaration is reached during execution. Stack-dynamic arrays are described in detail in Chapter 6.

Several features that might have had a dramatic impact on the success or failure of the language were proposed and rejected. Most important among these were input and output statements with formatting, which were omitted because they were thought to be machine-dependent.

The ALGOL 60 report was published in May 1960 (Naur, 1960). A number of ambiguities still remained in the language description, and a third meeting was scheduled for April 1962 in Rome to address the problems. At this meeting the group dealt only with problems; no additions to the language were allowed. The results of this meeting were published under the title "Revised Report on the Algorithmic Language ALGOL 60" (Backus et al., 1963).

2.5.7 Evaluation

In some ways, ALGOL 60 was a great success; in other ways, it was a dismal failure. It succeeded in becoming, almost immediately, the only acceptable formal means of communicating algorithms in computing literature, and it remained that for more than 20 years. Every imperative programming language designed since 1960 owes something to ALGOL 60. In fact, most are direct or indirect descendants; examples include PL/I, SIMULA 67, ALGOL 68, C, Pascal, Ada, C++, Java, and C#.

The ALGOL 58/ALGOL 60 design effort included a long list of firsts. It was the first time that an international group attempted to design a programming language. It was the first language that was designed to be machine independent. It was also the first language whose syntax was formally described. This successful use of the BNF formalism initiated several important fields of computer science: formal languages, parsing theory, and BNF-based compiler design. Finally, the structure of ALGOL 60 affected machine architecture. In the most striking example of this, an extension of the language was used as the systems language of a series of large-scale computers, the Burroughs B5000, B6000, and B7000 machines, which were designed with a hardware stack to implement efficiently the block structure and recursive subprograms of the language.

On the other side of the coin, ALGOL 60 never achieved widespread use in the United States. Even in Europe, where it was more popular than in the United States, it never became the dominant language. There are a number of reasons for its lack of acceptance. For one thing, some of the features of ALGOL 60 turned out to be too flexible; they made understanding difficult and implementation inefficient. The best example of this is the pass-by-name method of passing parameters to subprograms, which is explained in Chapter 9. The difficulties of implementing ALGOL 60 are evidenced by Rutishauser's statement in 1967 that few, if any, implementations included the full ALGOL 60 language (Rutishauser, 1967, p. 8).

The lack of input and output statements in the language was another major reason for its lack of acceptance. Implementation-dependent input/output made programs difficult to port to other computers.

Ironically, one of the most important contributions to computer science associated with ALGOL 60, BNF, was also a factor in its lack of acceptance.

Although BNF is now considered a simple and elegant means of syntax description, in 1960 it seemed strange and complicated.

Finally, although there were many other problems, the entrenchment of Fortran among users and the lack of support by IBM were probably the most important factors in ALGOL 60's failure to gain widespread use.

The ALGOL 60 effort was never really complete, in the sense that - ambiguities and obscurities were always a part of the language description (Knuth, 1967).

The following is an example of an ALGOL 60 program:

```
comment ALGOL 60 Example Program
 Input:  An integer, listlen, where listlen is less than
         100, followed by listlen-integer values
 Output: The number of input values that are greater than
         the average of all the input values ;
begin
  integer array intlist [1:99];
  integer listlen, counter, sum, average, result;
  sum := 0;
  result := 0;
  readint (listlen);
  if (listlen > 0) ^ (listlen < 100) then
    begin
comment Read input into an array and compute the average;
    for counter := 1 step 1 until listlen do
      begin
      readint (intlist[counter]);
      sum := sum + intlist[counter]
      end;
comment Compute the average;
    average := sum / listlen;
comment Count the input values that are > average;
    for counter := 1 step 1 until listlen do
      if intlist[counter] > average
        then result := result + 1;
comment Print result;
    printstring("The number of values > average is:");
    printint (result)
    end
  else
    printstring ("Error-input list length is not legal";
end
```

2.6 Computerizing Business Records: COBOL

The story of COBOL is, in a sense, the opposite of that of ALGOL 60. Although it has been used for nearly 60 years, COBOL has had little effect on the design of subsequent languages, except for PL/I. It may still be the most widely used language,5 although it is difficult to be sure one way or the other. Perhaps the most important reason why COBOL has had little influence is that few have attempted to design a new language for business applications since it appeared. That is due in part to how well COBOL's capabilities meet the needs of its application area. Another reason is that a great deal of growth in business computing over the past 30 years has occurred in small businesses. In these businesses, very little software development has taken place. Instead, most of the software used is purchased as off-the-shelf packages for various general business applications.

5. In the late 1990s, in a study associated with the Y2K problem, it was estimated that there were approximately 800 million lines of COBOL in regular use in the 22 square miles of Manhattan.

2.6.1 Historical Background

The beginning of COBOL is somewhat similar to that of ALGOL 60, in the sense that the language was designed by a committee of people meeting for relatively short periods of time. At the time, in 1959, the state of business computing was similar to the state of scientific computing several years earlier, when Fortran was being designed. One compiled language for business applications, FLOW-MATIC, had been implemented in 1957, but it belonged to one manufacturer, UNIVAC, and was designed for that company's computers. Another language, AIMACO, was being used by the U.S. Air Force, but it was only a minor variation of FLOW-MATIC. IBM had designed a programming language for business applications, COMTRAN

(COMmercial TRANslator), but it had not yet been implemented. Several other language design projects were being planned.

2.6.2 FLOW-MATIC

The origins of FLOW-MATIC are worth at least a brief discussion, because it was the primary progenitor of COBOL. In December 1953, Grace Hopper at Remington-Rand UNIVAC wrote a proposal that was indeed prophetic. It suggested that “mathematical programs should be written in mathematical notation, data processing programs should be written in English statements” (Wexelblat, 1981, p. 16). Unfortunately, in 1953, it was impossible to convince nonprogrammers that a computer could be made to understand English words. It was not until 1955 that a similar proposal had some hope of being funded by UNIVAC management, and even then it took a prototype system to do the final convincing. Part of this selling process involved compiling and running a small program, first using English keywords, then using French keywords, and then using German keywords. This demonstration was considered remarkable by UNIVAC management and was instrumental in their acceptance of Hopper’s proposal.

2.6.3 COBOL Design Process

The first formal meeting on the subject of a common language for business applications, which was sponsored by the Department of Defense, was held at the Pentagon on May 28 and 29, 1959 (exactly one year after the Zurich ALGOL meeting). The consensus of the group was that the language, then named CBL (Common Business Language), should have the following general characteristics: Most agreed that it should use English as much as possible, although a few argued for a more mathematical notation. The language must be easy to use, even at the expense of being less powerful, in order to broaden the base of those who could program computers. In addition to making the language easy to use, it was believed that the use of English would allow managers to read programs. Finally, the design should not be overly restricted by the problems of its implementation.

One of the overriding concerns at the meeting was that steps to create this universal language should be taken quickly, as a lot of work was already being done to create other business languages. In addition to the existing languages, RCA and Sylvania were working on their own business applications languages. It was clear that the longer it took to produce a universal language, the more difficult it would be for the language to become widely used. On this basis, it was decided that there should be a quick study of existing languages. For this task, the Short Range Committee was formed.

There were early decisions to separate the statements of the language into two categories—data description and executable operations—and to have statements in these two categories be in different parts of programs. One of the debates of the Short Range Committee was over the inclusion of subscripts. Many committee members argued that subscripts were too complex for the people in data processing, who were thought to be uncomfortable with mathematical notation. Similar arguments revolved around whether arithmetic expressions should be included. The final report of the Short Range Committee, which was completed in December 1959, described the language that was later named COBOL 60.

The language specification for COBOL 60, published by the Government Printing Office in April 1960 (Department of Defense, 1960), was described as “initial.” Revised versions were published in 1961 and 1962 (Department of Defense, 1961, 1962). The language was standardized by the American National Standards Institute (ANSI) group in 1968. The next three revisions were standardized by ANSI in 1974, 1985, and 2002. The language continues to evolve today.

2.6.4 Evaluation

The COBOL language originated a number of novel concepts, some of which eventually appeared in other languages. For example, the `DEFINE` verb of COBOL 60 was the first high-level language construct for macros. More important, hierarchical data structures (records), which first appeared in Plankalkül, were first implemented in COBOL. They have been included in most of the imperative languages designed since then. COBOL was also the

first language that allowed names to be truly connotative, because it allowed both long names (up to 30 characters) and word-connector characters (hyphens).

Overall, the data division is the strong part of COBOL's design, whereas the procedure division is relatively weak. Every variable is defined in detail in the data division, including the number of decimal digits and the location of the implied decimal point. File records are also described with this level of detail, as are lines to be output to a printer, which makes COBOL ideal for printing accounting reports. Perhaps the most important weakness of the original procedure division was in its lack of functions. Versions of COBOL prior to the 1974 standard also did not allow subprograms with parameters.

Our final comment on COBOL: It was the first programming language whose use was mandated by the Department of Defense (DoD). This mandate came after its initial development, because COBOL was not designed specifically for the DoD. In spite of its merits, COBOL probably would not have survived without that mandate. The poor performance of the early compilers simply made the language too expensive to use. Eventually, of course, compilers became more efficient and computers became much faster and cheaper and had much larger memories. Together, these factors allowed COBOL to succeed, inside and outside DoD. Its appearance led to the electronic mechanization of accounting, an important development by any measure.

The following is an example of a COBOL program. This program reads a file named `BAL-FWD-File` that contains inventory information about a certain collection of items. Among other things, each item record includes the number currently on hand (`BAL-ON-HAND`) and the item's reorder point (`BAL-REORDER-POINT`). The reorder point is the threshold number of items on hand at which more must be ordered. The program produces a list of items that must be reordered as a file named `REORDER-LISTING`.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRODUCE-REORDER-LISTING.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. DEC-VAX.
OBJECT-COMPUTER. DEC-VAX.
INPUT-OUTPUT SECTION.
```

```
FILE-CONTROL.
    SELECT BAL-FWD-FILE ASSIGN TO READER.
    SELECT REORDER-LISTING ASSIGN TO LOCAL-PRINTER.
DATA DIVISION.
FILE SECTION.
FD  BAL-FWD-FILE
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 80 CHARACTERS.
01  BAL-FWD-CARD.
    02 BAL-ITEM-NO          PICTURE IS 9(5).
    02 BAL-ITEM-DESC        PICTURE IS X(20).
    02 FILLER               PICTURE IS X(5).
    02 BAL-UNIT-PRICE       PICTURE IS 999V99.
    02 BAL-REORDER-POINT    PICTURE IS 9(5).
    02 BAL-ON-HAND          PICTURE IS 9(5).
    02 BAL-ON-ORDER         PICTURE IS 9(5).
    02 FILLER               PICTURE IS X(30).
FD  REORDER-LISTING
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 132 CHARACTERS.
01  REORDER-LINE.
    02 RL-ITEM-NO           PICTURE IS Z(5).
    02 FILLER               PICTURE IS X(5).
    02 RL-ITEM-DESC         PICTURE IS X(20).
    02 FILLER               PICTURE IS X(5).
    02 RL-UNIT-PRICE        PICTURE IS ZZZ.99.
    02 FILLER               PICTURE IS X(5).
    02 RL-AVAILABLE-STOCK   PICTURE IS Z(5).
    02 FILLER               PICTURE IS X(5).
    02 RL-REORDER-POINT     PICTURE IS Z(5).
    02 FILLER               PICTURE IS X(71).
WORKING-STORAGE SECTION.
01  SWITCHES.
    02 CARD-EOF-SWITCH      PICTURE IS X.
01  WORK-FIELDS.
    02 AVAILABLE-STOCK      PICTURE IS 9(5).
PROCEDURE DIVISION.
000-PRODUCE-REORDER-LISTING.
    OPEN INPUT BAL-FWD-FILE.
    OPEN OUTPUT REORDER-LISTING.
    MOVE "N" TO CARD-EOF-SWITCH.
    PERFORM 100-PRODUCE-REORDER-LINE
        UNTIL CARD-EOF-SWITCH IS EQUAL TO "Y".
    CLOSE BAL-FWD-File.
    CLOSE REORDER-LISTING.
    STOP RUN.
100-PRODUCE-REORDER-LINE.
    PERFORM 110-READ-INVENTORY-RECORD.
```

```
    IF CARD-EOF-SWITCH IS NOT EQUAL TO "Y"]
        PERFORM 120-CALCULATE-AVAILABLE-STOCK
        IF AVAILABLE-STOCK IS LESS THAN BAL-REORDER-POINT
            PERFORM 130-PRINT-REORDER-LINE.
110-READ-INVENTORY-RECORD.
    READ BAL-FWD-FILE RECORD
         AT END
             MOVE "Y" TO CARD-EOF-SWITCH.
120-CALCULATE-AVAILABLE-STOCK.
ADD BAL-ON-HAND BAL-ON-ORDER
    GIVING AVAILABLE-STOCK.
130-PRINT-REORDER-LINE.
    MOVE SPACE               TO REORDER-LINE.
    MOVE BAL-ITEM-NO         TO RL-ITEM-NO.
    MOVE BAL-ITEM-DESC       TO RL-ITEM-DESC.
    MOVE BAL-UNIT-PRICE      TO RL-UNIT-PRICE.
    MOVE AVAILABLE-STOCK     TO RL-AVAILABLE-STOCK.
    MOVE BAL-REORDER-POINT   TO RL-REORDER-POINT.
    WRITE REORDER-LINE.
```

2.7 The Beginnings of Timesharing: Basic

Basic (Mather and Waite, 1971) is another programming language that has enjoyed widespread use but has gotten little respect. Like COBOL, it has largely been ignored by computer scientists. Also, like COBOL, in its earliest versions it was inelegant and included only a meager set of control statements.

Basic was very popular on microcomputers in the late 1970s and early 1980s. This followed directly from two of the main characteristics of early versions of Basic. It was easy for beginners to learn, especially those who were not science oriented, and its smaller dialects could be implemented on computers with very small memories.6 When the capabilities of microcomputers grew and other languages were implemented, the use of Basic waned. A strong resurgence in the use of Basic began with the appearance of Visual Basic (Microsoft, 1991) in the early 1990s.

6. Some early microcomputers included Basic interpreters that resided in 4096 bytes of ROM.

2.7.1 Design Process

Basic (Beginner's All-purpose Symbolic Instruction Code) was originally designed at Dartmouth College (now Dartmouth University) in New Hampshire by two mathematicians, John Kemeny and Thomas Kurtz, who, in the early 1960s, developed compilers for a variety of dialects of Fortran and ALGOL 60. Their science students generally had little trouble learning or using those languages in their studies. However, Dartmouth was primarily a liberal arts institution, where science and engineering students made up only about 25 percent of the student body. It was decided in the spring of 1963 to design a new language especially for liberal arts students. This new language

would use terminals as the method of computer access. The goals of the system were as follows:

1. It must be easy for nonscience students to learn and use.
2. It must be “pleasant and friendly.”
3. It must provide fast turnaround for homework.
4. It must allow free and private access.
5. It must consider user time more important than computer time.

The last goal was indeed a revolutionary concept. It was based at least partly on the belief that computers would become significantly cheaper as time went on, which they certainly did.

The combination of the second, third, and fourth goals led to the time-shared aspect of Basic. Only with individual access through terminals by numerous simultaneous users could these goals be met in the early 1960s.

In the summer of 1963, Kemeny began work on the compiler for the first version of Basic, using remote access to a GE 225 computer. Design and coding of the operating system for Basic began in the fall of 1963. At 4:00 a.m. on May 1, 1964, the first program using the timeshared Basic was typed in and run. In June, the number of terminals on the system grew to 11, and by the fall it had ballooned to 20.

2.7.2 Language Overview

The original version of Basic was very small and, oddly, was not interactive: There was no way for an executing program to get input data from the user. Programs were typed in, compiled, and run, in a sort of batch-oriented way. The original Basic had only 14 different statement types and a single data type—floating-point. Because it was believed that few of the targeted users would appreciate the difference between integer and floating-point types, the type was referred to as “numbers.” Overall, it was a very limited language,

though quite easy to learn.

2.7.3 Evaluation

The most important aspect of the original Basic was that it was the first widely used language that was used through terminals connected to a remote computer.7 Terminals had just begun to be available at that time. Before then, most programs were entered into computers through either punched cards or paper tape.

7. Lisp initially was used through terminals, but it was not widely used in the early 1960s.

Much of the design of Basic came from Fortran, with some minor influence from the syntax of ALGOL 60. Later, it grew in a variety of ways, with little or no effort made to standardize it. The American National Standards Institute issued a Minimal Basic standard (ANSI, 1978b), but this represented only the bare minimum of language features. In fact, the original Basic was very similar to Minimal Basic.

Although it may seem surprising, Digital Equipment Corporation used a rather elaborate version of Basic named Basic-PLUS to write significant portions of their largest operating system for the PDP-11 minicomputers, RSTS, in the 1970s.

Basic has been criticized for the poor structure of programs written in it, among other things. By the evaluation criteria discussed in Chapter 1, specifically readability and reliability, the language does indeed fare very poorly. Clearly, the early versions of the language were not meant for and should not have been used for serious programs of any significant size. Later versions are much better suited to such tasks.

The resurgence of Basic in the 1990s was driven by the appearance of Visual Basic (VB). VB became widely used in large part because it provided a simple way of building graphical user interfaces (GUIs), hence the name Visual Basic. When .NET appeared, a new version of VB came with it,

VB.NET. Although it was a significant departure from earlier versions of VB, it quickly displaced the older language. Perhaps the most important difference between VB and the .NET version is that the later version fully supports object-oriented programming.

The following is an example of a Basic program:

```
REM  Basic Example Program
REM  Input:  An integer, listlen, where listlen is less
REM          than 100, followed by listlen-integer values
REM  Output: The number of input values that are greater
REM          than the average of all input values
  DIM intlist(99)
  result = 0
  sum = 0
  INPUT listlen
  IF listlen > 0 AND listlen < 100 THEN
REM  Read input into an array and compute the sum
   FOR counter = 1 TO listlen
     INPUT intlist(counter)
     sum = sum + intlist(counter)
   NEXT counter
REM  Compute the average
   average = sum / listlen
REM  Count the number of input values that are > average
   FOR counter = 1 TO listlen
     IF intlist(counter) > average
        THEN result = result + 1
   NEXT counter
REM  Print the result
    PRINT "The number of values that are > average is:";
          result
  ELSE
   PRINT "Error-input list length is not legal"
  END IF
END
```

interview

User Design and Language Design

ALAN COOPER

Best-selling author of About Face: The Essentials of User Interface Design, Alan Cooper also had a large hand in designing what can be touted as the language with the most concern for user interface design, Visual Basic. For him, it all comes down to a vision for humanizing technology.

SOME INFORMATION ON THE BASICS

How did you get started in all of this? I'm a high school dropout with an associate degree in programming from a California community college. My first job was as a programmer for American President Lines (one of the United States' oldest ocean transportation companies) in San Francisco. Except for a few months here and there, I've remained self-employed.

What is your current job? Founder and chairman of Cooper, the company that humanizes technology (www.cooper.com).

What is or was your favorite job? Interaction design consultant.

You are very well known in the fields of language design and user interface design. Any thoughts on designing languages versus designing software, versus designing anything else? It's pretty much the same in the world of software: Know your user.

ABOUT THAT EARLY WINDOWS RELEASE

In the 1980s, you started using Windows and have talked about being lured by its plusses: the graphical user interface support and the dynamically linked library that let you create tools that configured themselves. What about the parts of Windows that you eventually helped shape? I was very impressed by Microsoft's inclusion of support for practical multitasking in Windows. This included dynamic relocation and interprocess communications.

MSDOS.exe was the shell program for the first few releases of Windows. It was a terrible program, and I believed that it could be improved dramatically, and I was the guy to do it. In my spare time, I immediately began to write a better shell program than the one Windows came with. I called it Tripod. Microsoft's original shell, MSDOS.exe, was one of the main stumbling blocks to the initial success of Windows. Tripod attempted to solve the problem by being easier to use and to configure.

When was that "Aha!" moment? It wasn't until late in 1987, when I was interviewing a corporate client, that the key design strategy for Tripod popped into my head. As the IS manager explained to me his need to create and publish a wide range of shell solutions to his disparate user base, I realized the conundrum that there is no such thing as an ideal shell. Every user would need their own personal shell, configured to their own needs and skill levels. In an instant, I perceived the solution to the shell design problem: It would be a shell construction set; a tool where each user would be able to construct exactly the shell that he or she needed for a unique mix of applications and training.

What is so compelling about the idea of a shell that can be individualized? Instead of me telling the users what the ideal shell was, they could design their own, personalized ideal shell. With a customizable shell, a programmer would create a shell that was powerful and wide ranging but also somewhat dangerous, whereas an IT manager would create a shell that could be given to a desk clerk that exposed only those few application-specific tools that the clerk used.

> *"MSDOS.exe was the shell program for the first few releases of Windows. It was a terrible program, and I believed that it could be improved dramatically, and I was the guy to do it. In my spare time, I immediately began to write a better shell program than the one Windows came with."*

How did you get from writing a shell program to collaborating with Microsoft? Tripod and Ruby are the same thing. After I signed a deal with Bill Gates, I changed the name of the prototype from Tripod to Ruby. I then used the Ruby prototype as prototypes should be used: as a disposable model for constructing release-quality code. Which is what I did. MS took the release version of Ruby and added QuickBasic to it, creating VB. All of those original innovations were in Tripod/Ruby.

RUBY AS THE INCUBATOR FOR VISUAL BASIC

Let's revisit your interest in early Windows and that DLL feature. The DLL wasn't a thing, it was a facility in the OS. It allowed a programmer to build code objects that could be linked to at run time as opposed to only at compile time. This is what allowed me to invent the dynamically extensible parts of VB, where controls can be added by third-party vendors.

The Ruby product embodied many significant advances in software design, but two of them stand out as exceptionally successful. As I mentioned, the dynamic linking capability of Windows had always intrigued me, but having the tools and knowing what to do with them were two different things. With

Ruby, I finally found two practical uses for dynamic linking, and the original program contained both. First, the language was both installable and could be extended dynamically. Second, the palette of gizmos could be added to dynamically.

Was your language in Ruby the first to have a dynamic linked library and to be linked to a visual front end? As far as I know, yes.

Using a simple example, what would this enable a programmer to do with his or her program? Purchase a control, such as a grid control, from a third-party vendor, install it on his or her computer, and have the grid control appear as an integral part of the language, including the visual programming front end.

Why do they call you "the father of Visual Basic"? Ruby came with a small language, one suited only for executing the dozen or so simple commands that a shell program needs. However, this language was implemented as a chain of DLLs, any number of which could be installed at run time. The internal parser would identify a verb and then pass it along the chain of DLLs until one of them acknowledged that it knew how to process the verb. If all of the DLLs passed, there was a syntax error. From our earliest discussions, both Microsoft and I had entertained the idea of growing the language, possibly even replacing it altogether with a "real" language. C was the candidate most frequently mentioned, but eventually, Microsoft took advantage of this dynamic interface to unplug our little shell language and replace it entirely with Quick-Basic. This new marriage of language to visual front end was static and permanent, and although the original dynamic interface made the coupling possible, it was lost in the process.

SOME FINAL COMMENTS ON NEW IDEAS

In the world of programming and programming tools, including languages and environments, what projects most interest you? I'm interested in creating programming tools that are designed to help users

instead of programmers.

What's the most critical rule, famous quote, or design idea to keep in mind? Bridges are not built by engineers. They are built by ironworkers.

Similarly, software programs are not built by engineers. They are built by programmers.

2.8 Everything for Everybody: PL/I

PL/I represents the first large-scale attempt to design a language that could be used for a broad spectrum of application areas. All previous and most subsequent languages have focused on one particular application area, such as science, artificial intelligence, or business.

2.8.1 Historical Background

Like Fortran, PL/I was developed as an IBM product. By the early 1960s, the users of computers in industry had settled into two separate and quite different camps: scientific and business. From the IBM point of view, scientific programmers could use either the large-scale 7090 or the small-scale 1620 IBM computers. This group used floating-point data and arrays extensively. Fortran was the primary language, although some assembly language also was used. They had their own user group, SHARE, and had little contact with anyone who worked on business applications.

For business applications, people used the large 7080 or the small 1401 IBM computers. They needed the decimal and character string data types, as well as elaborate and efficient input and output facilities. They used COBOL, although in early 1963 when the PL/I story begins, the conversion from assembly language to COBOL was far from complete. This category of users also had its own user group, GUIDE, and seldom had contact with scientific users.

In early 1963, IBM planners perceived the beginnings of a change in this situation. The two widely separated computer user groups were moving toward each other in ways that were thought certain to create problems. Scientists began to gather large files of data to be processed. This data required more sophisticated and more efficient input and output facilities. Business applications people began to use regression analysis to build management information systems, which required floating-point data and

arrays. It began to appear that computing installations would soon require two separate computers and technical staffs, supporting two very different programming languages.8

8. At the time, large computer installations required both full-time hardware and full-time system software maintenance staff.

These perceptions naturally led to the concept of designing a single universal computer that would be capable of doing both floating-point and decimal arithmetic, and therefore both scientific and business applications. Thus was born the concept of the IBM System/360 line of computers. Along with this came the idea of a programming language that could be used for both business and scientific applications. For good measure, features to support systems programming and list processing were thrown in. Therefore, the new language was to replace Fortran, COBOL, Lisp, and the systems applications of assembly language.

2.8.2 Design Process

The design effort began when IBM and SHARE formed the Advanced Language Development Committee of the SHARE Fortran Project in October 1963. This new committee quickly met and formed a subcommittee called the 3 × 3 Committee, so named because it had three members from IBM and three from SHARE. The 3 × 3 Committee met for three or four days every other week to design the language.

As with the Short Range Committee for COBOL, the initial design was scheduled for completion in a remarkably short time. Apparently, regardless of the scope of a language design effort, in the early 1960s the prevailing belief was that it could be done in three months. The first version of PL/I, which was then named Fortran VI, was supposed to be completed by December, less than three months after the committee was formed. The committee pleaded successfully on two different occasions for extensions, moving the due date back to January and then to late February 1964.

The initial design concept was that the new language would be an extension

of Fortran IV, maintaining compatibility, but that goal was dropped quickly along with the name Fortran VI. Until 1965, the language was known as NPL (New Programming Language). The first published report on NPL was given at the SHARE meeting in March 1964. A more complete description followed in April, and the version that would actually be implemented was published in December 1964 (IBM, 1964) by the compiler group at the IBM Hursley Laboratory in England, which was chosen to do the implementation. In 1965, the name was changed to PL/I to avoid the confusion of the name NPL with the National Physical Laboratory in England. If the compiler had been developed outside the United Kingdom, the name might have remained NPL.

2.8.3 Language Overview

Perhaps the best single-sentence description of PL/I is that it included what were then considered the best parts of ALGOL 60 (recursion and block structure), Fortran IV (separate compilation with communication through global data), and COBOL 60 (data structures, input/output, and report-generating facilities), along with an extensive collection of new constructs, all somehow cobbled together. Because PL/I is no longer a popular language, we will not attempt, even briefly, to discuss all the features of the language, or even its most controversial constructs. Instead, we will mention some of the language's contributions to the pool of knowledge of programming languages.

PL/I was the first programming language to have the following facilities:

- Programs were allowed to create concurrently executing subprograms. Although this was a good idea, it was poorly developed in PL/I.
- It was possible to detect and handle 23 different types of exceptions, or run-time errors.
- Subprograms were allowed to be used recursively, but the capability could be disabled, allowing more efficient linkage for nonrecursive subprograms.

- Pointers were included as a data type.
- Cross-sections of arrays could be referenced. For example, the third row of a matrix could be referenced as if it were a single-dimensioned array.

2.8.4 Evaluation

Any evaluation of PL/I must begin by recognizing the ambitiousness of the design effort. In retrospect, it appears naive to think that so many constructs could have been combined successfully. However, that judgment must be tempered by acknowledging that there was little language design experience at the time. Overall, the design of PL/I was based on the premise that any construct that was useful and could be implemented should be included, with insufficient concern about how a programmer could understand and make effective use of such a collection of constructs and features. Edsger Dijkstra, in his Turing Award Lecture (Dijkstra, 1972), made one of the strongest criticisms of the complexity of PL/I: “I absolutely fail to see how we can keep our growing programs firmly within our intellectual grip when by its sheer baroqueness the programming language—our basic tool, mind you!—already escapes our intellectual control.”

In addition to the problem with the complexity due to its large size, PL/I suffered from a number of what are now considered to be poorly designed constructs. Among these were pointers, exception handling, and concurrency, although we must point out that in all cases, these constructs had not appeared in any previous language.

In terms of usage, PL/I must be considered at least a partial success. In the 1970s, it enjoyed significant use in both business and scientific applications. It was also widely used during that time as an instructional vehicle in colleges, primarily in several subset forms, such as PL/C (Cornell, 1977) and PL/CS (Conway and Constable, 1976).

The following is an example of a PL/I program:

```
/* PL/I PROGRAM EXAMPLE
 INPUT:   AN INTEGER, LISTLEN, WHERE LISTLEN IS LESS THAN
```

```
          100, FOLLOWED BY LISTLEN-INTEGER VALUES
 OUTPUT:  THE NUMBER OF INPUT VALUES THAT ARE GREATER THAN
          THE AVERAGE OF ALL INPUT VALUES */
PLIEX: PROCEDURE OPTIONS (MAIN);
  DECLARE INTLIST (1:99) FIXED.
  DECLARE (LISTLEN, COUNTER, SUM, AVERAGE, RESULT) FIXED;
  SUM = 0;
  RESULT = 0;
  GET LIST (LISTLEN);
  IF (LISTLEN > 0) & (LISTLEN < 100) THEN
    DO;
/* READ INPUT DATA INTO AN ARRAY AND COMPUTE THE SUM */
    DO COUNTER = 1 TO LISTLEN;
      GET LIST (INTLIST (COUNTER));
      SUM = SUM + INTLIST (COUNTER);
    END;
/* COMPUTE THE AVERAGE */
    AVERAGE = SUM / LISTLEN;
/* COUNT THE NUMBER OF VALUES THAT ARE > AVERAGE */
    DO COUNTER = 1 TO LISTLEN;
      IF INTLIST (COUNTER) > AVERAGE THEN
        RESULT = RESULT + 1;
    END;
/* PRINT RESULT */
    PUT SKIP LIST ('THE NUMBER OF VALUES > AVERAGE IS:');
    PUT LIST (RESULT);
    END;
ELSE
    PUT SKIP LIST ('ERROR-INPUT LIST LENGTH IS ILLEGAL');
END PLIEX;
```

2.9 Two Early Dynamic Languages: APL and SNOBOL

The structure of this section is different from that of the other sections because the languages discussed here are very different. Neither APL nor SNOBOL had much influence on later mainstream languages.9 Some of the interesting features of APL are discussed later in the book.

9. However, they have had some influence on some nonmainstream languages (J is based on APL, ICON is based on SNOBOL, and AWK is partially based on SNOBOL).

In appearance and in purpose, APL and SNOBOL are quite different. They share two fundamental characteristics, however: dynamic typing and dynamic storage allocation. Variables in both languages are essentially untyped. A variable acquires a type when it is assigned a value, at which time it assumes the type of the value assigned. Storage is allocated to a variable only when it is assigned a value, because before that there is no way to know the amount of storage that will be needed.

2.9.1 Origins and Characteristics of APL

APL (Brown et al., 1988) was designed around 1960 by Kenneth E. Iverson at IBM. It was not originally designed to be an implemented programming language but rather was intended to be a vehicle for describing computer architecture. APL was first described in the book from which it gets its name, *A Programming Language* (Iverson, 1962). In the mid-1960s, the first implementation of APL was developed at IBM.

APL has a large number of powerful operators that are specified with a large

number of symbols, which created a problem for implementors. Initially, APL was used through IBM printing terminals. These terminals had special optional print balls that provided the odd character set required by the -language. One reason APL has so many operators is that it provides a large number of unit operations on arrays. For example, the transpose of any matrix is done with a single operator. The large collection of operators provides very high expressivity but also makes APL programs difficult to read. Therefore, people think of APL as a language that is best used for "throw-away" programming. Although programs can be written quickly, they should be discarded after use because they are difficult to maintain.

APL has been around for over 55 years and is still used today, although not widely. Furthermore, it has not changed a great deal over its lifetime.

2.9.2 Origins and Characteristics of SNOBOL

SNOBOL (pronounced "snowball"; Griswold et al., 1971) was designed in the early 1960s by three people at Bell Laboratories: D. J. Farber, R. E. Griswold, and I. P. Polonsky (Farber et al., 1964). It was designed specifically for text processing. The heart of SNOBOL is a collection of powerful operations for string pattern matching. One of the early applications of SNOBOL was for writing text editors. Because the dynamic nature of SNOBOL makes it slower than alternative languages, it is no longer used for such programs. However, SNOBOL is still a live and supported language that is used for a variety of text-processing tasks in several different application areas.

2.10 The Beginnings of Data Abstraction: SIMULA 67

Although SIMULA 67 never achieved widespread use and had little impact on the programmers and computing of its time, some of the constructs it introduced make it historically important.

2.10.1 Design Process

Two Norwegians, Kristen Nygaard and Ole-Johan Dahl, developed the language SIMULA I between 1962 and 1964 at the Norwegian Computing Center (NCC) in Oslo. They were primarily interested in using computers for simulation but also worked in operations research. SIMULA I was designed exclusively for system simulation and was first implemented in late 1964 on a UNIVAC 1107 computer.

As soon as the SIMULA I implementation was completed, Nygaard and Dahl began efforts to extend the language by adding new features and modifying some existing constructs in order to make the language useful for general-purpose applications. The result of this work was SIMULA 67, whose design was first presented publicly in March 1967 (Dahl and Nygaard, 1967). We will discuss only SIMULA 67, although some of the features of interest in SIMULA 67 are also in SIMULA I.

2.10.2 Language Overview

SIMULA 67 is an extension of ALGOL 60, taking both block structure and the control statements from that language. The primary deficiency of ALGOL 60 (and other languages at that time) for simulation applications was the design of its subprograms. Simulation requires subprograms that are allowed to restart at the position where they previously stopped. Subprograms with

this kind of control are known as **coroutines** because the caller and called subprograms have a somewhat equal relationship with each other, rather than the rigid master/slave relationship they have in most imperative languages.

To provide support for coroutines in SIMULA 67, the class construct was developed. This was an important development because the concept of data abstraction began with it and data abstraction provides the foundation for object-oriented programming.

It is interesting to note that the important concept of data abstraction was not developed and attributed to the class construct until 1972, when Hoare (1972) recognized the connection.

2.11 Orthogonal Design: ALGOL 68

ALGOL 68 was the source of several new ideas in language design, some of which were subsequently adopted by other languages. We include it here for that reason, even though it never achieved widespread use in either Europe or the United States.

2.11.1 Design Process

The development of the ALGOL family did not end when the revised report on ALGOL 60 appeared in 1962, although it was six years until the next design iteration was published. The resulting language, ALGOL 68 (van Wijngaarden et al., 1969), was dramatically different from its predecessor.

One of the most interesting innovations of ALGOL 68 was one of its primary design criteria: orthogonality. Recall our discussion of orthogonality in Chapter 1. The use of orthogonality resulted in several innovative features of ALGOL 68, one of which is described in the following section.

2.11.2 Language Overview

One important result of orthogonality in ALGOL 68 was its inclusion of user-defined data types. Earlier languages, such as Fortran, included only a few basic data structures. PL/I included a larger number of data structures, which made it harder to learn and difficult to implement, but it still could not provide an appropriate data structure for every need.

The approach of ALGOL 68 to data structures was to provide a few primitive types and structures and allow the user to combine those primitives to define a large number of different structures. This provision for user-defined data types was carried over to some extent into all of the major imperative languages designed since then. User-defined data types are valuable because

they allow the user to design data abstractions that fit particular problems very closely. All aspects of data types are discussed in Chapter 6.

As another first in the area of data types, ALGOL 68 introduced the kind of dynamic arrays that will be termed *implicit heap-dynamic* in Chapter 5. A dynamic array is one in which the declaration does not specify subscript bounds. Assignments to a dynamic array cause allocation of required storage. In ALGOL 68, dynamic arrays are called **`flex`** arrays.

2.11.3 Evaluation

ALGOL 68 includes a significant number of features that had not been previously used. Its use of orthogonality, which some may argue was overdone, was nevertheless revolutionary.

ALGOL 68 repeated one of the sins of ALGOL 60, however, and it was an important factor in its limited popularity. The language was described using an elegant and concise but also unknown metalanguage. Before one could read the language-describing document (van Wijngaarden et al., 1969), he or she had to learn the new metalanguage, called van Wijngaarden grammars, which were far more complex than BNF. To make matters worse, the designers invented a collection of words to explain the grammar and the language. For example, keywords were called *indicants*, substring extraction was called *trimming*, and the process of a subprogram execution was called a *coercion of deproceduring*, which might be *meek*, *firm*, or something else.

It is natural to contrast the design of PL/I with that of ALGOL 68, because they appeared only a few years apart. ALGOL 68 achieved writability by the principle of orthogonality: a few primitive concepts and the unrestricted use of a few combining mechanisms. PL/I achieved writability by including a large number of fixed constructs. ALGOL 68 extended the elegant simplicity of ALGOL 60, whereas PL/I simply threw together the features of several languages to attain its goals. Of course, it must be remembered that the goal of PL/I was to provide a unified tool for a broad class of problems, whereas ALGOL 68 was targeted to a single class: scientific applications.

PL/I achieved far greater acceptance than ALGOL 68, due largely to IBM's promotional efforts and the problems of understanding and implementing ALGOL 68. Implementation was a difficult problem for both, but PL/I had the resources of IBM to apply to building a compiler. ALGOL 68 enjoyed no such benefactor.

2.12 Some Early Descendants of the ALGOLs

All imperative languages owe some of their design to ALGOL 60 and/or ALGOL 68. This section discusses some of the early descendants of these languages.

2.12.1 Simplicity by Design: Pascal

2.12.1.1 Historical Background

Niklaus Wirth (Wirth is pronounced “Virt”) was a member of the International Federation of Information Processing (IFIP) Working Group 2.1, which was created to continue the development of ALGOL in the mid-1960s. In August 1965, Wirth and C. A. R. (“Tony”) Hoare contributed to that effort by presenting to the group a somewhat modest proposal for additions and modifications to ALGOL 60 (Wirth and Hoare, 1966). The majority of the group rejected the proposal as being too small an improvement over ALGOL 60. Instead, a much more complex revision was developed, which eventually became ALGOL 68. Wirth, along with a few other group members, did not believe that the ALGOL 68 report should have been released, based on the complexity of both the language and the metalanguage used to describe it. This position later proved to have some validity because the ALGOL 68 documents, and therefore the language, were indeed found to be challenging by the computing community.

The Wirth and Hoare version of ALGOL 60 was named ALGOL-W. It was implemented at Stanford University and was used primarily as an instructional vehicle, but only at a few universities. The primary contributions of ALGOL-W were the value-result method of passing parameters and the **`case`** statement for multiple selection. The value-result

method is an alternative to ALGOL 60's pass-by-name method. Both are discussed in Chapter 9. The **case** statement is discussed in Chapter 8.

Wirth's next major design effort, again based on ALGOL 60, was his most successful: Pascal.10 The original published definition of Pascal appeared in 1971 (Wirth, 1971). This version was modified somewhat in the implementation process and is described in Wirth (1973). The features that are often ascribed to Pascal in fact came from earlier languages. For example, user-defined data types were introduced in ALGOL 68, the **case** statement in ALGOL-W, and Pascal's records are similar to those of COBOL and PL/I.

10. Pascal is named after Blaise Pascal, a seventeenth-century French philosopher and mathematician who invented the first mechanical adding machine in 1642 (among other things).

2.12.1.2 Evaluation

The largest impact of Pascal was on the teaching of programming. In 1970, most students of computer science, engineering, and science were introduced to programming with Fortran, although some universities used PL/I, languages based on PL/I, and ALGOL-W. By the mid-1970s, Pascal had become the most widely used language for this purpose. This was quite natural, because Pascal was designed specifically for teaching programming. It was not until the late 1990s that Pascal was no longer the most commonly used language for teaching programming in colleges and universities.

Because Pascal was designed as a teaching language, it lacks several features that are essential for many kinds of applications. The best example of this is the impossibility of writing a subprogram that takes as a parameter an array of variable length. Another example is the lack of any separate compilation capability. These deficiencies naturally led to many nonstandard dialects, such as Turbo Pascal.

Pascal's popularity, for both teaching programming and other applications, was based primarily on its remarkable combination of simplicity and expressivity. Although there are some insecurities in Pascal, it is still a

relatively safe language, particularly when compared with Fortran or C. By the mid-1990s, the popularity of Pascal was on the decline, both in industry and in universities, primarily due to the rise of Modula-2, Ada, and C++, all of which included features not available in Pascal.

The following is an example of a Pascal program:

```
{Pascal Example Program
 Input:   An integer, listlen, where listlen is less than
          100, followed by listlen-integer values
 Output:  The number of input values that are greater than
          the average of all input values }
program pasex (input, output);
  type intlisttype = array [1..99] of integer;
  var
    intlist : intlisttype;
    listlen, counter, sum, average, result : integer;
  begin
  result := 0;
  sum := 0;
  readln (listlen);
  if ((listlen > 0) and (listlen < 100)) then
    begin
{ Read input into an array and compute the sum }
  for counter := 1 to listlen do
    begin
    readln (intlist[counter]);
    sum := sum + intlist[counter]
    end;
{ Compute the average }
    average := sum / listlen;
{ Count the number of input values that are > average }
      for counter := 1 to listlen do
        if (intlist[counter] > average) then
          result := result + 1;
{ Print the result }
    writeln ('The number of values > average is:',
              result)
    end  { of the then clause of if (( listlen > 0 ... }
  else
    writeln ('Error-input list length is not legal')
end.
```

2.12.2 A Portable Systems

Language: C

Like Pascal, C contributed little to the previously known collection of language features, but it has been widely used over a long period of time. Although originally designed for systems programming, C is well suited for a wide variety of applications.

2.12.2.1 Historical Background

C's ancestors include CPL, BCPL, B, and ALGOL 68. CPL was developed at Cambridge University in the early 1960s. BCPL is a simple systems language, also developed at Cambridge, this time by Martin Richards in 1967 (Richards, 1969).

The first work on the UNIX operating system was done in the late 1960s by Ken Thompson at Bell Laboratories. The first version was written in assembly language. The first high-level language implemented under UNIX was B, which was based on BCPL. B was designed and implemented by Thompson in 1970.

Neither BCPL nor B is a typed language, which is an oddity among high-level languages, although both are much lower-level than a language such as Java. Being untyped means that all data are considered machine words, which, although simple, leads to many complications and insecurities. For example, there is the problem of specifying floating-point rather than integer arithmetic in an expression. In one implementation of BCPL, the variable operands of a floating-point operation were preceded by periods. Variable operands not preceded by periods were considered to be integers. An alternative to this would have been to use different symbols for the floating-point operators.

This problem, along with several others, led to the development of a new typed language based on B. Originally called NB but later named C, it was designed and implemented by Dennis Ritchie at Bell Laboratories in 1972 (Kernighan and Ritchie, 1978). In some cases through BCPL, and in other

cases directly, C was influenced by ALGOL 68. This is seen in its **`for`** and **`switch`** statements, in its assigning operators, and in its treatment of pointers.

The only “standard” for C in its first decade and a half was the book by Kernighan and Ritchie (1978).11 Over that time span, the language slowly evolved, with different implementors adding different features. In 1989, ANSI produced an official description of C (ANSI, 1989), which included many of the features that implementors had already incorporated into the language. This standard was updated in 1999 (ISO, 1999). This later version includes a few significant changes to the language. Among these are a complex data type, a Boolean data type, and C++-style comments (`//`). We will refer to the 1989 version, which has long been called ANSI C, as C89; we will refer to the 1999 version as C99.

11. This language is often referred to as “K & R C.”

2.12.2.2 Evaluation

C has adequate control statements and data-structuring facilities to allow its use in many application areas. It also has a rich set of operators that provide a high degree of expressiveness.

One of the most important reasons why C is both liked and disliked is its lack of complete type checking. For example, in versions before C99, functions could be written for which parameters were not type checked. Those who like C appreciate the flexibility; those who do not like it find it too insecure. A major reason for its great increase in popularity in the 1980s was that a compiler for it was part of the widely used UNIX operating system. This inclusion in UNIX provided an essentially free and quite good compiler that was available to programmers on many different kinds of computers.

The following is an example of a C program:

```
/* C Example Program
 Input:  An integer, listlen, where listlen is less than
         100, followed by listlen-integer values
 Output: The number of input values that are greater than
```

```
         the average of all input values */
int main (){
  int intlist[99], listlen, counter, sum, average, result;
  sum = 0;
  result = 0;
  scanf("%d", &listlen);
  if ((listlen > 0) && (listlen < 100)) {
/* Read input into an array and compute the sum */
    for (counter = 0; counter < listlen; counter++) {
     scanf("%d", &intlist[counter]);
     sum += intlist[counter];
    }
/* Compute the average */
    average = sum / listlen;
/* Count the input values that are > average */
     for  (counter = 0; counter < listlen; counter++)
       if  (intlist[counter] > average) result++;
/* Print result */
    printf("Number of values > average is:%d\n", result);
  }
  else
    printf("Error-input list length is not legal\n");
}
```

2.13 Programming Based on Logic: Prolog

Simply put, logic programming is the use of a formal logic notation to communicate computational processes to a computer. Predicate calculus is the notation used in current logic programming languages.

Programming in logic programming languages is nonprocedural. Programs in such languages do not state exactly *how* a result is to be computed but rather describe the necessary form and/or characteristics of the result. What is needed to provide this capability in logic programming languages is a concise means of supplying the computer with both the relevant information and an inferencing process for computing desired results. Predicate calculus supplies the basic form of communication to the computer, and the proof method, named **resolution**, developed first by Robinson (1965), supplies the inferencing technique.

2.13.1 Design Process

During the early 1970s, Alain Colmerauer and Phillippe Roussel in the - Artificial Intelligence Group at the University of Aix-Marseille, together with Robert Kowalski of the Department of Artificial Intelligence at the University of Edinburgh, developed the fundamental design of Prolog. The primary components of Prolog are a method for specifying predicate calculus propositions and an implementation of a restricted form of resolution. Both predicate calculus and resolution are described in Chapter 16. The first Prolog interpreter was developed at Marseille in 1972. The version of the language that was implemented is described in Roussel (1975). The name Prolog is from *pro*gramming *log*ic.

2.13.2 Language Overview

Prolog programs consist of collections of statements. Prolog has only a few kinds of statements, but they can be complex.

One common use of Prolog is as a kind of intelligent database. This application provides a simple framework for discussing the Prolog language.

The database of a Prolog program consists of two kinds of statements: facts and rules. The following are examples of fact statements:

```
mother(joanne, jake).
father(vern, joanne).
```

These state that `joanne` is the `mother` of `jake`, and `vern` is the `father` of `joanne`.

An example of a rule statement is

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
```

This states that it can be deduced that `X` is the `grandparent` of `Z` if it is true that `X` is the `parent` of `Y` and `Y` is the parent of `Z`, for some specific values for the variables `X`, `Y`, and `Z`.

The Prolog database can be interactively queried with goal statements, an example of which is

```
father(bob, darcie).
```

This asks if `bob` is the `father` of `darcie`. When such a query, or goal, is presented to the Prolog system, it uses its resolution process to attempt to determine the truth of the statement. If it can conclude that the goal is true, it displays “true.” If it cannot prove it, it displays “false.”

2.13.3 Evaluation

In the 1980s, there was a relatively small group of computer scientists who believed that logic programming provided the best hope for escaping from the complexity of imperative languages, and also from the enormous problem

of producing the large amount of reliable software that was needed. So far, however, there are two major reasons why logic programming has not become more widely used. First, as with some other nonimperative approaches, programs written in logic languages thus far have proven to be highly inefficient relative to equivalent imperative programs. Second, it has been determined that it is an effective approach for only a few relatively small areas of application: certain kinds of database management systems and some areas of AI.

There is a dialect of Prolog that supports object-oriented programming: Prolog++ (Moss, 1994). Logic programming and Prolog are described in greater detail in Chapter 16.

2.14 History's Largest Design Effort: Ada

The Ada language is the result of the most extensive and expensive language design effort ever undertaken. The following paragraphs briefly describe the evolution of Ada.

2.14.1 Historical Background

The Ada language was developed for the Department of Defense (DoD), so the state of their computing environment was instrumental in determining its form. By 1974, over half of the applications of computers in DoD were embedded systems. An embedded system is one in which the computer hardware is embedded in the device it controls or for which it provides services. Software costs were rising rapidly, primarily because of the increasing complexity of systems. More than 450 different programming languages were in use for DoD projects, and none of them was standardized by DoD. Every defense contractor could define a new and different language for every contract.12 Because of this language proliferation, application software was rarely reused. Furthermore, no software development tools were created (because they are usually language dependent). A great many languages were in use, but none was actually suitable for embedded systems applications. For these reasons, in 1974, the Army, Navy, and Air Force each independently proposed the development of a single high-level language for embedded systems.

12. This result was largely due to the widespread use of assembly language for embedded systems, along with the fact that most embedded systems used specialized processors.

2.14.2 Design Process

Noting this widespread interest, in January 1975, Malcolm Currie, director of Defense Research and Engineering, formed the High-Order Language Working Group (HOLWG), initially headed by Lt. Col. William Whitaker of the Air Force. The HOLWG had representatives from all of the military services and liaisons with Great Britain, France, and what was then West Germany. Its initial charter was to do the following:

- Identify the requirements for a new DoD high-level language.
- Evaluate existing languages to determine whether there was a viable candidate.
- Recommend adoption or implementation of a minimal set of programming languages.

In April 1975, the HOLWG produced the Strawman requirements document for the new language (Department of Defense, 1975a). This was distributed to military branches, federal agencies, selected industrial and university representatives, and interested parties in Europe.

The Strawman document was followed by Woodenman (Department of Defense, 1975b) in August 1975, Tinman (Department of Defense, 1976) in January 1976, Ironman (Department of Defense, 1977) in January 1977, and finally Steelman (Department of Defense, 1978) in June 1978.

After a tedious process, the many submitted proposals for the language were narrowed down to four finalists, all of which were based on Pascal. In May 1979, the Cii Honeywell/Bull language design proposal was chosen from the four finalists as the design that would be used. The Cii Honeywell/Bull design team in France, the only foreign competitor among the final four, was led by Jean Ichbiah.

In the spring of 1979, Jack Cooper of the Navy Materiel Command recommended the name for the new language, Ada, which was then adopted. The name commemorates Augusta Ada Byron (1815–1851), countess of Lovelace, mathematician, and daughter of poet Lord Byron. She is generally recognized as being the world's first programmer. She worked with Charles Babbage on his mechanical computers, the Difference and Analytical

Engines, writing programs for several numerical processes.

The design and the rationale for Ada were published by ACM in its *SIGPLAN Notices* (ACM, 1979) and distributed to a readership of more than 10,000 people. A public test and evaluation conference was held in October 1979 in Boston, with representatives from over 100 organizations from the United States and Europe. By November, more than 500 language reports had been received from 15 different countries. Most of the reports suggested small modifications rather than drastic changes and outright rejections. Based on the language reports, the next version of the requirements specification, the Stoneman document (Department of Defense, 1980), was released in February 1980.

A revised version of the language design was completed in July 1980 and was accepted as MIL-STD 1815, the standard *Ada Language Reference Manual*. The number 1815 was chosen because it was the year of the birth of Augusta Ada Byron. Another revised version of the *Ada Language Reference Manual* was released in July 1982. In 1983, the American National Standards Institute standardized Ada. This “final” official version is described in Goos and Hartmanis (1983). The Ada language design was then frozen for a minimum of five years.

2.14.3 Language Overview

This subsection briefly describes four of the major contributions of the Ada language.

Packages in the Ada language provide the means for encapsulating data objects, specifications for data types, and procedures. This, in turn, provides the support for the use of data abstraction in program design.

The Ada language includes extensive facilities for exception handling, which allow the programmer to gain control after any one of a wide variety of exceptions, or run-time errors, has been detected.

Program units can be generic in Ada. For example, it is possible to write a

sort procedure that uses an unspecified type for the data to be sorted. Such a generic procedure must be instantiated for a specified type before it can be used, which is done with a statement that causes the compiler to generate a version of the procedure with the given type. The availability of such generic units increases the range of program units that might be reused, rather than duplicated, by programmers.

The Ada language also provides for concurrent execution of special program units, named tasks, using the rendezvous mechanism. Rendezvous is the name of a method of intertask communication and synchronization.

2.14.4 Evaluation

Perhaps the most important aspects of the design of the Ada language to consider are the following:

- Because the design was competitive, there were no limits on participation.
- The Ada language embodies most of the concepts of software engineering and language design of the late 1970s. Although one can question the actual approaches used to incorporate these features, as well as the wisdom of including such a large number of features in a language, most agree that the features are valuable.
- Although most people did not anticipate it, the development of a compiler for the Ada language was a difficult task. Only in 1985, almost four years after the language design was completed, did truly usable Ada compilers begin to appear.

The most serious criticism of Ada in its first few years was that it was too large and too complex. In particular, Hoare (1981) stated that it should not be used for any application where reliability is critical, which is precisely the type of application for which it was designed. On the other hand, others have praised it as the epitome of language design for its time. In fact, even Hoare eventually softened his view of the language.

The following is an example of an Ada program:

```
-- Ada Example Program
-- Input:   An integer, List_Len, where List_Len is less
--          than 100, followed by List_Len-integer values
-- Output:  The number of input values that are greater
--          than the average of all input values
with Ada.Text_IO, Ada.Integer.Text_IO;
use Ada.Text_IO, Ada.Integer.Text_IO;
procedure Ada_Ex is
  type Int_List_Type is array (1..99) of Integer;
  Int_List : Int_List_Type;
  List_Len, Sum, Average, Result : Integer;
begin
  Result:= 0;
  Sum := 0;
  Get (List_Len);
  if (List_Len > 0) and (List_Len < 100) then
-- Read input data into an array and compute the sum
    for Counter := 1 .. List_Len loop
      Get (Int_List(Counter));
      Sum := Sum + Int_List(Counter);
    end loop;
-- Compute the average
    Average := Sum / List_Len;
-- Count the number of values that are > average
    for Counter := 1 .. List_Len loop
      if Int_List(Counter) > Average then
        Result:= Result+ 1;
      end if;
    end loop;
-- Print result
    Put ("The number of values > average is:");
    Put (Result);
    New_Line;
  else
    Put_Line ("Error-input list length is not legal");
  end if;
end Ada_Ex;
```

2.14.5 Ada 95 and Ada 2005

Two of the most important new features of Ada 95 are described briefly in the following paragraphs. In the remainder of the book, we will use the name

Ada 83 for the original version and Ada 95 (its actual name) for the later version when it is important to distinguish between the two versions. In discussions of language features common to both versions, we will use the name Ada. The Ada 95 standard language is defined in ARM (1995).

The type derivation mechanism of Ada 83 is extended in Ada 95 to allow adding new components to those inherited from a base class. This provides for inheritance, a key ingredient in object-oriented programming languages. Dynamic binding of subprogram calls to subprogram definitions is accomplished through subprogram dispatching, which is based on the tag value of derived types through classwide types. This feature provides for polymorphism, another principal feature of object-oriented programming.

The rendezvous mechanism of Ada 83 provided only a cumbersome and inefficient means of sharing data among concurrent processes. It was necessary to introduce a new task to control access to the shared data. The protected objects of Ada 95 offer an attractive alternative to this. The shared data is encapsulated in a syntactic structure that controls all access to the data, either by rendezvous or by subprogram call.

It is widely believed that the popularity of Ada 95 suffered because the Department of Defense stopped requiring its use in military software systems. There were, of course, other factors that hindered its growth in popularity. Most important among these was the widespread acceptance of C++ for object-oriented programming, which occurred before Ada 95 was released.

There were several additions to Ada 95 to get Ada 2005. Among these were interfaces, similar to those of Java, more control of scheduling algorithms, and synchronized interfaces.

Ada is widely used in both commercial and defense avionics, air traffic control, and rail transportation, as well as in other areas.

2.15 Object-Oriented Programming: Smalltalk

Smalltalk was the first programming language that fully supported object--oriented programming. It is therefore an important part of any discussion of the evolution of programming languages.

2.15.1 Design Process

The concepts that led to the development of Smalltalk originated in the Ph.D. dissertation work of Alan Kay in the late 1960s at the University of Utah (Kay, 1969). Kay had the remarkable foresight to predict the future availability of powerful desktop computers. Recall that the first microcomputer systems were not marketed until the mid-1970s, and they were only remotely related to the machines envisioned by Kay, which were seen to execute a million or more instructions per second and contain several megabytes of memory. Such machines, in the form of workstations, became widely available only in the early 1980s.

Kay believed that desktop computers would be used by nonprogrammers and thus would need very powerful human-interfacing capabilities. The computers of the late 1960s were largely batch oriented and were used exclusively by professional programmers and scientists. For use by nonprogrammers, Kay determined, a computer would have to be highly interactive and use sophisticated graphics in its user interface. Some of the graphics concepts came from the LOGO experience of Seymour Papert, in which graphics were used to aid children in the use of computers (Papert, 1980).

Kay originally envisioned a system he called Dynabook, which was meant to be a general information processor. It was based in part on the Flex language, which he had helped design. Flex was based primarily on SIMULA 67.

Dynabook used the paradigm of the typical desk, on which there are a number of papers, some partially covered. The top sheet is often the focus of attention, with the others temporarily out of focus. The display of Dynabook would model this scene, using screen windows to represent various sheets of paper on the desktop. The user would interact with such a display both through keystrokes and by touching the screen with his or her fingers. After the preliminary design of Dynabook earned him a Ph.D., Kay's goal became to see such a machine constructed.

Kay found his way to the Xerox Palo Alto Research Center (Xerox PARC) and presented his ideas on Dynabook. This led to his employment there and the subsequent birth of the Learning Research Group at Xerox. The first charge of the group was to design a language to support Kay's programming paradigm and implement it on the best personal computer then available. These efforts resulted in an "Interim" Dynabook, consisting of a Xerox Alto workstation and Smalltalk-72 software. Together, they formed a research tool for further development. A number of research projects were conducted with this system, including several experiments to teach programming to children. Along with the experiments came further developments, leading to a sequence of languages that ended with Smalltalk-80. As the language grew, so did the power of the hardware on which it resided. By 1980, both the language and the Xerox hardware nearly matched the early vision of Alan Kay.

2.15.2 Language Overview

The Smalltalk world is populated by nothing but objects, from integer constants to large complex software systems. All computing in Smalltalk is done by the same uniform technique: sending a message to an object to invoke one of its methods. A reply to a message is an object, which either returns the requested information or simply notifies the sender that the requested processing has been completed. The fundamental difference between a message and a subprogram call is this: A message is sent to a data object, specifically to one of the methods defined for the object. The called method is then executed, often modifying the data of the object to which the message was sent; a subprogram call is a message to the code of a

subprogram. Usually the data to be processed by the subprogram is sent to it as a parameter.[13]

13. Of course, a method call can also pass data to be processed by the called method.

In Smalltalk, object abstractions are classes, which are very similar to the classes of SIMULA 67. Instances of the class can be created and are then the objects of the program.

The syntax of Smalltalk is unlike that of most other programming language, in large part because of the use of messages, rather than arithmetic and logic expressions and conventional control statements. One of the Smalltalk control constructs is illustrated in the example in the next subsection.

2.15.3 Evaluation

Smalltalk has done a great deal to promote two separate aspects of computing: graphical user interfaces and object-oriented programming. The windowing systems that are now the dominant method of user interfaces to software systems grew out of Smalltalk. Today, the most significant software design methodologies and programming languages are object oriented. Although the origin of some of the ideas of object-oriented languages came from SIMULA 67, they reached maturation in Smalltalk. It is clear that Smalltalk's impact on the computing world is extensive and will be long-lived.

The following is an example of a Smalltalk class definition:

```
"Smalltalk Example Program"
"The following is a class definition, instantiations of which can
class name                    Polygon
superclass                    Object
instance variable names       ourPen
numSides
sideLength
"Class methods"
  "Create an instance"
```

```
new
   ^ super new getPen
"Get a pen for drawing polygons"
getPen
   ourPen <- Pen new defaultNib: 2
"Instance methods"
"Draw a polygon"
draw
   numSides timesRepeat: [ourPen go: sideLength;
                            turn: 360 // numSides]
"Set length of sides"
length: len
sideLength <- len
"Set number of sides"
sides: num
   numSides <- num
```

2.16 Combining Imperative and Object-Oriented Features: C++

The origins of C were discussed in Section 2.12; the origins of Simula 67 were discussed in Section 2.10; the origins of Smalltalk were discussed in Section 2.15. C++ builds language facilities, borrowed from Simula 67, on top of C to support much of what Smalltalk pioneered. C++ has evolved from C through a sequence of modifications to improve its imperative features and to add constructs to support object-oriented programming.

2.16.1 Design Process

The first step from C toward C++ was made by Bjarne Stroustrup at Bell Laboratories in 1980. The initial modifications to C included the addition of function parameter type checking and conversion and, more significantly, classes, which are related to those of SIMULA 67 and Smalltalk. Also included were derived classes, public/private access control of inherited components, constructor and destructor methods, and friend classes. During 1981, inline functions, default parameters, and overloading of the assignment operator were added. The resulting language was called C with Classes and is described in Stroustrup (1983).

It is useful to consider some goals of C with Classes. The primary goal was to provide a language in which programs could be organized as they could be organized in SIMULA 67—that is, with classes and inheritance. A second important goal was that there should be little or no performance penalty relative to C. For example, array index range checking was not even considered because a significant performance disadvantage, relative to C, would result. A third goal of C with Classes was that it could be used for every application for which C could be used, so virtually none of the features of C would be removed, not even those considered to be unsafe.

By 1984, this language was extended by the inclusion of virtual methods, which provide dynamic binding of method calls to specific method definitions, method name and operator overloading, and reference types. This version of the language was called C++. It is described in Stroustrup (1984).

In 1985, the first available implementation appeared: a system named Cfront, which translated C++ programs into C programs. This version of Cfront and the version of C++ it implemented were named Release 1.0. It is described in Stroustrup (1986).

Between 1985 and 1989, C++ continued to evolve, based largely on user reactions to the first distributed implementation. This next version was named Release 2.0. Its Cfront implementation was released in June 1989. The most important features added to C++ Release 2.0 were support for multiple inheritance (classes with more than one parent class) and abstract classes, along with some other enhancements. Abstract classes are described in Chapter 12.

Release 3.0 of C++ evolved between 1989 and 1990. It added templates, which provide parameterized types, and exception handling. The current version of C++, which was standardized in 1998, is described in ISO (1998).

In 2002, Microsoft released its .NET computing platform, which included a new version of C++, named Managed C++, or MC++. MC++ extends C++ to provide access to the functionality of the .NET Framework. The additions include properties, delegates, interfaces, and a reference type for garbage--collected objects. Properties are discussed in Chapter 11. Delegates are briefly discussed in the introduction to C# in Section 2.19. Because .NET does not support multiple inheritance, neither does MC++.

2.16.2 Language Overview

Because C++ has both functions and methods, it supports both procedural and object-oriented programming.

Operators in C++ can be overloaded, meaning the user can create operators

for existing operators on user-defined types. C++ methods can also be overloaded, meaning the user can define more than one method with the same name, provided either the numbers or types of their parameters are different.

Dynamic binding in C++ is provided by virtual methods. These methods define type-dependent operations, using overloaded methods, within a collection of classes that are related through inheritance. A pointer to an object of class A can also point to objects of classes that have class A as an ancestor. When this pointer points to an overloaded virtual method, the method of the current type is chosen dynamically.

Both methods and classes can be templated, which means that they can be parameterized. For example, a method can be written as a templated method to allow it to have versions for a variety of parameter types. Classes enjoy the same flexibility.

C++ supports multiple inheritance. The exception-handling constructs of C++ are discussed in Chapter 14.

2.16.3 Evaluation

C++ rapidly became and remains a widely used language. One factor in its popularity is the availability of good and inexpensive compilers. Another factor is that it is almost completely backward compatible with C (meaning that C programs, with few changes, can be compiled as C++ programs), and in most implementations it is possible to link C++ code with C code—and thus relatively easy for the many C programmers to learn C++. Finally, at the time C++ first appeared, when object-oriented programming began to receive widespread interest, C++ was the only available language that was suitable for large commercial software projects.

On the negative side, because C++ is a very large and complex language, it clearly suffers drawbacks similar to those of PL/I. It inherited most of the insecurities of C, which make it less safe than languages such as Ada and Java.

2.16.4 A Replacement for Objective-C, Swift

Beginning with MAC OS X in 2002, Apple systems software was written in Objective-C. Swift was developed by Apple as an improved replacement for Objective-C. Work on Swift began in 2010 by Chris Lattner. The language was introduced in 2014, with version 2 being introduced in 2015. The first version was proprietary, but the second is open source. It is currently implemented under all of Apple's operating systems, as well as on Linux.

Among the features of Swift are a tuple data type, an option type (variables of this type can have a special no-value value), protocols (similar to Java - interfaces), two categories of types, class and struct, which support reference types and value types, as in C#, generic types, no pointers, a safer switch - construct, in which the default is to not fall through to the next option, and all statement collections, including a single statement, must be enclosed in braces in all control constructs.

Statements need not be terminated with semicolons, unless there are two or more statements on the same line. Types of data need not be declared, as type inferencing is used. Unlike C and C++, assignment statements do not return a value, so using `x = 0` is not legal in a Boolean expression. This eliminates a common error in C and C++ programs, in which `x = 0` is typed instead of `x == 0`. Heap allocated objects are automatically reclaimed, using reference counters.

Swift programs can interact with Objective-C code and Swift uses the same libraries as that language. Swift is already the tenth most popular programming language, according to the TIOBE Community Report.

2.16.5 Another Related Language: Delphi

Delphi (Lischner, 2000) is a hybrid language, similar to C++ and Objective-C, in that it was created by adding object-oriented support, among other things, to an existing imperative language, in this case Pascal. Apple designed an object-oriented version of Pascal, named Object Pascal, but subsequently dropped the project. Borland, which had developed Turbo Pascal for Windows, also designed an object-oriented version of Pascal, based on Turbo Pascal, also named Object Pascal. For several reasons, Borland renamed Object Pascal as Delphi. The first product with that name, which included an integrated development environment (IDE), was released in 1995. Some consider the IDE to be Delphi and the underlying programming language to be Object Pascal. Other suppliers of similar products continue to refer to the language as Object Pascal.

Many of the differences between C++ and Delphi are a result of the predecessor language and the surrounding programming cultures from which they are derived. Because C is a powerful but potentially unsafe language, C++ also fits that description, at least in the areas of array subscript range checking, pointer arithmetic, and its numerous type coercions. Similarly, because Pascal is more elegant and safer than C, Delphi is more elegant and safer than C++. Delphi is also less complex than C++. For example, Delphi does not include user-defined operator overloading, generic subprograms, and parameterized classes, all of which are part of C++.

Delphi was designed by Anders Hejlsberg, who had previously developed the Turbo Pascal system. Hejlsberg, who moved to Microsoft in 1996, was the lead designer of C#.

2.17 An Imperative-Based Object-Oriented Language: Java

Java's designers started with C++, removed some constructs, changed some, and added a few others. The resulting language provides much of the power and flexibility of C++, but in a smaller, simpler, and safer language. Since that initial design, Java has grown considerably.

2.17.1 Design Process

Java, like many programming languages, was designed for an application for which there appeared to be no satisfactory existing language. In 1990, Sun Microsystems determined there was a need for a programming language for embedded consumer electronic devices, such as toasters, microwave ovens, and interactive TV systems. Reliability was one of the primary goals for such a language. It may not seem that reliability would be an important factor in the software for a microwave oven. If an oven had malfunctioning software, it probably would not pose a grave danger to anyone and most likely would not lead to large legal settlements. However, if the software in a particular model was found to be erroneous after a million units had been manufactured and sold, their recall would entail significant cost. Therefore, reliability *is* an important characteristic of the software in consumer electronic products.

After considering C and C++, it was decided that neither would be satisfactory for developing software for consumer electronic devices. Although C was relatively small, it did not provide support for object-oriented programming, which they deemed a necessity. C++ supported object-oriented programming, but it was judged to be too large and complex, in part because it also supported procedure-oriented programming. It was also believed that neither C nor C++ provided the necessary level of reliability. So, a new language, later named Java, was designed. Its design was guided by the fundamental goal of providing greater simplicity and reliability than

C++ was believed to provide.

Although the initial impetus for Java was consumer electronics, none of the products with which it was used in its early years were ever marketed. Starting in 1993, when the World Wide Web became widely used, and largely because of the new graphical browsers, Java was found to be a useful tool for Web programming. In particular, Java applets, which are relatively small Java programs that are interpreted in Web browsers and whose output can be included in displayed Web documents, quickly became very popular in the middle to late 1990s. In the first few years of Java popularity, the Web was its most common application.

The Java design team was headed by James Gosling, who had previously designed the UNIX emacs editor and the NeWS windowing system.

2.17.2 Language Overview

As we stated previously, Java is based on C++ but it was specifically designed to be smaller, simpler, and more reliable. Like C++, Java has both classes and primitive types. Java arrays are instances of a predefined class, whereas in C++ they are not, although many C++ users build wrapper classes for arrays to add features like index range checking, which is implicit in Java.

Java does not have pointers, but its reference types provide some of the capabilities of pointers. These references are used to point to class instances. All objects are allocated on the heap. References are always implicitly dereferenced, when necessary. So they behave more like ordinary scalar variables.

Java has a primitive Boolean type named **`boolean`**, used mainly for the control expressions of its control statements (such as **`if`** and **`while`**). Unlike C and C++, arithmetic expressions cannot be used for control expressions.

One significant difference between Java and many of its predecessors that support object-oriented programming, including C++, is that it is not possible to write stand-alone subprograms in Java. All Java subprograms are methods

and are defined in classes. Furthermore, methods can be called through a class or object only. One consequence of this is that while C++ supports both procedural and object-oriented programming, Java supports object-oriented programming only.

Another important difference between C++ and Java is that C++ supports multiple inheritance directly in its class definitions. Java supports only single inheritance of classes, although some of the benefits of multiple inheritance can be gained by using its interface construct.

Among the C++ constructs that were not copied into Java are structs and unions.

Java includes a relatively simple form of concurrency control through its **`synchronized`** modifier, which can appear on methods and blocks. In either case, it causes a lock to be attached. The lock ensures mutually exclusive access or execution. In Java, it is relatively easy to create concurrent processes, which in Java are called *threads*.

Java uses implicit storage deallocation for its objects, often called **garbage collection**. This frees the programmer from needing to delete objects explicitly when they are no longer needed. Programs written in languages that do not have garbage collection often suffer from what is sometimes called memory leakage, which means that storage is allocated but never deallocated. This can obviously lead to eventual depletion of all available storage. Object deallocation is discussed in detail in Chapter 6.

Unlike C and C++, Java includes assignment type coercions (implicit type conversions) only if they are widening (from a “smaller” type to a “larger” type). So **`int`** to **`float`** coercions are done across the assignment operator, but **`float`** to **`int`** coercions are not.

2.17.3 Evaluation

The designers of Java did well at trimming the excess and/or unsafe features of C++. For example, the elimination of half of the assignment coercions that

are done in C++ was clearly a step toward higher reliability. Index range checking of array accesses also makes the language safer. The addition of concurrency enhances the range of applications that can be written in the language, as do the class libraries for graphical user interfaces, database access, and networking.

Java's portability, at least in intermediate form, has often been attributed to the design of the language, but it is not. Any language can be translated to an intermediate form and "run" on any platform that has a virtual machine for that intermediate form. The price of this kind of portability is the cost of interpretation, which traditionally has been about an order of magnitude more than execution of machine code. The initial version of the Java interpreter, called the Java Virtual Machine (JVM), indeed was at least 10 times slower than equivalent compiled C programs. However, many Java programs are now translated to machine code before being executed, using Just-in-Time (JIT) compilers. This makes the efficiency of Java programs competitive with that of programs in conventionally compiled languages such as C++, at least when array index range checking is not considered.

The use of Java increased faster than that of any other programming language. Initially, this was due to its value in programming dynamic Web documents. Clearly, one of the reasons for Java's rapid rise to prominence is simply that programmers like its design. Some developers thought C++ was too large and complex to be practical and safe. Java offered them an alternative that has much of the power of C++, but in a simpler, safer language. Another reason is that the compiler/interpreter system for Java is free and easily obtained on the Web. Java is now widely used in a variety of different applications areas.

The most recent version of Java, Java SE8, appeared in 2014. Since the first version, significant features have been added to the language. Among these are an enumeration class, generics, a new iteration construct, lambda expressions, and numerous class libraries.

The following is an example of a Java program:

```
// Java Example Program
// Input:   An integer, listlen, where listlen is less
```

```
//          than 100, followed by length-integer values
// Output:  The number of input data that are greater than
//          the average of all input values
import java.io.*;
class IntSort {
public static void main(String args[]) throws IOException {
  DataInputStream in = new DataInputStream(System.in);
  int listlen,
      counter,
      sum = 0,
      average,
      result = 0;
  int[] intlist =  new int[99];
  listlen = Integer.parseInt(in.readLine());
  if ((listlen > 0) && (listlen < 100)) {
/* Read input into an array and compute the sum */
    for (counter = 0; counter < listlen; counter++) {
      intlist[counter] =
             Integer.valueOf(in.readLine()).intValue();
      sum += intlist[counter];
    }
/* Compute the average */
    average = sum / listlen;
/* Count the input values that are > average */
    for (counter = 0; counter < listlen; counter++)
      if (intlist[counter] > average) result++;
/* Print result */
      System.out.println(
          "\nNumber of values > average is:" + result);
    } //** end of then clause of if ((listlen > 0) ...
    else System.out.println(
              "Error-input list length is not legal\n");
  } //** end of method main
} //** end of class IntSort
```

2.18 Scripting Languages

Scripting languages have evolved over the past 35 years. Early scripting - languages were used by putting a list of commands, called a **script**, in a file to be interpreted. The first of these languages, named `sh` (for shell), began as a small collection of commands that were interpreted as calls to system subprograms that performed utility functions, such as file management and simple file filtering. To this were added variables, control flow statements, functions, and various other capabilities, and the result is a complete programming language. One of the most powerful and widely known of these is `ksh` (Bolsky and Korn, 1995), which was developed by David Korn at Bell Laboratories.

Another scripting language is `awk`, developed by Al Aho, Brian Kernighan, and Peter Weinberger at Bell Laboratories (Aho et al., 1988). `awk` began as a report-generation language but later became a more general-purpose language.

2.18.1 Origins and Characteristics of Perl

The Perl language, developed by Larry Wall, was originally a combination of `sh` and `awk`. Perl has grown significantly since its beginnings and is now a powerful, although still somewhat primitive, programming language. Although it is still often called a scripting language, it is actually more similar to a typical imperative language, since it is always compiled, at least into an intermediate language, before it is executed. Furthermore, it has all the constructs to make it applicable to a wide variety of areas of computational problems.

Perl has a number of interesting features, only a few of which are mentioned in this chapter and discussed later in the book.

Variables in Perl are statically typed and implicitly declared. There are three distinctive namespaces for variables, denoted by the first character of the variables' names. All scalar variable names begin with dollar signs (`$`), all array names begin with at signs (`@`), and all hash names (hashes are briefly described below) begin with percent signs (`%`). This convention makes variable names in programs more readable than those of most other programming languages.

Perl includes a large number of implicit variables. Some of them are used to store Perl parameters, such as the particular form of newline character or characters that are used in the implementation. Implicit variables are commonly used as default parameters to built-in functions and default operands for some operators. The implicit variables have distinctive—although cryptic—names, such as `$!` and `@_`. The implicit variables' names, like the user-defined variable names, use the three namespaces, so `$!` is the name of a scalar variable.

Perl's arrays have two characteristics that set them apart from the arrays of the common imperative languages. First, they have dynamic length, meaning that they can grow and shrink as needed during execution. Second, arrays can be sparse, meaning that there can be gaps between the elements. These gaps do not take space in memory, and the iteration statement used for arrays, **`foreach`**, iterates over the missing elements.

Perl includes associative arrays, which are called **hashes**. These data structures are indexed by strings and are implicitly controlled hash tables. The Perl system supplies the hash function and increases the size of the structure when necessary.

Perl is a powerful, but somewhat dangerous, language. Its scalar type stores both strings and numbers, which are normally stored in double-precision - floating-point form. Depending on the context, numbers may be coerced to strings and vice versa. If a string is used in numeric context and the string cannot be converted to a number, zero is used and there is no warning or error message provided for the user. This can lead to errors that are not detected by the compiler or run-time system. Array indexing cannot be checked, because there is no set subscript range for any array. References to nonexistent elements return **`undef`**, which is interpreted as zero in numeric

context. So, there is also no error detection in array element access.

Perl's initial use was as a UNIX utility for processing text files. It was and still is widely used as a UNIX system administration tool. When the World Wide Web appeared, Perl achieved widespread use as a common gateway interface language for use with the Web, although it is now rarely used for that purpose. Perl is used as a general-purpose language for a variety of applications, such as computational biology and artificial intelligence.

The following is an example of a Perl program:

```
# Perl Example Program
# Input:   An integer, $listlen, where $listlen is less
#          than 100, followed by $listlen-integer values.
# Output:  The number of input values that are greater than
#          the average of all input values.
($sum, $result) = (0, 0);
$listlen = <STDIN>;
if (($listlen > 0) && ($listlen < 100)) {
# Read input into an array and compute the sum
  for ($counter = 0; $counter < $listlen; $counter++) {
    $intlist[$counter] = <STDIN>;
  } #- end of for (counter ...
# Compute the average
  $average = $sum / $listlen;
# Count the input values that are > average
  foreach  $num (@intlist) {
    if ($num > $average) { $result++; }
  } #- end of foreach $num ...
# Print result
  print "Number of values > average is: $result \n";
} #- end of if (($listlen ...
else {
  print "Error--input list length is not legal \n";
}
```

2.18.2 Origins and Characteristics of JavaScript

Use of the Web exploded in the mid-1990s after the first graphical browsers

appeared. The need for computation associated with HTML documents, which by themselves are completely static, quickly became critical. Computation on the server side was made possible with the common gateway interface (CGI), which allowed HTML documents to request the execution of programs on the server, with the results of such computations returned to the browser in the form of HTML documents. Computation on the browser end became available with the advent of Java applets. Both of these approaches have now been replaced for the most part by newer technologies, primarily scripting languages.

JavaScript was originally developed by Brendan Eich at Netscape. Its original name was Mocha. It was later renamed LiveScript. In late 1995, LiveScript became a joint venture of Netscape and Sun Microsystems and its name was changed to JavaScript. JavaScript has gone through extensive evolution, moving from version 1.0 to version 1.5 by adding many new features and capabilities. A language standard for JavaScript was developed in the late 1990s by the European Computer Manufacturers Association (ECMA) as ECMA-262. This standard has also been approved by the International Standards Organization (ISO) as ISO-16262. Microsoft's version of JavaScript is named JScript .NET.

Although a JavaScript interpreter could be embedded in many different applications, its most common use is embedded in Web browsers. JavaScript code is embedded in HTML documents and interpreted by the browser when the documents are displayed. The primary uses of JavaScript in Web programming are to validate form input data and create dynamic HTML documents.

In spite of its name, JavaScript is related to Java only through the use of similar syntax. Java is strongly typed, but JavaScript is dynamically typed (see Chapter 5). JavaScript's character strings and its arrays have dynamic length. Because of this, array indices are not checked for validity, although this is required in Java. Java fully supports object-oriented programming, but JavaScript supports neither inheritance nor dynamic binding of method calls to methods.

One of the most important uses of JavaScript is for dynamically creating and modifying HTML documents. JavaScript defines an object hierarchy that

matches a hierarchical model of an HTML document, which is defined by the Document Object Model. Elements of an HTML document are accessed through these objects, providing the basis for dynamic control of the elements of documents.

Following is a JavaScript script for the problem previously solved in several languages in this chapter. Note that it is assumed that this script will be called from an HTML document and interpreted by a Web browser.

```
// example.js
//   Input:  An integer, listLen, where listLen is less
//           than 100, followed by listLen-numeric values
// Output:   The number of input values that are greater
//           than the average of all input values
var intList = new Array(99);
var listLen, counter, sum = 0, result = 0;
listLen = prompt (
        "Please type the length of the input list", "");
if ((listLen > 0) && (listLen < 100)) {
// Get the input and compute its sum
   for (counter = 0; counter < listLen; counter++) {
        intList[counter] = prompt (
                         "Please type the next number", "");
        sum += parseInt(intList[counter]);
   }
// Compute the average
   average = sum / listLen;
// Count the input values that are > average
   for (counter = 0; counter < listLen; counter++)
       if (intList[counter] > average) result++;
// Display the results
   document.write("Number of values > average is: ",
                  result, "<br />");
}  else
    document.write(
       "Error - input list length is not legal <br />");
```

2.18.3 Origins and Characteristics of PHP

PHP (Tatroe et al., 2013) was developed by Rasmus Lerdorf, an employee of

the Apache Group, in 1994. His initial motivation was to provide a tool to help track visitors to his personal Web site. In 1995, he developed a package called Personal Home Page Tools, which became the first publicly distributed version of PHP. Originally, PHP was an abbreviation for Personal Home Page. Later, its user community began using the recursive name PHP: Hypertext Preprocessor, which subsequently forced the original name into obscurity. PHP is now developed, distributed, and supported as an open-source product. PHP processors are resident on most Web servers.

PHP is an HTML-embedded server-side scripting language specifically designed for Web applications. PHP code is interpreted on the Web server when an HTML document in which it is embedded has been requested by a browser. PHP code usually produces HTML code as output, which replaces the PHP code in the HTML document. Therefore, a Web browser never sees PHP code.

PHP is similar to JavaScript in its syntactic appearance, the dynamic nature of its strings and arrays, and its use of dynamic typing. PHP's arrays are a combination of JavaScript's arrays and Perl's hashes.

The original version of PHP did not support object-oriented programming. Abstract classes, interfaces, destructors, and access controls for class members have since been added to the language.

PHP allows simple access to HTML form data, so form processing is easy with PHP. PHP provides support for many different database management systems. This makes it a useful language for building programs that need Web access to databases.

The current version of PHP is 7, released in 2015.

2.18.4 Origins and Characteristics of Python

Python (Lutz, 2013) is an object-oriented interpreted scripting language. Its

initial design was by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands in the early 1990s. Its development is being continued by the Python Software Foundation. Python is being used for the same kinds of applications as Perl: system administration and other relatively small computing tasks. Python is an open-source system that is available for most common computing platforms. The Python implementation is available at `www.python.org`, which also has extensive information regarding Python.

Python's syntax is not based directly on any commonly used language. It is type checked, but dynamically typed. Instead of arrays, Python includes three kinds of data structures: lists; immutable lists, which are called **tuples**; and hashes, which are called **dictionaries**. There is a collection of list methods, such as `append`, `insert`, `remove`, and `sort`, as well as a collection of methods for dictionaries, such as `keys`, `values`, `copy`, and `has_key`. Python also supports list comprehensions, which originated with the Haskell language. List comprehensions are discussed in Section 15.8.

Python is object oriented, includes the pattern-matching capabilities of Perl, and has exception handling. Garbage collection is used to reclaim objects when they are no longer needed.

Support for form processing is provided by the `cgi` module. Modules that support cookies, networking, and database access are also available.

Python includes support for concurrency with its threads, as well as support for network programming with its sockets. It also has more support for functional programming than other nonfunctional programming languages.

One of the more interesting features of Python is that it can be easily extended by any user. The modules that support the extensions can be written in any compiled language. Extensions can add functions, variables, and object types. These extensions are implemented as additions to the Python interpreter.

2.18.5 Origins and Characteristics

of Ruby

Ruby (Thomas et al., 2005) was designed by Yukihiro Matsumoto (aka Matz) in the early 1990s and released in 1996. Since then it has continually evolved. The motivation for Ruby was dissatisfaction of its designer with Perl and Python. Although both Perl and Python support object-oriented programming,14 neither is a pure object-oriented language, at least in the sense that each has primitive (nonobject) types and each supports functions.

14. Actually, Python's support for object-oriented programming is partial.

The primary characterizing feature of Ruby is that it is a pure object-oriented language, just as is Smalltalk. Every data value is an object and all operations are via method calls. The operators in Ruby are only syntactic mechanisms to specify method calls for the corresponding operations. Because they are methods, they can be redefined. All classes, predefined or user defined, can be subclassed.

Both classes and objects in Ruby are dynamic in the sense that methods can be dynamically added to either. This means that both classes and objects can have different sets of methods at different times during execution. So, different instantiations of the same class can behave differently. Collections of methods, data, and constants can be included in the definition of a class.

The syntax of Ruby is related to that of Eiffel and Ada. There is no need to declare variables, because dynamic typing is used. The scope of a variable is specified in its name: A variable whose name begins with a letter has local scope; one that begins with `@` is an instance variable; one that begins with `$` has global scope. A number of features of Perl are present in Ruby, including implicit variables with silly names, such as `$_`.

As is the case with Python, any user can extend and/or modify Ruby. Ruby is culturally interesting because it is the first programming language designed in Japan that has achieved relatively widespread use in the United States.

2.19 The Flagship .NET Language: C#

C#, along with the development platform .NET,15 was announced by Microsoft in 2000. In January 2002, production versions of both were released.

15. The .NET development system is briefly discussed in Chapter 1.

2.19.1 Design Process

C# is based on C++ and Java but includes some ideas from Delphi and Visual Basic. Its lead designer, Anders Hejlsberg, also designed Turbo Pascal and -Delphi, which explains the Delphi parts of the heritage of C#.

The purpose of C# is to provide a language for component-based software development, specifically for such development in the .NET Framework. In this environment, components from a variety of languages easily can be combined to form systems. All of the .NET languages, which include C#, VB.NET, Managed C++, F#, and JScript .NET,16 use the common type system (CTS). The CTS provides a common class library. All types in all five .NET languages inherit from a single class root, `System.Object`. Compilers that conform to the CTS specification create objects that can be combined into software systems. All .NET languages are compiled into the same intermediate form, Intermediate Language (IL).17 Unlike Java, however, the IL is never interpreted. A Just-in-Time compiler is used to translate IL into machine code before it is executed.

16. Many other languages have been modified to be .NET languages.

17. Initially, IL was called MSIL (Microsoft Intermediate Language), but apparently many people thought that name was too long.

2.19.2 Language Overview

Many believe that one of Java's most important advances over C++ lies in the fact that it excludes some of C++'s features. For example, C++ supports multiple inheritance, pointers, structs, **`enum`** types, operator overloading, and a goto statement, but Java includes none of these. The designers of C# obviously disagreed with this wholesale removal of features, because all of these except multiple inheritance have been included in C#.

To the credit of C#'s designers, however, in several cases, the C# version of a C++ feature has been improved. For example, the **enum** types of C# are safer than those of C++, because they are never implicitly converted to integers. This allows them to be more type safe. The **struct** type was changed significantly, resulting in a truly useful construct, whereas in C++ it serves virtually no purpose. C#'s structs are discussed in Chapter 12. C# takes a stab at improving the **switch** statement that is used in C, C++, and Java. C#'s switch is discussed in Chapter 8.

Although C++ includes function pointers, they share the lack of safety that is inherent in C++'s pointers to variables. C# includes a new type, delegates, which are both object-oriented and type-safe references to subprograms. Delegates are used for implementing event handlers, controlling the execution of threads, and callbacks.18 Callbacks are implemented in Java with interfaces; in C++, method pointers are used.

18. When an object calls a method of another object and needs to be notified when that method has completed its task, the called method calls its caller back. This is known as a callback.

In C#, methods can take a variable number of parameters, as long as they are all the same type. This is specified by the use of a formal parameter of array type, preceded by the **`params`** reserved word.

Both C++ and Java use two distinct typing systems: one for primitives and one for objects. In addition to being confusing, this leads to a frequent need to convert values between the two systems—for example, to put a primitive

value into a collection that stores objects. C# makes the conversion between values of the two typing systems partially implicit through the implicit boxing and unboxing operations, which are discussed in detail in Chapter 12.19

19. This feature was added to Java in Java 5.0.

Among the other features of C# are rectangular arrays, which are not supported in most programming languages, and a **`foreach`** statement, which is an iterator for arrays and collection objects. A similar **`foreach`** statement is found in Perl, PHP, and Java 5.0. Also, C# includes properties, which are an alternative to public data members. Properties are specified as data members with get and set methods, which are implicitly called when references and assignments are made to the associated data members.

C# has evolved continuously and quickly from its initial release in 2002. The most recent version is C# 7.0. New in C# 7.0 are tuples and a form of pattern matching.

2.19.3 Evaluation

C# was meant to be an improvement over both C++ and Java as a general--purpose programming language. Although it can be argued that some of its features are a step backward, C# includes some constructs that move it beyond its predecessors. Some of its features will surely be adopted by other programming languages of the future.

The following is an example of a C# program:

```
// C# Example Program
// Input:  An integer, listlen, where listlen is less than
//         100, followed by listlen-integer values.
// Output: The number of input values that are greater
//         than the average of all input values.
using System;
public class Ch2example {
  static void Main() {
    int[] intlist;
```

```
    int listlen,
        counter,
        sum = 0,
        average,
        result = 0;
    intList = new int[99];
    listlen = Int32.Parse(Console.readLine());
    if ((listlen > 0) && (listlen < 100)) {
// Read input into an array and compute the sum
      for (counter = 0; counter < listlen; counter++) {
        intList[counter] =
                        Int32.Parse(Console.readLine());
        sum += intList[counter];
      } //- end of for (counter ...
// Compute the average
      average = sum / listlen;
// Count the input values that are > average
      foreach (int num in intList)
        if (num > average) result++;
// Print result
      Console.WriteLine(
         "Number of values > average is:" + result);
    } //- end of if ((listlen ...
    else
      Console.WriteLine(
         "Error--input list length is not legal");
  } //- end of method Main
} //- end of class Ch2example
```

2.20 Markup-Programming Hybrid Languages

A markup-programming hybrid language is a markup language in which some of the elements can specify programming actions, such as control flow and computation. The following subsections introduce two such hybrid languages, XSLT and JSP.

2.20.1 XSLT

eXtensible markup language (XML) is a metamarkup language. Such a language is used to define markup languages. XML-derived markup languages are used to define XML data documents. Although XML documents are human readable, they are processed by computers. This processing sometimes consists only of transformations to alternative forms that can be effectively displayed or printed. In many cases, such transformations are to HTML, which can be displayed by a Web browser. In other cases, the data in the document is processed, just as with other forms of data files.

The transformation of XML documents to HTML documents is specified in another markup language, eXtensible stylesheet language transformations (XSLT) (`www.w3.org/TR/XSLT`). XSLT can specify programming-like operations. Therefore, XSLT is a markup-programming hybrid language. XSLT was defined by the World Wide Web Consortium (W3C) in the late 1990s.

An XSLT processor is a program that takes as input an XML data document and an XSLT document (which is also in the form of an XML document). In this processing, the XML data document is transformed to another XML document,20 using the transformations described in the XSLT document. The XSLT document specifies transformations by defining templates, which are

data patterns that could be found by the XSLT processor in the XML input file. Associated with each template in the XSLT document are its transformation instructions, which specify how the matching data is to be transformed before being put in the output document. So, the templates (and their associated processing) act as subprograms, which are “executed” when the XSLT processor finds a pattern match in the data of the XML document.

20. The output document of the XSLT processor could also be in HTML or plain text.

XSLT also has programming constructs at a lower level. For example, a looping construct is included, which allows repeated parts of the XML document to be selected. There is also a sort process. These lower-level constructs are specified with XSLT tags, such as `<for-each>`.

2.20.2 JSP

The “core” part of the Java Server Pages Standard Tag Library (JSTL) is another markup-programming hybrid language, although its form and purpose are different from those of XSLT. Before discussing JSTL, it is necessary to introduce the ideas of servlets and Java Server Pages (JSP). A **servlet** is an instance of a Java class that resides on and is executed on a Web server system. The execution of a servlet is requested by a markup document being displayed by a Web browser. The servlet’s output, which is in the form of an HTML document, is returned to the requesting browser. A program that runs in the Web server process, called a **servlet container**, controls the execution of servlets. Servlets are commonly used for form processing and for database access.

JSP is a collection of technologies designed to support dynamic Web documents and provide other processing needs of Web documents. When a JSP document, which is often a mixture of HTML and Java, is requested by a browser, the JSP processor program, which resides on a Web server system, converts the document to a servlet. The document’s embedded Java code is copied to the servlet. The plain HTML is copied into Java print statements that output it as is. The JSTL markup in the JSP document is processed, as

discussed in the following paragraph. The servlet produced by the JSP processor is run by the servlet container.

The JSTL defines a collection of XML action elements that control the processing of the JSP document on the Web server. These elements have the same form as other elements of HTML and XML. One of the most commonly used JSTL control action elements is `if`, which specifies a Boolean expression as an attribute.21 The content of the `if` element (the text between the opening tag (`<if>`) and its closing tag (`</if>`)) is HTML code that will be included in the output document only if the Boolean expression evaluates to true. The `if` element is related to the C/C++ `#if` preprocessor command. The JSP container processes the JSTL parts of JSP documents in a way that is similar to how the C/C++ preprocessor processes C and C++ programs. The preprocessor commands are instructions for the preprocessor to specify how the output file is to be constructed from the input file. Similarly, JSTL control action elements are instructions for the JSP processor to specify how to build the XML output file from the XML input file.

21. An attribute in HTML, which is embedded in the opening tag of an element, provides further information about that element.

One common use of the `if` element is for the validation of form data submitted by a browser user. Form data is accessible by the JSP processor and can be tested with the `if` element to ensure that it is sensible data. If not, the `if` element can insert an error message for the user in the output document.

For multiple selection control, JSTL has `choose`, `when`, and `otherwise` elements. JSTL also includes a `forEach` element, which iterates over collections, which typically are form values from a client. The `forEach` element can include `begin`, `end`, and `step` attributes to control its iterations.

SUMMARY

We have investigated the development of a number of programming languages. This chapter gives the reader a good perspective on current issues in language design. We have set the stage for an in-depth discussion of the important features of contemporary languages.

BIBLIOGRAPHIC NOTES

Perhaps the most important source of historical information about the development of early programming languages is *History of Programming Languages*, edited by Richard Wexelblat (1981). It contains the developmental background and environment of 13 important programming languages, as told by the designers themselves. A similar work resulted from a second "history" conference, published as a special issue of *ACM SIGPLAN Notices* (ACM, 1993a). In this work, the history and evolution of 13 more programming languages are discussed.

The paper "Early Development of Programming Languages" (Knuth and Pardo, 1977), which is part of the *Encyclopedia of Computer Science and Technology*, is an excellent 85-page work that details the development of languages up to and including Fortran. The paper includes example programs to demonstrate the features of many of those languages.

Another book of great interest is *Programming Languages: History and - Fundamentals*, by Jean Sammet (1969). It is a 785-page work filled with details of 80 programming languages of the 1950s and 1960s. Sammet has also published several updates to her book, such as *Roster of Programming Languages for 1974–75* (1976).

REVIEW QUESTIONS

1. In what year was Plankalkül designed? In what year was that design published?

2. What two common data structures were included in Plankalkül?

3. How were the pseudocodes of the early 1950s implemented?

4. Speedcoding was invented to overcome two significant shortcomings of the computer hardware of the early 1950s. What were they?

5. Why was the slowness of interpretation of programs acceptable in the early 1950s?

6. What hardware capability that first appeared in the IBM 704 computer strongly affected the evolution of programming languages? Explain why.

7. In what year was the Fortran design project begun?

8. What was the primary application area of computers at the time Fortran was designed?

9. What was the source of all of the control flow statements of Fortran I?

10. What was the most significant feature added to Fortran I to get Fortran II?

11. What control flow statements were added to Fortran IV to get Fortran 77?

12. Which version of Fortran was the first to have any sort of dynamic variables?

13. Which version of Fortran was the first to have character string handling?

14. Why were linguists interested in artificial intelligence in the late 1950s?

15. Where was Lisp developed? By whom?

16. In what way are Scheme and Common Lisp opposites of each other?

17. What dialect of Lisp is used for introductory programming courses at some universities?

18. What two professional organizations together designed ALGOL 60?

19. In what version of ALGOL did block structure appear?

20. What missing language element of ALGOL 60 damaged its chances for widespread use?

21. What language was designed to describe the syntax of ALGOL 60?

22. On what programming language was COBOL based?

23. In what year did the COBOL design process begin?

24. What data structure that appeared in COBOL originated with Plankalkül?

25. What organization was most responsible for the early success of COBOL (in terms of extent of use)?

26. What user group was the target of the first version of Basic?

27. Why was Basic an important language in the early 1980s?

28. PL/I was designed to replace what two languages?

29. For what new line of computers was PL/I designed?

30. What features of SIMULA 67 are now important parts of some object-oriented languages?

31. What innovation of data structuring was introduced in ALGOL 68 but is often credited to Pascal?

32. What design criterion was used extensively in ALGOL 68?

33. What language introduced the **`case`** statement?

34. What operators in C were modeled on similar operators in ALGOL 68?

35. What are two characteristics of C that make it less safe than Pascal?

36. What is a nonprocedural language?

37. What are the two kinds of statements that populate a Prolog database?

38. What is the primary application area for which Ada was designed?

39. What are the concurrent program units of Ada called?

40. What Ada construct provides support for abstract data types?

41. What populates the Smalltalk world?

42. What three concepts are the basis for object-oriented programming?

43. Why does C++ include the features of C that are known to be unsafe?

44. What language was Swift designed to replace?

45. What do the Ada and COBOL languages have in common?

46. What was the first application for Java?

47. What characteristic of Java is most evident in JavaScript?

48. How does the typing system of PHP and JavaScript differ from that of Java?

49. What array structure is included in C# but not in C, C++, or Java?

50. What two languages was the original version of Perl meant to replace?

51. For what application area is JavaScript most widely used?

52. What is the relationship between JavaScript and PHP, in terms of their use?

53. PHP's primary data structure is a combination of what two data structures from other languages?

54. What data structure does Python use in place of arrays?

55. What characteristic does Ruby share with Smalltalk?

56. What characteristic of Ruby's arithmetic operators makes them unique among those of other languages?

57. What deficiency of the **`switch`** statement of C is addressed with the changes made by C# to that statement?

58. What is the primary platform on which C# is used?

59. What are the inputs to an XSLT processor?

60. What is the output of an XSLT processor?

61. What element of the JSTL is related to a subprogram?

62. To what is a JSP document converted by a JSP processor?

63. Where are servlets executed?

PROBLEM SET

1. What features of Plankalkül do you think would have had the greatest influence on Fortran 0 if the Fortran designers had been familiar with Plankalkül?

2. Determine the capabilities of Backus's 701 Speedcoding system, and compare them with those of a contemporary programmable hand calculator.

3. Write a short history of the A-0, A-1, and A-2 systems designed by Grace Hopper and her associates.

4. Compare the facilities of Fortran 0 with those of the Laning and Zierler system.

5. Which of the three original goals of the ALGOL design committee, in your opinion, was most difficult to achieve at that time?

6. Make an educated guess as to the most common syntax error in Lisp programs.

7. Lisp began as a pure functional language but gradually acquired more and more imperative features. Why?

8. Describe in detail the three most important reasons, in your opinion, why ALGOL 60 did not become a very widely used language.

9. Why, in your opinion, did COBOL allow long identifiers when Fortran and ALGOL did not?

10. Outline the major motivation of IBM in developing PL/I.

11. Was IBM's assumption, on which it based its decision to develop PL/I, correct, given the history of computers and language developments since 1964?

12. Describe, in your own words, the concept of orthogonality in programming language design.

13. What is the primary reason why PL/I became more widely used than ALGOL 68?

14. What are the arguments both for and against the idea of a typeless language?

15. Are there any logic programming languages other than Prolog?

16. What is your opinion of the argument that languages that are too complex are too dangerous to use, and we should therefore keep all languages small and simple?

17. Do you think language design by committee is a good idea? Support your opinion.

18. Languages continually evolve. What sort of restrictions do you think are appropriate for changes in programming languages? Compare your answers with the evolution of Fortran.

19. Build a table identifying all of the major language developments, together with when they occurred, in what language they first appeared, and the identities of the developers.

20. There have been some public interchanges between Microsoft and Sun concerning the design of Microsoft's J++ and C# and Sun's Java. Read some of these documents, which are available on their respective Web sites, and write an analysis of the disagreements concerning the delegates.

21. In recent years data structures have evolved within scripting languages to replace traditional arrays. Explain the chronological sequence of these developments.

22. Explain two reasons why pure interpretation is an acceptable implementation method for several recent scripting languages.

23. Why, in your opinion, do new scripting languages appear more frequently than new compiled languages?

24. Give a brief general description of a markup-programming hybrid language.

PROGRAMMING EXERCISES

1. To understand the value of records in a programming language, write a small program in a C-based language that uses an array of structs that store student information, including name, age, GPA as a float, and grade level as a string (e.g., “freshmen,” etc.). Also, write the same program in the same language without using structs.

2. To understand the value of recursion in a programming language, write a program that implements quicksort, first using recursion and then without recursion.

3. To understand the value of counting loops, write a program that implements matrix multiplication using counting loop constructs. Then write the same program using only logical loops—for example, `while` loops.

3 Describing Syntax and Semantics

This chapter begins by defining the terms *syntax* and *semantics*. Then, a detailed discussion of the most common method of describing syntax, context-free grammars (also known as Backus-Naur Form), is presented. Included in this discussion are derivations, parse trees, ambiguity, descriptions of operator precedence and associativity, and extended Backus-Naur Form. Attribute grammars, which can be used to describe both the syntax and static semantics of programming languages, are discussed next. In the last section, three formal methods of describing semantics—operational, axiomatic, and denotational semantics—are introduced. Because of the inherent complexity of the semantics description methods, our discussion of them is brief. One could easily write an entire book on just one of the three (as several authors have).

3.1 Introduction

The task of providing a concise yet understandable description of a programming language is difficult but essential to the language's success. ALGOL 60 and ALGOL 68 were first presented using concise formal descriptions; in both cases, however, the descriptions were not easily understandable, partly because each used a new notation. The levels of acceptance of both languages suffered as a result. On the other hand, some languages have suffered the problem of having many slightly different dialects, a result of a simple but informal and imprecise definition.

One of the problems in describing a language is the diversity of the people who must understand the description. Among these are initial evaluators, implementors, and users. Most new programming languages are subjected to a period of scrutiny by potential users, often people within the organization that employs the language's designer, before their designs are completed. These are the initial evaluators. The success of this feedback cycle depends heavily on the clarity of the description.

Programming language implementors obviously must be able to determine how the expressions, statements, and program units of a language are formed, and also their intended effect when executed. The difficulty of the implementors' job is, in part, determined by the completeness and precision of the language description.

Finally, language users must be able to determine how to encode software solutions by referring to a language reference manual. Textbooks and courses enter into this process, but language manuals are usually the only authoritative printed information source about a language.

The study of programming languages, like the study of natural languages, can be divided into examinations of syntax and semantics. The **syntax** of a programming language is the form of its expressions, statements, and - program units. Its **semantics** is the meaning of those expressions, statements, and program units. For example, the syntax of a Java `while` statement is

- **`while`** (boolean_expr) statement

The semantics of this statement form is that when the current value of the Boolean expression is true, the embedded statement is executed. Then control implicitly returns to the Boolean expression to repeat the process. If the Boolean expression is false, control transfers to the statement following the **`while`** construct.

Although they are often separated for discussion purposes, syntax and semantics are closely related. In a well-designed programming language, semantics should follow directly from syntax; that is, the appearance of a statement should strongly suggest what the statement is meant to accomplish.

Describing syntax is easier than describing semantics, partly because a concise and universally accepted notation is available for syntax description, but none has yet been developed for semantics.

3.2 The General Problem of Describing Syntax

A language, whether natural (such as English) or artificial (such as Java), is a set of strings of characters from some alphabet. The strings of a language are called **sentences** or statements. The syntax rules of a language specify which strings of characters from the language's alphabet are in the language. English, for example, has a large and complex collection of rules for specifying the syntax of its sentences. By comparison, even the largest and most complex programming languages are syntactically very simple.

Formal descriptions of the syntax of programming languages, for simplicity's sake, often do not include descriptions of the lowest-level syntactic units. These small units are called **lexemes**. The description of lexemes can be given by a lexical specification, which is usually separate from the syntactic description of the language. The lexemes of a programming language include its numeric literals, operators, and special words, among others. One can think of programs as strings of lexemes rather than of characters.

Lexemes are partitioned into groups—for example, the names of variables, methods, classes, and so forth in a programming language form a group called *identifiers*. Each lexeme group is represented by a name, or token. So, a **token** of a language is a category of its lexemes. For example, an identifier is a token that can have lexemes, or instances, such as `sum` and `total`. In some cases, a token has only a single possible lexeme. For example, the token for the arithmetic operator symbol + has just one possible lexeme. Consider the following Java statement:

```
index = 2 * count + 17;
```

The lexemes and tokens of this statement are

Lexemes	*Tokens*
`index`	identifier
`=`	equal_sign
`2`	int_literal
`*`	mult_op
`count`	identifier
`+`	plus_op
`17`	int_literal
`;`	semicolon

The example language descriptions in this chapter are very simple, and most include lexeme descriptions.

3.2.1 Language Recognizers

In general, languages can be formally defined in two distinct ways: by **recognition** and by **generation** (although neither provides a definition that is practical by itself for people trying to learn or use a programming language). Suppose we have a language L that uses an alphabet ∑ of characters. To define L formally using the recognition method, we would need to construct a mechanism R, called a recognition device, capable of reading strings of characters from the alphabet ∑. R would indicate whether a given input string was or was not in L. In effect, R would either accept or reject the given string. Such devices are like filters, separating legal sentences from those that are incorrectly formed. If R, when fed any string of characters over ∑, accepts it only if it is in L, then R is a description of L. Because most useful languages are, for all practical purposes, infinite, this might seem like a

lengthy and ineffective process. Recognition devices, however, are not used to enumerate all of the sentences of a language—they have a different purpose.

The syntax analysis part of a compiler is a recognizer for the language the compiler translates. In this role, the recognizer need not test all possible strings of characters from some set to determine whether each is in the language. Rather, it need only determine whether given programs are in the language. In effect then, the syntax analyzer determines whether the given programs are syntactically correct. The structure of syntax analyzers, also known as parsers, is discussed in Chapter 4.

3.2.2 Language Generators

A language generator is a device that can be used to generate the sentences of a language. We can think of the generator as having a button that produces a sentence of the language every time it is pushed. Because the particular sentence that is produced by a generator when its button is pushed is unpredictable, a generator seems to be a device of limited usefulness as a language descriptor. However, people prefer certain forms of generators over recognizers because they can more easily read and understand them. By contrast, the syntax-checking portion of a compiler (a language recognizer) is not as useful a language description for a programmer because it can be used only in trial-and-error mode. For example, to determine the correct syntax of a particular statement using a compiler, the programmer can only submit a speculated version and note whether the compiler accepts it. On the other hand, it is often possible to determine whether the syntax of a particular statement is correct by comparing it with the structure of the generator.

There is a close connection between formal generation and recognition devices for the same language. This was one of the seminal discoveries in computer science, and it led to much of what is now known about formal languages and compiler design theory. We return to the relationship of generators and recognizers in the next section.

3.3 Formal Methods of Describing Syntax

This section discusses the formal language-generation mechanisms, usually called **grammars**, that are commonly used to describe the syntax of programming languages.

3.3.1 Backus-Naur Form and Context-Free Grammars

In the middle to late 1950s, two men, Noam Chomsky and John Backus, in unrelated research efforts, developed the same syntax description formalism, which subsequently became the most widely used method for programming language syntax.

3.3.1.1 Context-Free Grammars

In the mid-1950s, Noam Chomsky, a noted linguist (among other things), described four classes of generative devices or grammars that define four classes of languages (Chomsky, 1956, 1959). Two of these grammar classes, named *context-free* and *regular*, turned out to be useful for describing the syntax of programming languages. The forms of the tokens of programming languages can be described by regular grammars. The syntax of whole programming languages, with minor exceptions, can be described by context-free grammars. Because Chomsky was a linguist, his primary interest was the theoretical nature of natural languages. He had no interest at the time in the artificial languages used to communicate with computers. So it was not until later that his work was applied to programming languages.

3.3.1.2 Origins of Backus-Naur Form

Shortly after Chomsky's work on language classes, the ACM-GAMM group began designing ALGOL 58. A landmark paper describing ALGOL 58 was presented by John Backus, a prominent member of the ACM-GAMM group, at an international conference in 1959 (Backus, 1959). This paper introduced a new formal notation for specifying programming language syntax. The new notation was later modified slightly by Peter Naur for the description of ALGOL 60 (Naur, 1960). This revised method of syntax description became known as **Backus-Naur Form**, or simply **BNF**.

BNF is a natural notation for describing syntax. In fact, something similar to BNF was used by Panini to describe the syntax of Sanskrit several hundred years before Christ (Ingerman, 1967).

Although the use of BNF in the ALGOL 60 report was not immediately accepted by computer users, it soon became and is still the most popular method of concisely describing programming language syntax.

It is remarkable that BNF is nearly identical to Chomsky's generative devices for context-free languages, called **context-free grammars**. In the remainder of the chapter, we refer to context-free grammars simply as grammars. - Furthermore, the terms BNF and grammar are used interchangeably.

3.3.1.3 Fundamentals

A **metalanguage** is a language that is used to describe another language. BNF is a metalanguage for programming languages.

BNF uses abstractions for syntactic structures. A simple Java assignment statement, for example, might be represented by the abstraction <assign> (pointed brackets are often used to delimit names of abstractions). The actual definition of <assign> can be given by

- <assign> → <var> = <expression>

The text on the left side of the arrow, which is aptly called the **left-hand side** (LHS), is the abstraction being defined. The text to the right of the arrow is the definition of the LHS. It is called the **right-hand side** (RHS) and consists of some mixture of tokens, lexemes, and references to other abstractions. (Actually, tokens are also abstractions.) Altogether, the definition is called a **rule**, or **production**. In the example rule just given, the abstractions <var> and <expression> obviously must be defined for the <assign> definition to be useful.

This particular rule specifies that the abstraction <assign> is defined as an instance of the abstraction <var>, followed by the lexeme =, followed by an instance of the abstraction <expression>. One example sentence whose syntactic structure is described by the rule is

```
total = subtotal1 + subtotal2
```

The abstractions in a BNF description, or grammar, are often called **nonterminal symbols**, or simply **nonterminals**, and the lexemes and tokens of the rules are called **terminal symbols**, or simply **terminals**. A BNF description, or **grammar**, is a collection of rules.

Nonterminal symbols can have two or more distinct definitions, representing two or more possible syntactic forms in the language. Multiple definitions can be written as a single rule, with the different definitions separated by the symbol |, meaning logical OR. For example, a Java **`if`** statement can be described with the rules

- <if_stmt> → **`if`** (<logic_expr>) <stmt>
- <if_stmt> → **`if`** (<logic_expr>) <stmt> **`else`** <stmt>

or with the rule

- <if_stmt> → **`if`** (<logic_expr>) <stmt>
- | **`if`** (<logic_expr>) <stmt> **`else`** <stmt>

In these rules, <stmt> represents either a single statement or a compound statement.

Although BNF is simple, it is sufficiently powerful to describe nearly all of the syntax of programming languages. In particular, it can describe lists of similar constructs, the order in which different constructs must appear, and nested structures to any depth, and even imply operator precedence and operator associativity.

3.3.1.4 Describing Lists

Variable-length lists in mathematics are often written using an ellipsis (. . .); 1, 2, . . . is an example. BNF does not include the ellipsis, so an alternative method is required for describing lists of syntactic elements in programming languages (for example, a list of identifiers appearing on a data declaration statement). For BNF, the alternative is recursion. A rule is **recursive** if its LHS appears in its RHS. The following rules illustrate how recursion is used to describe lists:

- <ident_list> → identifier
- | identifier, <ident_list>

This defines <ident_list> as either a single token (identifier) or an identifier followed by a comma and another instance of <ident_list>. Recursion is used to describe lists in many of the example grammars in the remainder of this chapter.

3.3.1.5 Grammars and Derivations

A grammar is a generative device for defining languages. The sentences of the language are generated through a sequence of applications of the rules, beginning with a special nonterminal of the grammar called the **start symbol**. This sequence of rule applications is called a **derivation**. In a grammar for a complete programming language, the start symbol represents a complete

program and is often named <program>. The simple grammar shown in Example 3.1 is used to illustrate derivations.

EXAMPLE 3.1 A Grammar for a Small Language

- <program> → **begin** <stmt_list> **end**
- <stmt_list> → <stmt>
- | <stmt> ; <stmt_list>
- <stmt> → <var> = <expression>
- <var> → A | B | C
- <expression> → <var> + <var>
- | <var> - <var>
- | <var>

The language described by the grammar of Example 3.1 has only one statement form: assignment. A program consists of the special word **begin**, followed by a list of statements separated by semicolons, followed by the special word **end**. An expression is either a single variable or two variables separated by either a + or - operator. The only variable names in this language are A, B, and C.

A derivation of a program in this language follows:

- <program> => **begin** <stmt_list> **end**
- => **begin** <stmt> ; <stmt_list> **end**
- => **begin** <var> = <expression> ; <stmt_list> **end**

- => **begin** A = <expression> ; <stmt_list> **end**
- => **begin** A = <var> + <var> ; <stmt_list> **end**
- => **begin** A = B + <var> ; <stmt_list> **end**
- => **begin** A = B + C ; <stmt_list> **end**
- => **begin** A = B + ; <stmt> **end**
- => **begin** A = B + C ; <var> = <expression> **end**
- => **begin** A = B + C ; B = <expression> **end**
- => **begin** A = B + C ; B = <var> **end**
- => **begin** A = B + C ; B = C **end**

This derivation, like all derivations, begins with the start symbol, in this case <program>. The symbol => is read “derives.” Each successive string in the sequence is derived from the previous string by replacing one of the nonterminals with one of that nonterminal’s definitions. Each of the strings in the derivation, including <program>, is called a **sentential form**.

In this derivation, the replaced nonterminal is always the leftmost nonterminal in the previous sentential form. Derivations that use this order of replacement are called **leftmost derivations**. The derivation continues until the sentential form contains no nonterminals. That sentential form, consisting of only terminals, or lexemes, is the generated sentence.

In addition to leftmost, a derivation may be rightmost or in an order that is neither leftmost nor rightmost. Derivation order has no effect on the language generated by a grammar.

By choosing alternative RHSs of rules with which to replace nonterminals in the derivation, different sentences in the language can be generated. By exhaustively choosing all combinations of choices, the entire language can be generated. This language, like most others, is infinite, so one cannot generate *all* the sentences in the language in finite time.

Example 3.2 is another example of a grammar for part of a typical - programming language.

EXAMPLE 3.2 A Grammar for Simple Assignment Statements

- <assign> → <id> = <expr>
- <id> → A| B | C
- <expr> → <id> + <expr>
- | <id> * <expr>
- | (<expr>)
- | <id>

The grammar of Example 3.2 describes assignment statements whose right sides are arithmetic expressions with multiplication and addition operators and parentheses. For example, the statement

```
A = B * ( A + C )
```

is generated by the leftmost derivation:

- <assign> => <id> = <expr>
- => A = <expr>
- => A = <id> * <expr>
- => A = B * <expr>
- => A = B * (<expr>)
- => A = B * (<id> + <expr>)

- => A = B * (A + <expr>)
- => A = B * (A + <id>)
- => A = B * (A + C)

3.3.1.6 Parse Trees

One of the most attractive features of grammars is that they naturally describe the hierarchical syntactic structure of the sentences of the languages they define. These hierarchical structures are called **parse trees**. For example, the parse tree in Figure 3.1 shows the structure of the assignment statement derived previously.

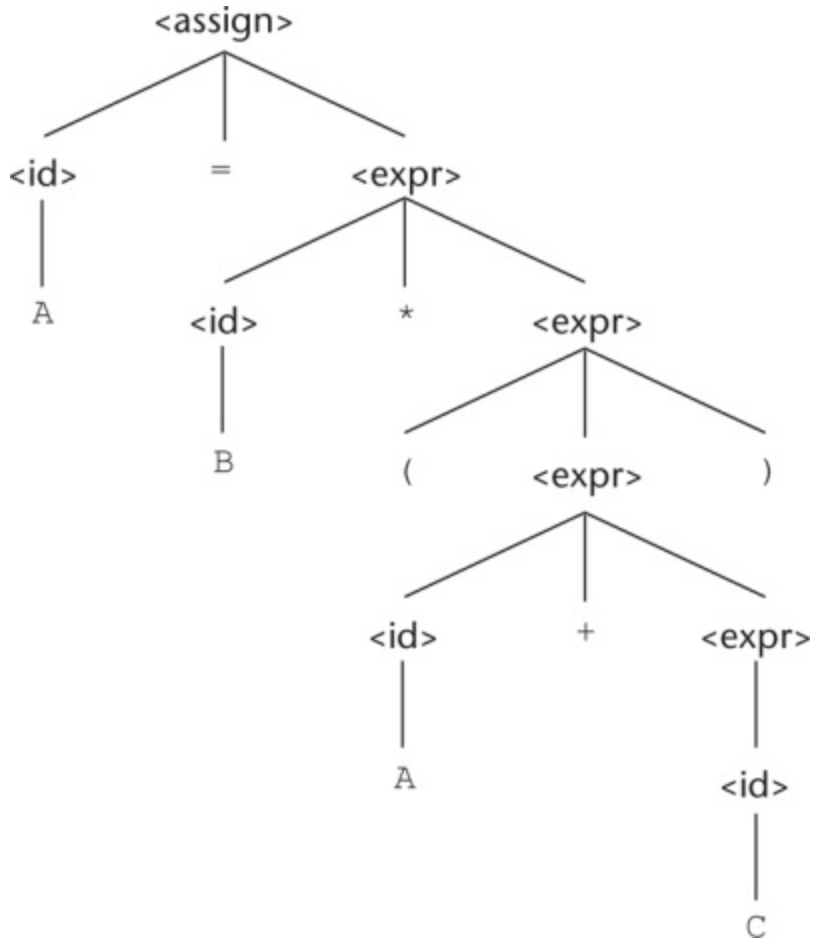


Figure 3.1 A parse tree for the simple statement `A = B * (A + C)`

Figure 3.1 Full Alternative Text

Every internal node of a parse tree is labeled with a nonterminal symbol; every leaf is labeled with a terminal symbol. Every subtree of a parse tree describes one instance of an abstraction in the sentence.

3.3.1.7 Ambiguity

A grammar that generates a sentential form for which there are two or more distinct parse trees is said to be **ambiguous**. Consider the grammar shown in Example 3.3, which is a minor variation of the grammar shown in Example 3.2.

EXAMPLE 3.3 An Ambiguous Grammar for Simple Assignment Statements

- <assign> → <id> = <expr>
- <id> → A | B | C
- <expr> → <expr> + <expr>
- | <expr> * <expr>
- | (<expr>)

- | <id>

The grammar of Example 3.3 is ambiguous because the sentence

```
A = B + C * A
```

has two distinct parse trees, as shown in Figure 3.2. The ambiguity occurs because the grammar specifies slightly less syntactic structure than does the grammar of Example 3.2. Rather than allowing the parse tree of an expression to grow only on the right, this grammar allows growth on both the left and the right.

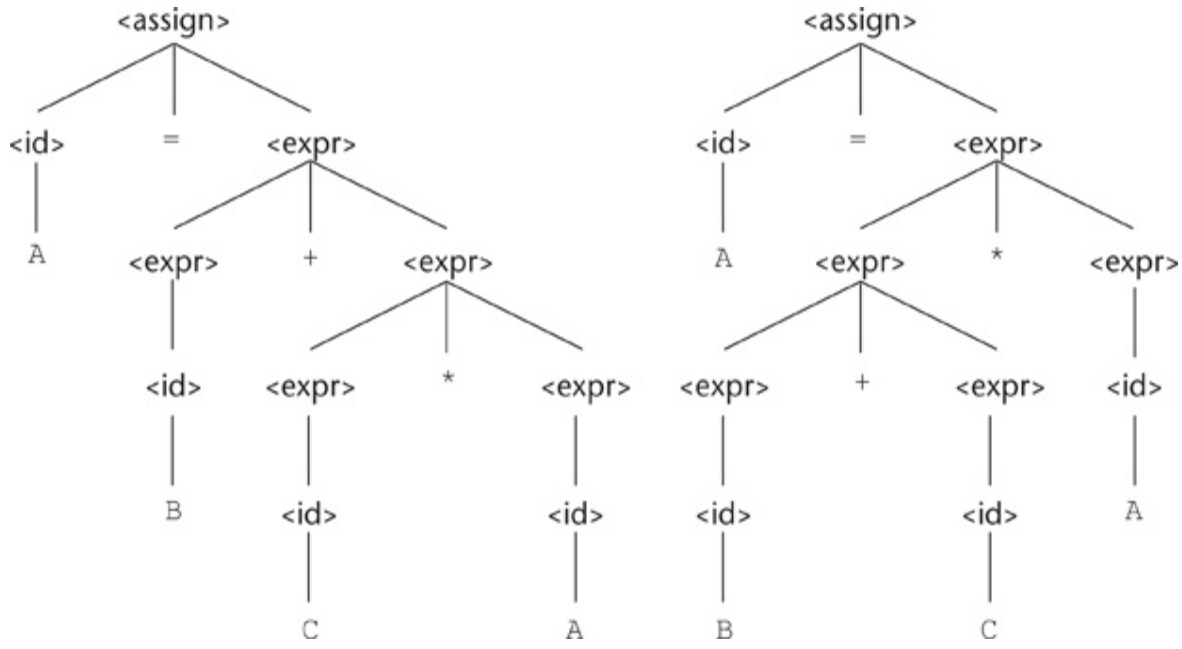


Figure 3.2 Two distinct parse trees for the same sentence, `A = B + C * A`

Figure 3.2 Full Alternative Text

Syntactic ambiguity of language structures is a problem because compilers often base the semantics of those structures on their syntactic form.

Specifically, the compiler chooses the code to be generated for a statement by examining its parse tree. If a language structure has more than one parse tree, then the meaning of the structure cannot be determined uniquely. This problem is discussed in two specific examples in the following subsections.

There are several other characteristics of a grammar that are sometimes useful in determining whether a grammar is ambiguous.1 They include the following: (1) if the grammar generates a sentence with more than one leftmost derivation and (2) if the grammar generates a sentence with more than one rightmost derivation.

1. Note that it is mathematically impossible to determine whether an arbitrary grammar is ambiguous.

Some parsing algorithms can be based on ambiguous grammars. When such a parser encounters an ambiguous construct, it uses nongrammatical information provided by the designer to construct the correct parse tree. In many cases, an ambiguous grammar can be rewritten to be unambiguous but still generate the desired language.

3.3.1.8 Operator Precedence

When an expression includes two different operators, for example, `x + y * z`, one obvious semantic issue is the order of evaluation of the two operators (for example, in this expression is it add and then multiply, or vice versa?). This semantic question can be answered by assigning different precedence levels to operators. For example, if `*` has been assigned higher precedence than `+` (by the language designer), multiplication will be done first, regardless of the order of appearance of the two operators in the expression.

As stated previously, a grammar can describe a certain syntactic structure so that part of the meaning of the structure can be determined from its parse tree. In particular, the fact that an operator in an arithmetic expression is generated lower in the parse tree (and therefore must be evaluated first) can be used to indicate that it has precedence over an operator produced higher up in the tree. In the first parse tree of Figure 3.2, for example, the multiplication

operator is generated lower in the tree, which could indicate that it has precedence over the addition operator in the expression. The second parse tree, however, indicates just the opposite. It appears, therefore, that the two parse trees indicate conflicting precedence information.

Notice that although the grammar of Example 3.2 is not ambiguous, the precedence order of its operators is not the usual one. In this grammar, a parse tree of a sentence with multiple operators, regardless of the particular operators involved, has the rightmost operator in the expression at the lowest point in the parse tree, with the other operators in the tree moving progressively higher as one moves to the left in the expression. For example, in the expression `A + B * C`, `*` is the lowest in the tree, indicating it is to be done first. However, in the expression `A * B + C`, `+` is the lowest, indicating it is to be done first.

A grammar can be written for the simple expressions we have been discussing that is both unambiguous and specifies a consistent precedence of the `+` and `*` operators, regardless of the order in which the operators appear in an expression. The correct ordering is specified by using separate nonterminal symbols to represent the operands of the operators that have different precedence. This requires additional nonterminals and some new rules. Instead of using <expr> for both operands of both `+` and `*`, we could use three nonterminals to represent operands, which allows the grammar to force different operators to different levels in the parse tree. If <expr> is the root symbol for expressions, `+` can be forced to the top of the parse tree by having <expr> directly generate only `+` operators, using the new nonterminal, <term>, as the right operand of `+`. Next, we can define <term> to generate `*` operators, using <term> as the left operand and a new nonterminal, <factor>, as its right operand. Now, `*` will always be lower in the parse tree, simply because it is farther from the start symbol than `+` in every derivation. The grammar of Example 3.4 is such a grammar.

EXAMPLE 3.4 An Unambiguous Grammar for Expressions

- <assign> → <id> = <expr>
- <id> → A | B | C
- <expr> → <expr> + <term>
- | <term>
- <term> → <term> * <factor>
- | <factor>
- <factor> → (<expr>)
- | <id>

The grammar in Example 3.4 generates the same language as the grammars of Examples 3.2 and 3.3, but it is unambiguous and it specifies the usual precedence order of multiplication and addition operators. The following derivation of the sentence A = B + C * A uses the grammar of Example 3.4:

- <assign> => <id> = <expr>
- => A = <expr>
- => A = <expr> + <term>
- => A = <term> + <term>
- => A = <factor> + <term>
- => A = <id> + <term>
- => A = B + <term>
- => A = B + <term> * <factor>
- => A = B + <factor> * <factor>

- => A = B + <id> * <factor>
- => A = B + C * <factor>
- => A = B + C * <id>
- => A = B + C * A

The unique parse tree for this sentence, using the grammar of Example 3.4, is shown in Figure 3.3.

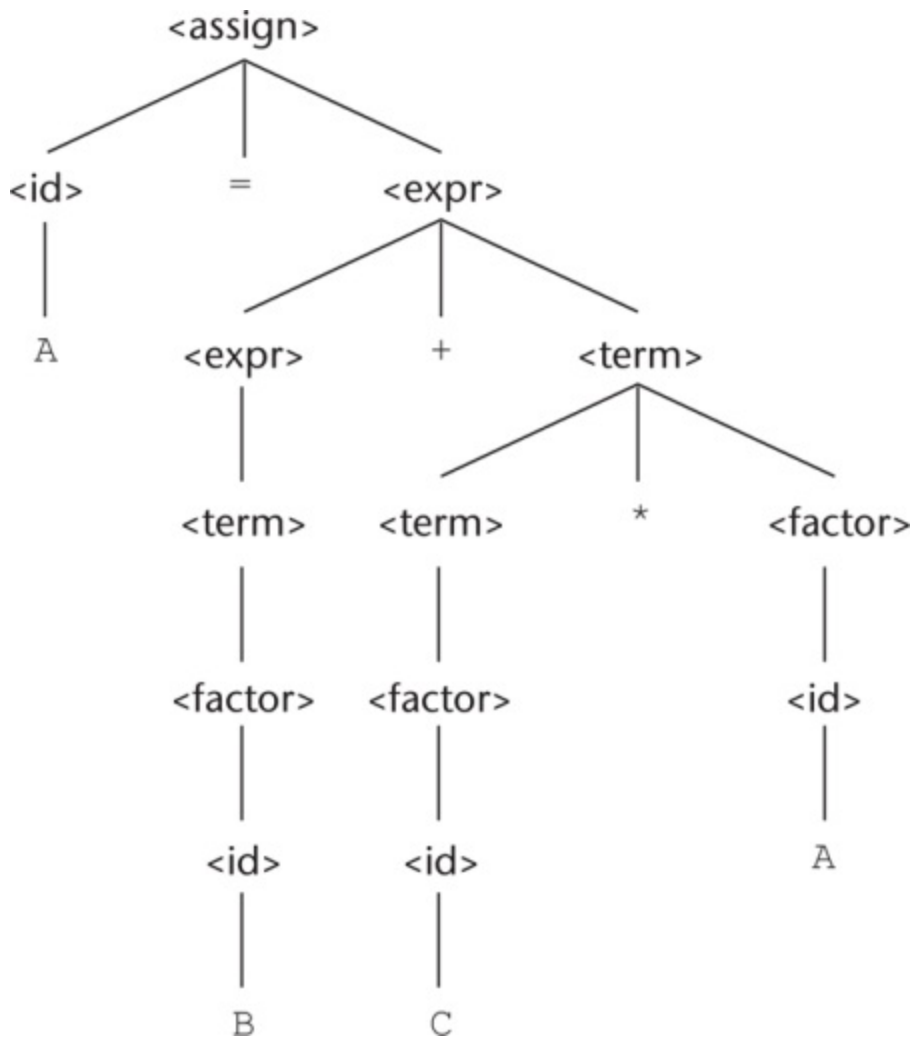


Figure 3.3 The unique parse

tree for `A = B + C * A` using an unambiguous grammar

Figure 3.3 Full Alternative Text

The connection between parse trees and derivations is very close: Either can easily be constructed from the other. Every derivation with an unambiguous grammar has a unique parse tree, although that tree can be represented by different derivations. For example, the following derivation of the sentence `A = B + C * A` is different from the derivation of the same sentence given previously. This is a rightmost derivation, whereas the previous one is leftmost. Both of these derivations, however, are represented by the same parse tree.

- <assign> => <id> = <expr>
- => <id> = <expr> + <term>
- => <id> = <expr> + <term> * <factor>
- => <id> = <expr> + <term> * <id>
- => <id> = <expr> + <term> * A
- => <id> = <expr> + <factor> * A
- => <id> = <expr> + <id> * A
- => <id> = <expr> + C * A
- => <id> = <term> + C * A
- => <id> = <factor> + C * A
- => <id> = <id> + C * A

- => <id> = B + C * A
- => A = B + C * A

3.3.1.9 Associativity of Operators

When an expression includes two operators that have the same precedence (as * and / usually have)—for example, A / B * C—a semantic rule is required to specify which should have precedence.2 This rule is named *associativity*.

2. An expression with two occurrences of the same operator has the same issue; for example, A / B / C.

As was the case with precedence, a grammar for expressions may correctly imply operator associativity. Consider the following example of an assignment statement:

```
A = B + C + A
```

The parse tree for this sentence, as defined with the grammar of Example 3.4, is shown in Figure 3.4.

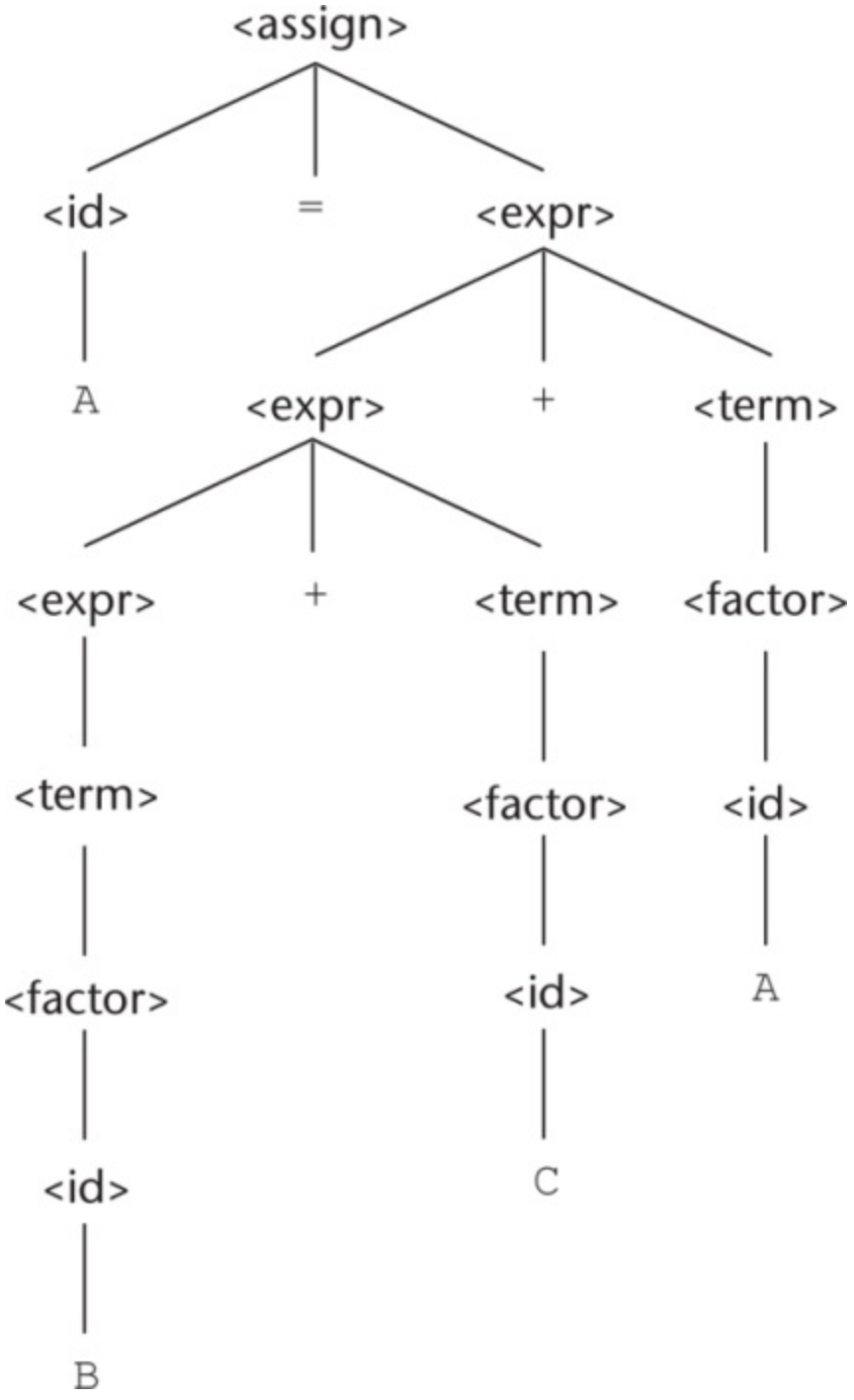


Figure 3.4 A parse tree for `A = B + C + A` illustrating the

associativity of addition

Figure 3.4 Full Alternative Text

The parse tree of Figure 3.4 shows the left addition operator lower than the right addition operator. This is the correct order if addition is meant to be left associative, which is typical. In most cases, the associativity of addition in a computer is irrelevant. In mathematics, addition is associative, which means that left and right associative orders of evaluation mean the same thing. That is, `(A + B) + C = A + (B + C)`. Floating-point addition in a computer, however, is not necessarily associative. For example, suppose floating-point values store seven digits of accuracy. Consider the problem of adding 11 numbers together, where one of the numbers is 107 and the other ten are 1. If the small numbers (the 1's) are each added to the large number, one at a time, there is no effect on that number, because the small numbers occur in the eighth digit of the large number. However, if the small numbers are first added together and the result is added to the large number, the result in seven-digit accuracy is 1.000001 * 107. Subtraction and division are not associative, whether in mathematics or in a computer. Therefore, correct associativity may be essential for an expression that contains either of them.

When a grammar rule has its LHS also appearing at the beginning of its RHS, the rule is said to be **left recursive**. This left recursion specifies left associativity. For example, the left recursion of the rules of the grammar of Example 3.4 causes it to make both addition and multiplication left associative. Unfortunately, left recursion disallows the use of some important syntax analysis algorithms. When one of these algorithms is to be used, the grammar must be modified to remove the left recursion. This, in turn, disallows the grammar from precisely specifying that certain operators are left associative. Fortunately, left associativity can be enforced by the compiler, even though the grammar does not dictate it.

In most languages that provide it, the exponentiation operator is right associative. To indicate right associativity, right recursion can be used. A grammar rule is **right recursive** if the LHS appears at the right end of the RHS. Rules such as

- <factor> → <exp> ** <factor>
- |<exp>
- <exp> → (<expr>)
- |id

could be used to describe exponentiation as a right-associative operator.

3.3.1.10 An Unambiguous Grammar for `if-else`

The BNF rules for a Java **`if-else`** statement are as follows:

- <if_stmt> → **`if`** (<logic_expr>) <stmt>
- **`if`** (<logic_expr>) <stmt> **`else`** <stmt>

If we also have <stmt> → <if_stmt>, this grammar is ambiguous. The simplest sentential form that illustrates this ambiguity is

- **`if`** (<logic_expr>) **`if`** (<logic_expr>) <stmt> **`else`** <stmt>

The two parse trees in Figure 3.5 show the ambiguity of this sentential form. Consider the following example of this construct:

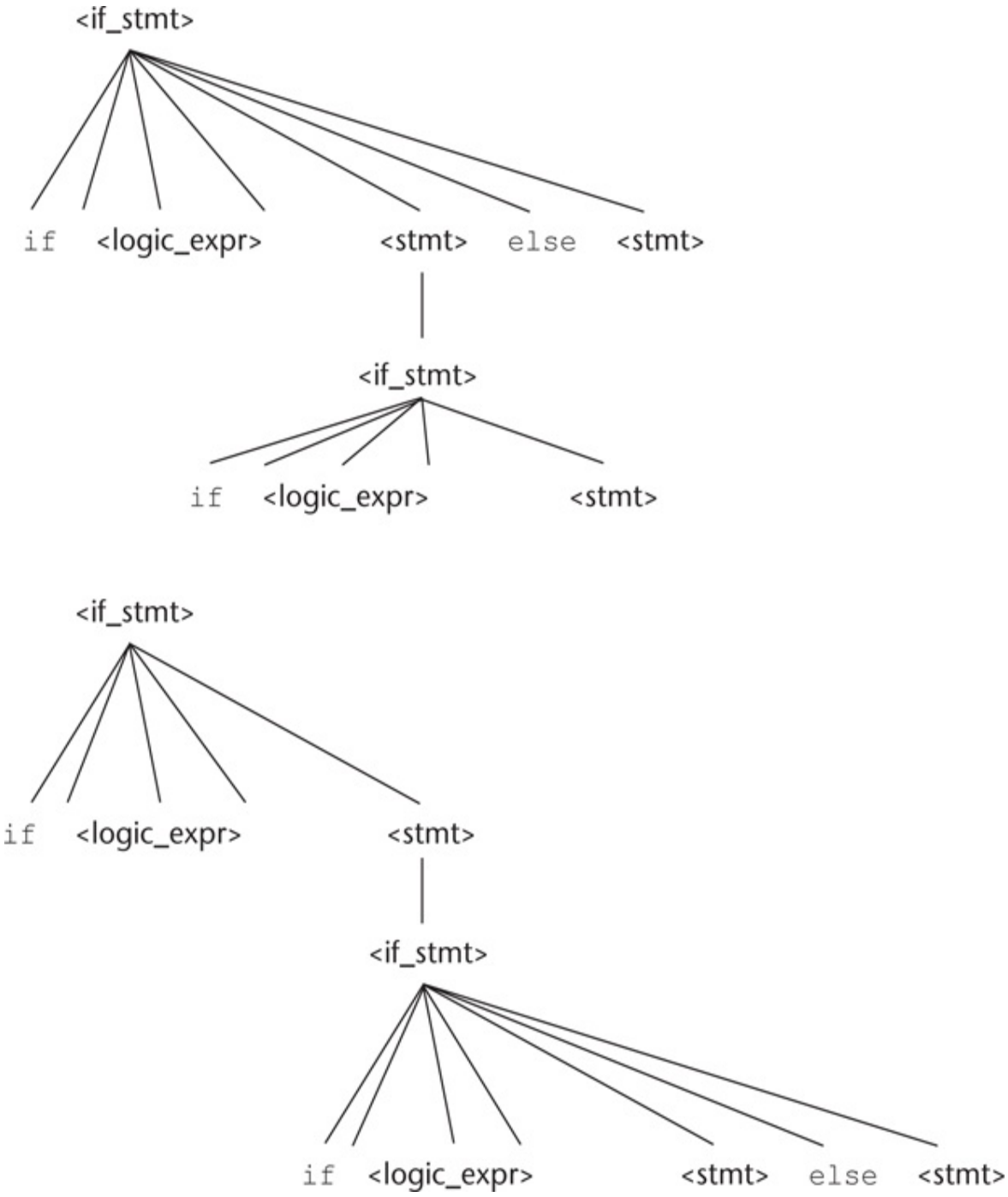


Figure 3.5 Two distinct parse trees for the same sentential form

Figure 3.5 Full Alternative Text

- `if (done == true)`
- `if (denom == 0)`
- `    quotient = 0;`
- `    else quotient = num / denom;`

The problem is that if the upper parse tree in Figure 3.5 is used as the basis for translation, the else clause would be executed when `done` is not true, which probably is not what was intended by the author of the construct. We will examine the practical problems associated with this else-association problem in Chapter 8.

We will now develop an unambiguous grammar that describes this **`if`** statement. The rule for **`if`** constructs in many languages is that an else clause, when present, is matched with the nearest previous unmatched then clause. Therefore, there cannot be an **`if`** statement without an **`else`** between a then clause and its matching **`else`**. So, for this situation, statements must be distinguished between those that are matched and those that are unmatched, where unmatched statements are **`else`**-less **`if`**s and all other statements are matched. The problem with the earlier grammar is that it treats all statements as if they had equal syntactic significance—that is, as if they were all matched.

To reflect the different categories of statements, different abstractions, or nonterminals, must be used. The unambiguous grammar based on these ideas follows:

- <stmt> → <matched> | <unmatched>
- <matched> → **`if`** (<logic_expr>) <matched> **`else`** <matched>
- |any non-if statement
- <unmatched> → **`if`** (<logic_expr>) <stmt>

- |**if** (<logic_expr>) <matched> **else** <unmatched>

There is just one possible parse tree, using this grammar, for the following sentential form:

- **if** (<logic_expr>) **if** (<logic_expr>) <stmt> **else** <stmt>

3.3.2 Extended BNF

Because of a few minor inconveniences in BNF, it has been extended in several ways. Most extended versions are called Extended BNF, or simply EBNF, even though they are not all exactly the same. The extensions do not enhance the descriptive power of BNF; they only increase its readability and writability.

Three extensions are commonly included in the various versions of EBNF. The first of these denotes an optional part of an RHS, which is delimited by brackets. For example, a C **if-else** statement can be described as

- <if_stmt> → **if** (<expression>) <statement> [**else** <statement>]

Without the use of the brackets, the syntactic description of this statement would require the following two rules:

- <if_stmt> → **if** (<expression>) <statement>
- |**if** (<expression>) <statement> **else** <statement>

The second extension is the use of braces in a RHS to indicate that the enclosed part can be repeated indefinitely or left out altogether. This extension allows lists to be built with a single rule, instead of using recursion and two rules. For example, lists of identifiers separated by commas can be described by the following rule:

- <ident_list> → <identifier> {, <identifier>}

This is a replacement of the recursion by a form of implied iteration; the part

enclosed within braces can be iterated any number of times.

The third common extension deals with multiple-choice options. When a single element must be chosen from a group, the options are placed in parentheses and separated by the OR operator, `|`. For example,

- <term> → <term> (`*` | `/` | `%`) <factor>

In BNF, a description of this <term> would require the following three rules:

- <term> → <term> `*` <factor>
- | <term> `/` <factor>
- | <term> `%` <factor>

The brackets, braces, and parentheses in the EBNF extensions are **metasymbols**, which means they are notational tools and not terminal symbols in the syntactic entities they help describe. In cases where these metasymbols are also terminal symbols in the language being described, the instances that are terminal symbols can be underlined or quoted. Example 3.5 illustrates the use of braces and multiple choices in an EBNF grammar.

EXAMPLE 3.5 BNF and EBNF Versions of an Expression Grammar

BNF:

- <expr> → <expr> + <term>
- | <expr> - <term>
- | <term>

- <term> → <term> * <factor>
- | <term> / <factor>
- | <factor>
- <factor> → <exp> ** <factor>
- <exp>
- <exp> → (<expr>)
- | id

EBNF:

- <expr> → <term> {(+ | -) <term>}
- <term> → <factor> {(* | /) <factor>}
- <factor> → <exp> { ** <exp>}
- <exp> → (<expr>)
- | id

The BNF rule

- <expr> → <expr> + <term>

clearly specifies—in fact forces—the + operator to be left associative. However, the EBNF version,

- <expr> → <term> {+ <term>}

does not imply the direction of associativity. This problem is overcome in a syntax analyzer based on an EBNF grammar for expressions by designing the syntax analysis process to enforce the correct associativity. This is discussed further in Chapter 4.

Some versions of EBNF allow a numeric superscript to be attached to the right brace to indicate an upper limit to the number of times the enclosed part can be repeated. Also, some versions use a plus (+) superscript to indicate one or more repetitions. For example,

- <compound> → **`begin`** <stmt> `{`<stmt>`}` **`end`**

and

- <compound> → **`begin`** `{`<stmt>`}`$^{+}$ **`end`**

are equivalent.

In recent years, some variations on BNF and EBNF have appeared. Among these are the following:

- In place of the arrow, a colon is used and the RHS is placed on the next line.
- Instead of a vertical bar to separate alternative RHSs, they are simply placed on separate lines.
- In place of square brackets to indicate something being optional, the subscript opt is used. For example,
- Constructor Declarator → SimpleName (FormalParameterListopt)
- Rather than using the `|` symbol in a parenthesized list of elements to indicate a choice, the words “one of” are used. For example,
    - AssignmentOperator → one of `= *= /= %= += -=`
    - `<<= >>= &= ^= |=`

There is a standard for EBNF, ISO/IEC 14977:1996 (1996), but it is rarely used. The standard uses the equal sign (=) instead of an arrow in rules, terminates each RHS with a semicolon, and requires quotes on all terminal symbols. It also specifies a host of other notational rules.

3.3.3 Grammars and Recognizers

Earlier in this chapter, we suggested that there is a close relationship between generation and recognition devices for a given language. In fact, given a - context-free grammar, a recognizer for the language generated by the grammar can be algorithmically constructed. A number of software systems have been developed that perform this construction. Such systems allow the quick creation of the syntax analysis part of a compiler for a new language and are therefore quite valuable. One of the first of these syntax analyzer generators is named yacc (*y*et *a*nother *c*ompiler *c*ompiler) (Johnson, 1975). There are now many such systems available.

3.4 Attribute Grammars

An **attribute grammar** is a device used to describe more of the structure of a programming language than can be described with a context-free grammar. An attribute grammar is an extension to a context-free grammar. The extension allows certain language rules to be conveniently described, such as type compatibility. Before we formally define the form of attribute grammars, we must clarify the concept of static semantics.

history note

Attribute grammars have been used in a wide variety of applications. They have been used to provide complete descriptions of the syntax and static semantics of programming languages (Watt, 1979); they have been used as the formal definition of a language that can be input to a compiler generation system (Farrow, 1982); and they have been used as the basis of several syntax-directed editing systems (Teitelbaum and Reps, 1981; Fischer et al., 1984). In addition, attribute grammars have been used in natural-language processing systems (Correa, 1992).

3.4.1 Static Semantics

There are some characteristics of programming languages that are difficult to describe with BNF, and some that are impossible. As an example of a syntax rule that is difficult to specify with BNF, consider type compatibility rules. In Java, for example, a floating-point value cannot be assigned to an integer type variable, although the opposite is legal. Although this restriction can be specified in BNF, it requires additional nonterminal symbols and rules. If all of the typing rules of Java were specified in BNF, the grammar would become too large to be useful, because the size of the grammar determines the size of the syntax analyzer.

As an example of a syntax rule that cannot be specified in BNF, consider the common rule that all variables must be declared before they are referenced. It has been proven that this rule cannot be specified in BNF.

These problems exemplify the categories of language rules called static semantics rules. The **static semantics** of a language is only indirectly related to the meaning of programs during execution; rather, it has to do with the legal forms of programs (syntax rather than semantics). Many static semantic rules of a language state its type constraints. Static semantics is so named because the analysis required to check these specifications can be done at compile time.

Because of the problems of describing static semantics with BNF, a variety of more powerful mechanisms has been devised for that task. One such mechanism, attribute grammars, was designed by Knuth (1968) to describe both the syntax and the static semantics of programs.

Attribute grammars are a formal approach both to describing and checking the correctness of the static semantics rules of a program. Although they are not always used in a formal way in compiler design, the basic concepts of attribute grammars are at least informally used in every compiler (see Aho et al., 1988).

Dynamic semantics, which is the meaning of expressions, statements, and program units, is discussed in Section 3.5.

3.4.2 Basic Concepts

Attribute grammars are context-free grammars to which have been added attributes, attribute computation functions, and predicate functions. **Attributes**, which are associated with grammar symbols (the terminal and nonterminal symbols), are similar to variables in the sense that they can have values assigned to them. **Attribute computation functions**, sometimes called semantic functions, are associated with grammar rules. They are used to specify how attribute values are computed. **Predicate functions**, which state the static semantic rules of the language, are associated with grammar

rules.

These concepts will become clearer after we formally define attribute grammars and provide an example.

3.4.3 Attribute Grammars Defined

An attribute grammar is a grammar with the following additional features:

- Associated with each grammar symbol X is a set of attributes A(X). The set A(X) consists of two disjoint sets S(X) and I(X), called synthesized and inherited attributes, respectively. **Synthesized attributes** are used to pass semantic information up a parse tree, while **inherited attributes** pass semantic information down and across a tree.

- Associated with each grammar rule is a set of semantic functions and a possibly empty set of predicate functions over the attributes of the symbols in the grammar rule. For a rule the $X0 \rightarrow X1 \ldots Xn$, synthesized attributes of X0 are computed with semantic functions of the form S(X0)=f(A(X1), …, A(Xn)). So the value of a synthesized attribute on a parse tree node depends only on the values of the attributes on that node's children nodes. Inherited attributes of symbols Xj, $1 \leq j \leq n$ (in the rule above), are computed with a semantic function of the form I(Xj=f(A(X0), …, A(Xn)). So the value of an inherited attribute on a parse tree node depends on the attribute values of that node's parent node and those of its sibling nodes. Note that, to avoid circularity, inherited attributes are often restricted to functions of the form I(Xj)=f(A(X0), …, A(X(_{j−1}))). This form prevents an inherited attribute from depending on itself or on attributes to the right in the parse tree.

- A predicate function has the form of a Boolean expression on the union of the attribute set {A(X0), …, A(Xn)} and a set of literal attribute values. The only derivations allowed with an attribute grammar are those in which every predicate associated with every nonterminal is true. A false predicate function value indicates a violation of the syntax or

static semantics rules of the language.

A parse tree of an attribute grammar is the parse tree based on its underlying BNF grammar, with a possibly empty set of attribute values attached to each node. If all the attribute values in a parse tree have been computed, the tree is said to be **fully attributed**. Although in practice it is not always done this way, it is convenient to think of attribute values as being computed after the complete unattributed parse tree has been constructed by the compiler.

3.4.4 Intrinsic Attributes

Intrinsic attributes are synthesized attributes of leaf nodes whose values are determined outside the parse tree. For example, the type of an instance of a variable in a program could come from the symbol table, which is used to store variable names and their types. The contents of the symbol table are set based on earlier declaration statements. Initially, assuming that an unattributed parse tree has been constructed and that attribute values are needed, the only attributes with values are the intrinsic attributes of the leaf nodes. Given the intrinsic attribute values on a parse tree, the semantic functions can be used to compute the remaining attribute values.

3.4.5 Examples of Attribute Grammars

As a very simple example of how attribute grammars can be used to describe static semantics, consider the following fragment of an attribute grammar that describes the rule that the name on the **`end`** of an Ada procedure must match the procedure's name. (This rule cannot be stated in BNF.) The string attribute of <proc_name>, denoted by <proc_name>.string, is the actual string of characters that were found immediately following the reserved word **`procedure`** by the compiler. Notice that when there is more than one occurrence of a nonterminal in a syntax rule in an attribute grammar, the nonterminals are subscripted with brackets to distinguish them. Neither the

subscripts nor the brackets are part of the described language.

- Syntax rule: <proc_def> → **`procedure`** <proc_name>`[1]`
- <proc_body> **`end`** <proc_name>`[2];`
- Predicate: <proc_name>`[1]`string == <proc_name>`[2]`.string

In this example, the predicate rule states that the name string attribute of the <proc_name> nonterminal in the subprogram header must match the name string attribute of the <proc_name> nonterminal following the end of the subprogram.

Next, we consider a larger example of an attribute grammar. In this case, the example illustrates how an attribute grammar can be used to check the type rules of a simple assignment statement. The syntax and static semantics of this assignment statement are as follows: The only variable names are `A`, `B`, and `C`. The right side of the assignments can be either a variable or an expression in the form of a variable added to another variable. The variables can be one of two types: int or real. When there are two variables on the right side of an assignment, they need not be the same type. The type of the expression when the operand types are not the same is always real. When they are the same, the expression type is that of the operands. The type of the left side of the assignment must match the type of the right side. So the types of operands in the right side can be mixed, but the assignment is valid only if the target and the value resulting from evaluating the right side have the same type. The attribute grammar specifies these static semantic rules.

The syntax portion of our example attribute grammar is

- <assign> → <var> = <expr>
- <expr> → <var> + <var>
- | <var>
- <var> → `A | B | C`

The attributes for the nonterminals in the example attribute grammar are

described in the following paragraphs:

- actual_type—A synthesized attribute associated with the nonterminals <var> and <expr>. It is used to store the actual type, int or real, of a variable or expression. In the case of a variable, the actual type is intrinsic. In the case of an expression, it is determined from the actual types of the child node or children nodes of the <expr> nonterminal.

- expected_type—An inherited attribute associated with the nonterminal <expr>. It is used to store the type, either int or real, that is expected for the expression, as determined by the type of the variable on the left side of the assignment statement.

  The complete attribute grammar follows in Example 3.6.

EXAMPLE 3.6 An Attribute Grammar for Simple Assignment Statements

1. Syntax rule: <assign> → <var> = <expr>

   Semantic rule: <expr>.expected_type ← <var>.actual_type

2. Syntax rule: <expr> → <var>[2] + <var>[3]

   Semantic rule: <expr>.actual_type ←

   if (<var>[2].actual_type = int) and

   (<var>[3].actual_type = int)

   then int

   else real

end if

Predicate: <expr>.actual_type == <expr>.expected_type

3. Syntax rule: <expr> → <var>

   Semantic rule: <expr>.actual_type ← <var>.actual_type

   Predicate: <expr>.actual_type == <expr>.expected_type

4. Syntax rule: <var> → `A | B | C`

   Semantic rule: <var>.actual_type ← look-up(<var>.string)

The look-up function looks up a given variable name in the symbol table and returns the variable's type.

A parse tree of the sentence `A = A + B` generated by the grammar in Example 3.6 is shown in Figure 3.6. As in the grammar, bracketed numbers are added after the repeated node labels in the tree so they can be referenced unambiguously.

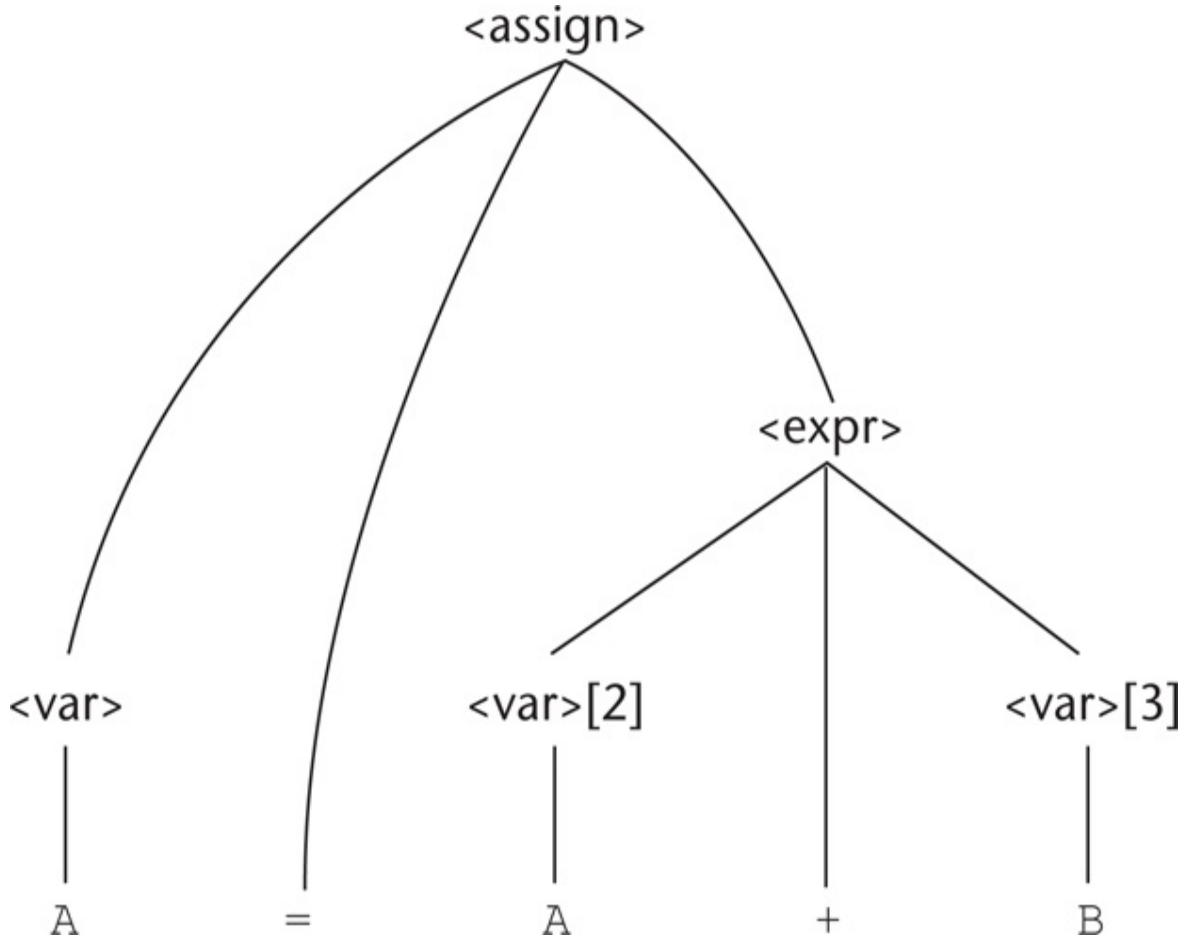


Figure 3.6 A parse tree for `A = A + B`

Figure 3.6 Full Alternative Text

3.4.6 Computing Attribute Values

Now, consider the process of computing the attribute values of a parse tree, which is sometimes called **decorating** the parse tree. If all attributes were inherited, this could proceed in a completely top-down order, from the root to the leaves. Alternatively, it could proceed in a completely bottom-up order, from the leaves to the root, if all the attributes were synthesized. Because our

grammar has both synthesized and inherited attributes, the evaluation process cannot be in any single direction. The following is an evaluation of the attributes, in an order in which it is possible to compute them:

1. <var>.actual_type ← look-up(A) (Rule 4)
2. <expr>.expected_type ← <var>.actual_type (Rule 1)
3. <var>[2].actual_type ← look-up(A) (Rule 4)

   <var>[3].actual_type ← look-up(B) (Rule 4)
4. <expr>.actual_type ← either int or real (Rule 2)
5. <expr>.expected_type == <expr>.actual_type is either

   TRUE or FALSE (Rule 2)

The tree in Figure 3.7 shows the flow of attribute values in the example of Figure 3.6. Solid lines show the parse tree; dashed lines show attribute flow in the tree.

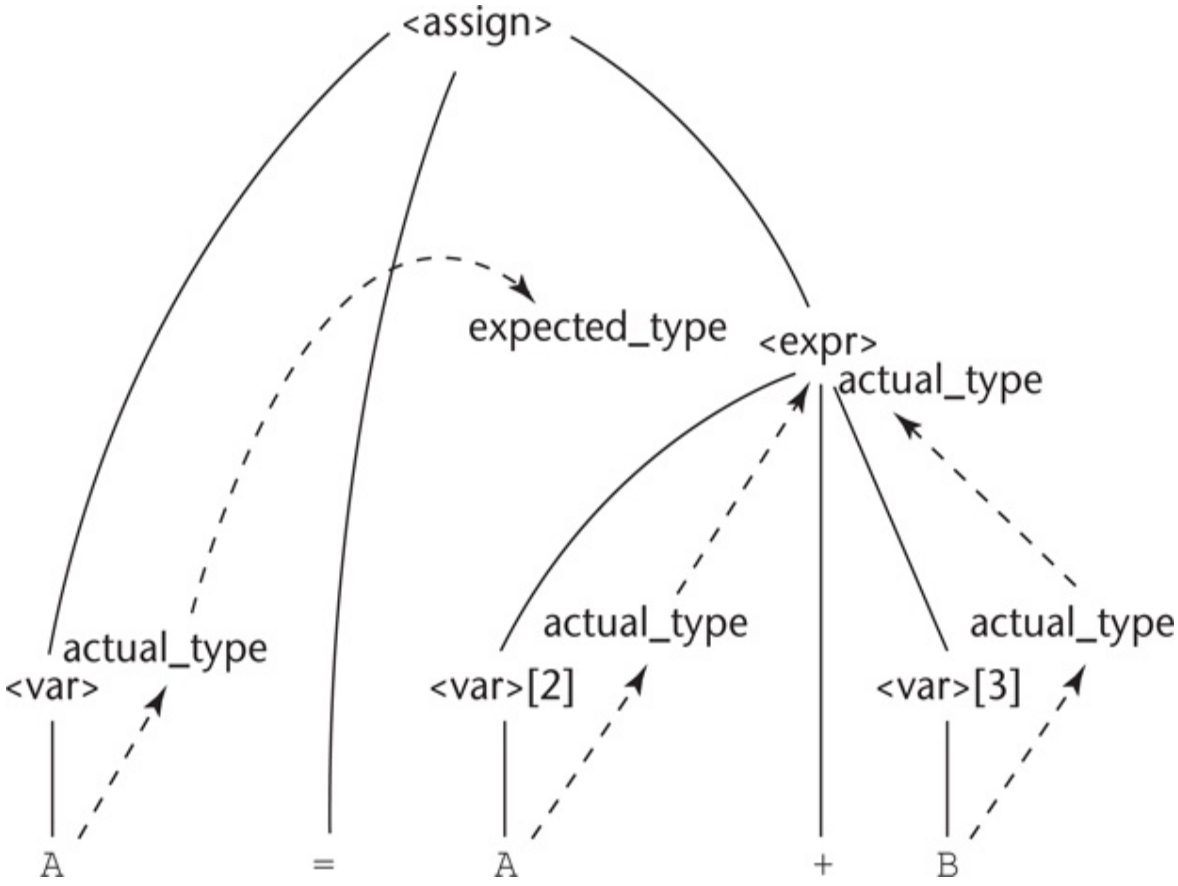


Figure 3.7 The flow of attributes in the tree

Figure 3.7 Full Alternative Text

The tree in Figure 3.8 shows the final attribute values on the nodes. In this example, A is defined as a real and B is defined as an int.

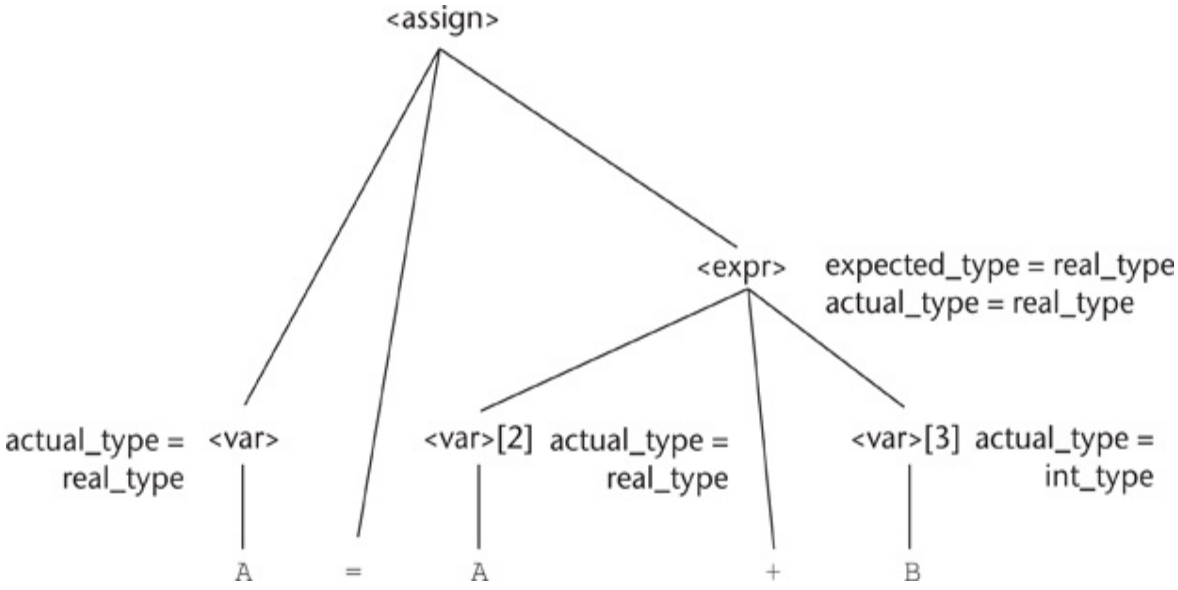


Figure 3.8 A fully attributed parse tree

Figure 3.8 Full Alternative Text

Determining attribute evaluation order for the general case of an attribute grammar is a complex problem, requiring the construction of a dependency graph to show all attribute dependencies.

3.4.7 Evaluation

Checking the static semantic rules of a language is an essential part of all compilers. Even if a compiler writer has never heard of an attribute grammar, he or she would need to use the fundamental ideas of attribute grammars to design the checks of static semantics rules for his or her compiler.

One of the main difficulties in using an attribute grammar to describe all of the syntax and static semantics of a real contemporary programming - language is the size and complexity of the attribute grammar. The large number of attributes and semantic rules required for a complete programming

language make such grammars difficult to write and read. Furthermore, the attribute values on a large parse tree are costly to evaluate. On the other hand, less formal attribute grammars are a powerful and commonly used tool for compiler writers, who are more interested in the process of producing a compiler than they are in formalism.

3.5 Describing the Meanings of Programs: Dynamic Semantics

We now turn to the difficult task of describing the **dynamic semantics**, or meaning, of the expressions, statements, and program units of a programming language. Because of the power and naturalness of the available notation, describing syntax is a relatively simple matter. On the other hand, no universally accepted notation or approach has been devised for dynamic semantics. In this section, we briefly describe several of the methods that have been developed. For the remainder of this section, when we use the term *semantics*, we mean dynamic semantics.

There are several different reasons underlying the need for a methodology and notation for describing semantics. Programmers obviously need to know precisely what the statements of a language do before they can use them effectively in their programs. Compiler writers must know exactly what language constructs mean to design implementations for them correctly. If there were a precise semantics specification of a programming language, programs written in the language potentially could be proven correct without testing. Also, compilers could be shown to produce programs that exhibited exactly the behavior given in the language definition; that is, their correctness could be verified. A complete specification of the syntax and semantics of a programming language could be used by a tool to generate a compiler for the language automatically. Finally, language designers, who would develop the semantic descriptions of their languages, could in the process discover ambiguities and inconsistencies in their designs.

Software developers and compiler designers typically determine the semantics of programming languages by reading English explanations in language manuals. Because such explanations are often imprecise and incomplete, this approach is clearly unsatisfactory. Due to the lack of complete semantics specifications of programming languages, programs are rarely proven correct without testing, and commercial compilers are never generated automatically from language descriptions.

Scheme, a functional language described in Chapter 15, is one of only a few programming languages whose definition includes a formal semantics description. However, the method used is not one described in this chapter, as this chapter is focused on approaches that are suitable for imperative languages.

3.5.1 Operational Semantics

The idea behind **operational semantics** is to describe the meaning of a statement or program by specifying the effects of running it on a machine. The effects on the machine are viewed as the sequence of changes in its state, where the machine's state is the collection of the values in its storage. An obvious operational semantics description, then, is given by executing a compiled version of the program on a computer. Most programmers have, on at least one occasion, written a small test program to determine the meaning of some programming language construct, often while learning the language. Essentially, what such a programmer is doing is using operational semantics to determine the meaning of the construct.

There are several problems with using this approach for complete formal semantics descriptions. First, the individual steps in the execution of machine language and the resulting changes to the state of the machine are too small and too numerous. Second, the storage of a real computer is too large and complex. There are usually several levels of memory devices, as well as connections to enumerable other computers and memory devices through networks. Therefore, machine languages and real computers are not used for formal operational semantics. Rather, intermediate-level languages and interpreters for idealized computers are designed specifically for the process.

There are different levels of uses of operational semantics. At the highest level, the interest is in the final result of the execution of a complete program. This is sometimes called **natural operational semantics**. At the lowest level, operational semantics can be used to determine the precise meaning of a program through an examination of the complete sequence of state changes that occur when the program is executed. This use is sometimes called **structural operational semantics**.

3.5.1.1 The Basic Process

The first step in creating an operational semantics description of a language is to design an appropriate intermediate language, where the primary desired characteristic of the language is clarity. Every construct of the intermediate language must have an obvious and unambiguous meaning. This language is at the intermediate level, because machine language is too low-level to be easily understood and another high-level language is obviously not suitable. If the semantics description is to be used for natural operational semantics, a virtual machine (an interpreter) must be constructed for the intermediate language. The virtual machine can be used to execute either single statements, code segments, or whole programs. The semantics description can be used without a virtual machine if the meaning of a single statement is all that is required. In this use, which is structural operational semantics, the intermediate code can be visually inspected.

The basic process of operational semantics is not unusual. In fact, the concept is frequently used in programming textbooks and programming language reference manuals. For example, the semantics of the C `for` construct can be described in terms of simpler statements, as in

C Statement	*Meaning*

`for (expr1; expr2; expr3) {`
`  ...`
`}`
	`expr1;`
	`loop: if expr2 == 0 goto out`
	`...`
	`expr3;`
	`goto loop`
	`out: ...`

The human reader of such a description is the virtual computer and is assumed to be able to “execute” the instructions in the definition correctly and recognize the effects of the “execution.”

The intermediate language and its associated virtual machine used for formal operational semantics descriptions are often highly abstract. The intermediate

language is meant to be convenient for the virtual machine, rather than for human readers. For our purposes, however, a more human-oriented intermediate language could be used. As such an example, consider the following list of statements, which would be adequate for describing the semantics of the simple control statements of a typical programming language:

- ident = var
- ident = ident + 1
- ident = ident – 1
- **`goto`** label
- **`if`** var relop var **`goto`** label

In these statements, relop is one of the relational operators from the set `{=, <>, >, <, <=, <=}`, ident is an identifier, and var is either an identifier or a constant. These statements are all simple and therefore easy to understand and implement.

A slight generalization of these three assignment statements allows more general arithmetic expressions and assignment statements to be described. The new statements are

- ident = var bin_op var
- ident = un_op var

where bin_op is a binary arithmetic operator and un_op is a unary operator. Multiple arithmetic data types and automatic type conversions, of course, complicate this generalization. Adding just a few more relatively simple instructions would allow the semantics of arrays, records, pointers, and subprograms to be described.

In Chapter 8, the semantics of various control statements are described using this intermediate language.

3.5.1.2 Evaluation

The first and most significant use of formal operational semantics was to describe the semantics of PL/I (Wegner, 1972). That particular abstract machine and the translation rules for PL/I were together named the Vienna Definition Language (VDL), after the city where IBM designed it.

Operational semantics provides an effective means of describing semantics for language users and language implementors, as long as the descriptions are kept simple and informal. The VDL description of PL/I, unfortunately, is so complex that it serves no practical purpose.

Operational semantics depends on programming languages of lower levels, not mathematics. The statements of one programming language are described in terms of the statements of a lower-level programming language. This approach can lead to circularities, in which concepts are indirectly defined in terms of themselves. The methods described in the following two sections are much more formal, in the sense that they are based on mathematics and logic, not programming languages.

3.5.2 Denotational Semantics

Denotational semantics is the most rigorous and most widely known formal method for describing the meaning of programs. It is solidly based on recursive function theory. A thorough discussion of the use of denotational semantics to describe the semantics of programming languages is necessarily long and complex. It is our intent to provide the reader with an introduction to the central concepts of denotational semantics, along with a few simple examples that are relevant to programming language specifications.

The process of constructing a denotational semantics specification for a programming language requires one to define for each language entity both a mathematical object and a function that maps instances of that language entity onto instances of the mathematical object. Because the objects are rigorously defined, they model the exact meaning of their corresponding

entities. The idea is based on the fact that there are rigorous ways of manipulating mathematical objects but not programming language constructs. The difficulty with this method lies in creating the objects and the mapping functions. The method is named *denotational* because the mathematical objects denote the meaning of their corresponding syntactic entities.

The mapping functions of a denotational semantics programming language specification, like all functions in mathematics, have a domain and a range. The domain is the collection of values that are legitimate parameters to the function; the range is the collection of objects to which the parameters are mapped. In denotational semantics, the domain is called the **syntactic domain**, because it is syntactic structures that are mapped. The range is called the **semantic domain**.

Denotational semantics is related to operational semantics. In operational semantics, programming language constructs are translated into simpler programming language constructs, which become the basis of the meaning of the construct. In denotational semantics, programming language constructs are mapped to mathematical objects, either sets or, more often, functions. However, unlike operational semantics, denotational semantics does not model the step-by-step computational processing of programs.

3.5.2.1 Two Simple Examples

We use a very simple language construct, character string representations of binary numbers, to introduce the denotational method. The syntax of such binary numbers can be described by the following grammar rules:

- <bin_num> → `'0'`
- | `'1'`
- | <bin_num> `'0'`
- | <bin_num> `'1'`

A parse tree for the example binary number, `110`, is shown in Figure 3.9.

Notice that we put apostrophes around the syntactic digits to show they are not mathematical digits. This is similar to the relationship between ASCII coded digits and mathematical digits. When a program reads a number as a string, it must be converted to a mathematical number before it can be used as a value in the program.

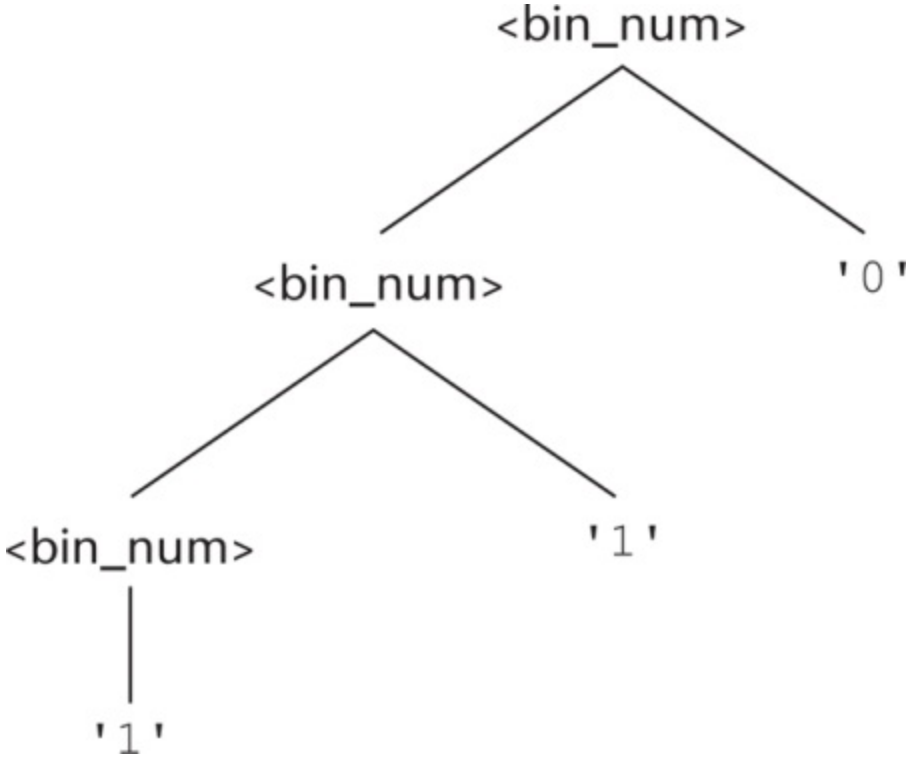


Figure 3.9 A parse tree of the binary number 110

Figure 3.9 Full Alternative Text

The syntactic domain of the mapping function for binary numbers is the set of all character string representations of binary numbers. The semantic domain is the set of nonnegative decimal numbers, symbolized by N.

To describe the meaning of binary numbers using denotational semantics, we associate the actual meaning (a decimal number) with each rule that has a single terminal symbol as its RHS.

In our example, decimal numbers must be associated with the first two grammar rules. The other two grammar rules are, in a sense, computational rules, because they combine a terminal symbol, to which an object can be associated, with a nonterminal, which can be expected to represent some construct. Presuming an evaluation that progresses upward in the parse tree, the nonterminal in the right side would already have its meaning attached. So, a syntax rule with a nonterminal as its RHS would require a function that computed the meaning of the LHS, which represents the meaning of the complete RHS.

The semantic function, named Mbin, maps the syntactic objects, as described in the previous grammar rules, to the objects in N, the set of non-negative decimal numbers. The function Mbin is defined as follows:

- Mbin `('0') = 0`
- Mbin `('1') = 1`
- Mbin `(<bin_num> '0') = 2 *` Mbin `(<bin_num>)`
- Mbin `(<bin_num>'1') = 2 *` Mbin `(<bin_num>) + 1`

The meanings, or denoted objects (which in this case are decimal numbers), can be attached to the nodes of the parse tree shown on the previous page, yielding the tree in Figure 3.10. This is syntax-directed semantics. Syntactic entities are mapped to mathematical objects with concrete meaning.

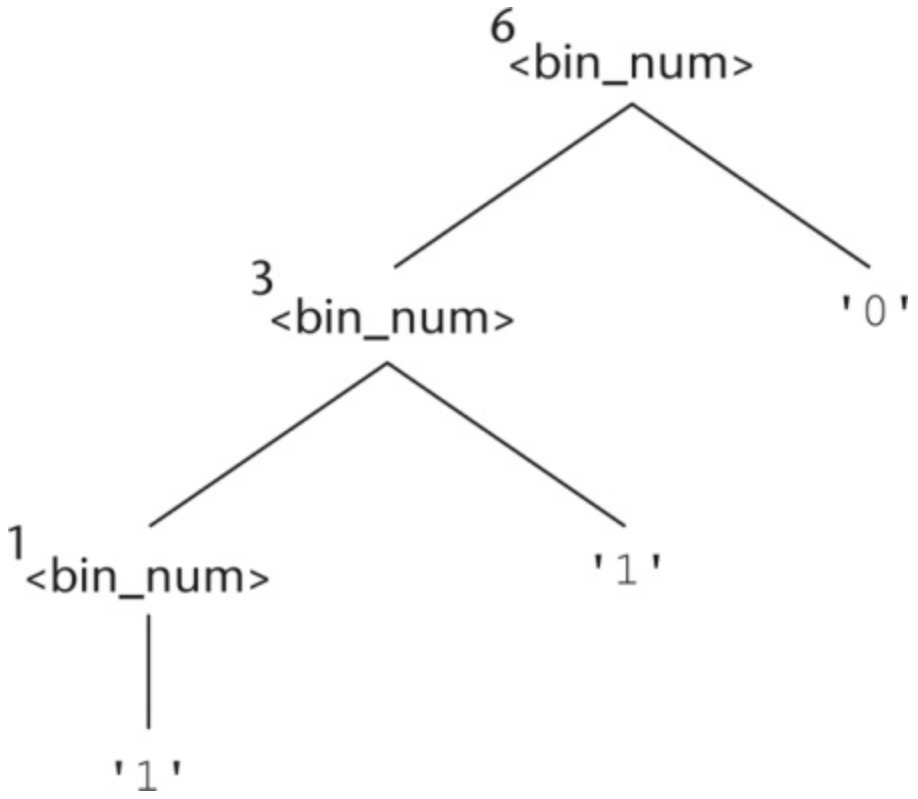


Figure 3.10 A parse tree with denoted objects for 110

Figure 3.10 Full Alternative Text

In part because we need it later, we now show a similar example for describing the meaning of syntactic decimal literals. In this case, the syntactic domain is the set of character string representations of decimal numbers. The semantic domain is once again the set N.

- <dec_num> → '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7''8'|'9'
- |<dec_num> ('0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9')

The denotational mappings for these syntax rules are

- Mdec ('0') = 0, Mdec ('1') = 1, Mdec ('2') = 2, . . ., Mdec ('9') = 9

- Mdec (<dec_num> '0') = 10 * Mdec (<dec_num>)
- Mdec (<dec_num> '1') = 10 * Mdec (<dec_num>) +1
- . . .
- Mdec (<dec_num> '9') = 10 * Mdec (<dec_num>) +9

In the following sections, we present the denotational semantics descriptions of a few simple constructs. The most important simplifying assumption made here is that both the syntax and static semantics of the constructs are correct. In addition, we assume that only two scalar types are included: integer and Boolean.

3.5.2.2 The State of a Program

The denotational semantics of a program could be defined in terms of state changes in an ideal computer. Operational semantics are defined in this way, and denotational semantics are defined in nearly the same way. In a further simplification, however, denotational semantics is defined in terms of only the values of all of the program's variables. So, denotational semantics uses the state of the program to describe meaning, whereas operational semantics uses the state of a machine. The key difference between operational semantics and denotational semantics is that state changes in operational semantics are defined by coded algorithms, written in some programming language, whereas in denotational semantics, state changes are defined by mathematical functions.

Let the state s of a program be represented as a set of ordered pairs, as follows:

- s = { <i1, v1>, <i2, v2>, …, <in, vn>}

Each i is the name of a variable, and the associated v's are the current values of those variables. Any of the v's can have the special value **undef**, which indicates that its associated variable is currently undefined. Let VARMAP be a function of two parameters: a variable name and the program state. The

value of VARMAP (ij, s) is vj (the value paired with ij in state s). Most semantics mapping functions for programs and program constructs map states to states. These state changes are used to define the meanings of programs and program constructs. Some language constructs—for example, expressions—are mapped to values, not states.

3.5.2.3 Expressions

Expressions are fundamental to most programming languages. We assume here that expressions have no side effects. Furthermore, we deal with only very simple expressions: The only operators are + and *, and an expression can have at most one operator; the only operands are scalar integer variables and integer literals; there are no parentheses; and the value of an expression is an integer. Following is the BNF description of these expressions:

- <expr> → <dec_num> | <var> | <binary_expr>
- <binary_expr> → <left_expr> <operator> <right_expr>
- <left_expr> → <dec_num> | <var>
- <right_expr> → <dec_num> | <var>
- <operator> → + | *

The only error we consider in expressions is a variable having an undefined value. Obviously, other errors can occur, but most of them are machine--dependent. Let Z be the set of integers, and let **`error`** be the error value. Then $Z \cup$ {error} is the semantic domain for the denotational specification for our expressions.

The mapping function for a given expression E and state s follows. To distinguish between mathematical function definitions and the assignment statements of programming languages, we use the symbol Δ = to define mathematical functions. The implication symbol, =>, used in this definition connects the form of an operand with its associated case (or switch) construct. Dot notation is used to refer to the child nodes of a node. For

example, <binary_expr>.<left_expr> refers to the left child node of <binary_expr>.

- Me(<expr>, s) Δ= case <expr> of
- <dec_num>=> Mdec (<dec_num>, s)
- <var> =>if VARMAP (<var>, s) == **undef**
- then **error**
- else VARMAP(<var>, s)
- <binary_expr> =>
- if(Me (<binary_expr>.<left_expr>,s) == **undef** OR
- Me (<binary_expr>.<right_expr>, s) == **undef**)
- then **error**
- else if (<binary_expr>.<operator> == '+')
- then Me (<binary_expr>.<left_expr>, s) +
- Me (<binary_expr>.<right_expr>, s)
- else Me (<binary_expr>.<left_expr>, s) *
- Me (<binary_expr>.<right_expr>, s)

3.5.2.4 Assignment Statements

An assignment statement is an expression evaluation plus the setting of the target variable to the expression's value. In this case, the meaning function maps a state to a state. This function can be described with the following:

- Ma (x = E, s) Δ= if Me (E, s) == **error**
- then **error**
- else s' = {< i1, v1′ >, < i2, v2′>, . . . , < in, vn′>}, where
- for j = 1, 2, . . . , n
- if ij == x
- then vj′ = Me (E, s)
- else vj′ = VARMAP(ij,, s)

Note that the comparison in the third last line above, ij == x, is of names, not values.

3.5.2.5 Logical Pretest Loops

The denotational semantics of a logical pretest loop is deceptively simple. To expedite the discussion, we assume that there are two other existing mapping functions, Msl and Mb, that map statement lists and states to states and Boolean expressions to Boolean values (or **error**), respectively. The function is

- M1 (**while** B **do** L, s) Δ= if Mb (B, s) == **undef**
- then **error**
- else if Mb (B, s) == false
- then s

- else if Msl (L, s) == **error**
- then **error**
- else M1 (**while** B **do** L, Msl (L, s))

The meaning of the loop is simply the value of the program variables after the statements in the loop have been executed the prescribed number of times, assuming there have been no errors. In essence, the loop has been converted from iteration to recursion, where the recursion control is mathematically defined by other recursive state mapping functions. Recursion is easier to describe with mathematical rigor than iteration.

One significant observation at this point is that this definition, like actual program loops, may compute nothing because of nontermination.

3.5.2.6 Evaluation

Objects and functions, such as those used in the earlier constructs, can be defined for the other syntactic entities of programming languages. When a complete system has been defined for a given language, it can be used to determine the meaning of complete programs in that language. This provides a framework for thinking about programming in a highly rigorous way.

history note

A significant amount of work has been done on the possibility of using denotational language descriptions to generate compilers automatically (Jones, 1980; Milos et al., 1984; Bodwin et al., 1982). These efforts have shown that the method is feasible, but the work has never progressed to the point where it can be used to generate useful compilers.

As stated previously, denotational semantics can be used as an aid to language design. For example, statements for which the denotational

semantic description is complex and difficult may indicate to the designer that such statements may also be difficult for language users to understand and that an alternative design may be in order.

Because of the complexity of denotational descriptions, they are of little use to language users. On the other hand, they provide an excellent way to describe a language concisely.

Although the use of denotational semantics is normally attributed to Scott and Strachey (1971), the general denotational approach to language description can be traced to the nineteenth century (Frege, 1892).

3.5.3 Axiomatic Semantics

Axiomatic semantics, thus named because it is based on mathematical logic, is the most abstract approach to semantics specification discussed in this chapter. Rather than directly specifying the meaning of a program, axiomatic semantics specifies what can be proven about the program. Recall that one of the possible uses of semantic specifications is to prove the correctness of programs.

In axiomatic semantics, there is no model of the state of a machine or program or model of state changes that take place when the program is executed. The meaning of a program is based on relationships among program variables and constants, which are the same for every execution of the program.

Axiomatic semantics has two distinct applications: program verification and program semantics specification. This section focuses on program verification in its description of axiomatic semantics.

Axiomatic semantics was defined in conjunction with the development of an approach to proving the correctness of programs. Such correctness proofs, when they can be constructed, show that a program performs the computation described by its specification. In a proof, each statement of a program is both preceded and followed by a logical expression that specifies constraints on

program variables. These, rather than the entire state of an abstract machine (as with operational semantics), are used to specify the meaning of the statement. The notation used to describe constraints—indeed, the language of axiomatic semantics—is predicate calculus. Although simple Boolean expressions are often adequate to express constraints, in some cases they are not.

When axiomatic semantics is used to specify formally the meaning of a statement, the meaning is defined by the statement's effect on assertions about the data affected by the statement.

3.5.3.1 Assertions

The logical expressions used in axiomatic semantics are called predicates, or **assertions**. An assertion immediately preceding a program statement describes the constraints on the program variables at that point in the program. An assertion immediately following a statement describes the new constraints on those variables (and possibly others) after execution of the statement. These assertions are called the **precondition** and **postcondition**, respectively, of the statement. For two adjacent statements, the postcondition of the first serves as the precondition of the second. Developing an axiomatic description or proof of a given program requires that every statement in the program has both a precondition and a postcondition.

In the following sections, we examine assertions from the point of view that preconditions for statements are computed from given postconditions, although it is possible to consider these in the opposite sense. We assume all variables are integer type. As a simple example, consider the following assignment statement and postcondition:

```
sum = 2 * x + 1 {sum > 1}
```

Precondition and postcondition assertions are presented in braces to distinguish them from parts of program statements. One possible precondition for this statement is `{x > 10}`.

In axiomatic semantics, the meaning of a specific statement is defined by its

precondition and its postcondition. In effect, the two assertions specify precisely the effect of executing the statement.

In the following subsections, we focus on correctness proofs of statements and programs, which is a common use of axiomatic semantics. The more general concept of axiomatic semantics is to state precisely the meaning of statements and programs in terms of logic expressions. Program verification is one application of axiomatic descriptions of languages.

3.5.3.2 Weakest Preconditions

The **weakest precondition** is the least restrictive precondition that will guarantee the validity of the associated postcondition. For example, in the statement and postcondition given in Section 3.5.3.1, `{x > 10}`, `{x > 50}`, and `{x > 1000}` are all valid preconditions. The weakest of all preconditions in this case is `{x > 0}`.

If the weakest precondition can be computed from the most general postcondition for each of the statement types of a language, then the processes used to compute these preconditions provide a concise description of the semantics of that language. Furthermore, correctness proofs can be constructed for programs in that language. A program proof is begun by using the characteristics of the results of the program's execution as the postcondition of the last statement of the program. This postcondition, along with the last statement, is used to compute the weakest precondition for the last statement. This precondition is then used as the postcondition for the second last statement. This process continues until the beginning of the program is reached. At that point, the precondition of the first statement states the conditions under which the program will compute the desired results. If these conditions are implied by the input specification of the program, the program has been verified to be correct.

An **inference rule** is a method of inferring the truth of one assertion on the basis of the values of other assertions. The general form of an inference rule is as follows:

S1, S2, …, SnS

This rule states that if S1, S2, . . . , and S*n* are true, then the truth of S can be inferred. The top part of an inference rule is called its **antecedent**; the bottom part is called its **consequent**.

An **axiom** is a logical statement that is assumed to be true. Therefore, an axiom is an inference rule without an antecedent.

For some program statements, the computation of a weakest precondition from the statement and a postcondition is simple and can be specified by an axiom. In most cases, however, the weakest precondition can be specified only by an inference rule.

To use axiomatic semantics with a given programming language, whether for correctness proofs or for formal semantics specifications, either an axiom or an inference rule must exist for each kind of statement in the language. In the following subsections, we present an axiom for assignment statements and inference rules for statement sequences, selection statements, and logical pretest loop statements. Note that we assume that neither arithmetic nor - Boolean expressions have side effects.

3.5.3.3 Assignment Statements

The precondition and postcondition of an assignment statement together define its meaning. To define the meaning of an assignment statement there must be a way to compute its precondition from its postcondition.

Let x = E be a general assignment statement and Q be its postcondition. Then, its weakest precondition, P, is defined by the axiom

- $P = Q_{xSE}$

which means that P is computed as Q with all instances of x replaced by E. For example, if we have the assignment statement and postcondition

```
a = b / 2 - 1 {a < 10}
```

the weakest precondition is computed by substituting `b / 2 - 1` for `a` in the postcondition `{a < 10}`, as follows:

```
b / 2 - 1 < 10
b < 22
```

Thus, the weakest precondition for the given assignment statement and postcondition is `{b < 22}`. Remember that the assignment axiom is guaranteed to be correct only in the absence of side effects. An assignment statement has a side effect if it changes some variable other than its target.

The usual notation for specifying the axiomatic semantics of a given statement form is

{P} S {Q}

where P is the precondition, Q is the postcondition, and S is the statement form. In the case of the assignment statement, the notation is

- $\{Q_{x \to E}\}\ x = E \{Q\}$

As another example of computing a precondition for an assignment statement, consider the following:

```
x = 2 * y - 3 {x > 25}
```

The precondition is computed as follows:

```
2 * y - 3 > 25
y > 14
```

So `{y > 14}` is the weakest precondition for this assignment statement and postcondition.

Note that the appearance of the left side of the assignment statement in its right side does not affect the process of computing the weakest precondition. For example, for

```
x = x + y - 3 {x > 10}
```

the weakest precondition is

```
x + y - 3 > 10
y > 13 - x
```

Recall that axiomatic semantics was developed to prove the correctness of programs. In light of that, it is natural at this point to wonder how the axiom for assignment statements can be used to prove anything. Here is how: A given assignment statement with both a precondition and a postcondition can be considered a logical statement, or theorem. If the assignment axiom, when applied to the postcondition and the assignment statement, produces the given precondition, the theorem is proved. For example, consider the following logical statement:

```
{x > 3} x = x - 3 {x > 0}
```

Using the assignment axiom on the statement and its postcondition produces `{x > 3}`, which is the given precondition. Therefore, we have proven the example logical statement.

Next, consider the following logical statement:

```
{x > 5} x = x - 3 {x > 0}
```

In this case, the given precondition, `{x > 5}`, is not the same as the assertion produced by the axiom. However, it is obvious that `{x > 5}` implies `{x > 3}`. To use this in a proof, an inference rule named the **rule of consequence** is needed. The form of the rule of consequence is

{P} S {Q}, P′ => P, Q => Q′{P′} S{Q′}

The => symbol means “implies,” and S can be any program statement. The rule can be stated as follows: If the logical statement {P} S{Q} is true, the assertion P′ implies the assertion P, and the assertion Q implies the assertion Q′, then it can be inferred that {P′} S{Q′}. In other words, the rule of consequence says that a postcondition can always be weakened and a precondition can always be strengthened. This is quite useful in program proofs. For example, it allows the completion of the proof of the last logical statement example above. If we let P be `{x > 3}`, Q and Q′ be `{x > 0}`, and P

′ be {x > 5}, we have

{x > 3}x = x − 3{x > 0},(x > 5)=>{x > 3},(x > 0)=>(x > 0)
{x > 5}x = x − 3{x > 0}

The first term of the antecedent ({x > 3} x = x - 3 {x > 0}) was proven with the assignment axiom. The second and third terms are obvious. Therefore, by the rule of consequence, the consequent is true.

3.5.3.4 Sequences

The weakest precondition for a sequence of statements cannot be described by an axiom, because the precondition depends on the particular kinds of statements in the sequence. In this case, the precondition can only be described with an inference rule. Let S1 and S2 be adjacent program statements. If S1 and S2 have the following pre- and postconditions

{P1} S1 {P2} {P2} S2 {P3}

the inference rule for such a two-statement sequence is

{P1} S1 {P2}, {P2} S2 {P3}{P1} S1, S2 {P3}

So, for our example, {P1} S1; S2 {P3} describes the axiomatic semantics of the sequence S1; S2. The inference rule states that to get the sequence precondition, the precondition of the second statement is computed. This new assertion is then used as the postcondition of the first statement, which can then be used to compute the precondition of the first statement, which is also the precondition of the whole sequence. If S1 and S2 are the assignment statements

x1=E1

and

x2=E2

then we have

{P3x2→E2} x2=E2 {P3} {(P3x2→E2)x1→E1} x1=E1 {P3x2→E2}

Therefore, the weakest precondition for the sequence x1 = E1; x2 = E2 with postcondition P3 is {(P3x2→E2)x1→E1}.

For example, consider the following sequence and postcondition:

```
y = 3 * x + 1;
x = y + 3;
{x < 10}
```

The precondition for the second assignment statement is

```
y < 7
```

which is used as the postcondition for the first statement. The precondition for the first assignment statement can now be computed:

```
3 * x + 1 < 7
x < 2
```

So, `{x < 2}` is the precondition of both the first statement and the two--statement sequence.

3.5.3.5 Selection

We next consider the inference rule for selection statements, the general form of which is

- **`if`** B **`then`** S1 **`else`** S2

We consider only selections that include **`else`** clauses. The inference rule is

{B and P} S1 {Q}, {(not B) and P} S2 {Q}
{P} if B bb{then} S1 bb{else} S2 {Q}}

This rule specifies that selection statements must be proven both when the

Boolean control expression is true and when it is false. The first logical statement above the line represents the **`then`** clause; the second represents the **`else`** clause. According to the inference rule, we need a precondition P that can be used in the precondition of both the **`then`** and **`else`** clauses.

Consider the following example of the computation of the precondition using the selection inference rule. The example selection statement is

```
if x > 0 then
  y = y - 1
else
  y = y + 1
```

Suppose the postcondition, Q, for this selection statement is `{y > 0}`. We can use the axiom for assignment on the **`then`** clause

```
y = y - 1 {y > 0}
```

This produces `{y - 1 > 0}` or `{y > 1}`. It can be used as the P part of the precondition for the **`then`** clause. Now we apply the same axiom to the **`else`** clause

```
y = y + 1 {y > 0}
```

which produces the precondition `{y + 1 > 0}` or `{y > -1}`. Because `{y > 1}` => `{y > -1}`, the rule of consequence allows us to use `{y > 1}` for the precondition of the whole selection statement.

3.5.3.6 Logical Pretest Loops

Another essential construct of imperative programming languages is the logical pretest, or **`while`** loop. Computing the weakest precondition for a **`while`** loop is inherently more difficult than for a sequence, because the number of iterations cannot always be predetermined. In a case where the number of iterations is known, the loop can be unrolled and treated as a sequence.

The problem of computing the weakest precondition for loops is similar to

the problem of proving a theorem about all positive integers. In the latter case, induction is normally used, and the same inductive method can be used for some loops. The principal step in induction is finding an inductive hypothesis. The corresponding step in the axiomatic semantics of a **`while`** loop is finding an assertion called a **loop invariant**, which is crucial to finding the weakest precondition.

The inference rule for computing the precondition for a **`while`** loop is as follows:

{I and B} S {I}{I} while B do S end {I and (not B)}

In this rule, I is the loop invariant. This seems simple, but it is not. The complexity lies in finding an appropriate loop invariant.

The axiomatic description of a **`while`** loop is written as

- {P} **`while`** B **`do`** S **`end`** {Q}

The loop invariant must satisfy a number of requirements to be useful. First, the weakest precondition for the **`while`** loop must guarantee the truth of the loop invariant. In turn, the loop invariant must guarantee the truth of the postcondition upon loop termination. These constraints move us from the inference rule to the axiomatic description. During execution of the loop, the truth of the loop invariant must be unaffected by the evaluation of the loop-controlling Boolean expression and the loop body statements. Hence, the name *invariant*.

Another complicating factor for **`while`** loops is the question of loop termination. A loop that does not terminate cannot be correct, and in fact computes nothing. If Q is the postcondition that holds immediately after loop exit, then a precondition P for the loop is one that guarantees Q at loop exit and also guarantees that the loop terminates.

The complete axiomatic description of a **`while`** construct requires all of the following to be true, in which I is the loop invariant:

- P => I

- {I and B} S { I }
- (I and (not B)) => Q

the loop terminates

If a loop computes a sequence of numeric values, it may be possible to find a loop invariant using an approach that is used for determining the inductive hypothesis when mathematical induction is used to prove a statement about a mathematical sequence. The relationship between the number of iterations and the precondition for the loop body is computed for a few cases, with the hope that a pattern emerges that will apply to the general case. It is helpful to treat the process of producing a weakest precondition as a function, wp. In general

wp(statement, postcondition)=precondition

A wp function is often called a **predicate transformer**, because it takes a predicate, or assertion, as a parameter and returns another predicate.

To find I, the loop postcondition Q is used to compute preconditions for several different numbers of iterations of the loop body, starting with none. If the loop body contains a single assignment statement, the axiom for assignment statements can be used to compute these cases. Consider the example loop:

```
while y <> x do y = y + 1 end {y = x}
```

Remember that the equal sign is being used for two different purposes here. In assertions, it means mathematical equality; outside assertions, it means the assignment operator.

For zero iterations, the weakest precondition is, obviously,

```
{y = x}
```

For one iteration, it is

- `wp(y = y + 1, {y = x}) = {y + 1 = x}`, or `{y = x - 1}`

For two iterations, it is

- wp(y = y + 1, {y = x - 1})={y + 1 = x - 1}, or {y = x - 2}

For three iterations, it is

- wp(y = y + 1, {y = x - 2})={y + 1 = x - 2}, or {y = x – 3}

It is now obvious that {y < x} will suffice for cases of one or more iterations. Combining this with {y = x} for the zero iterations case, we get {y <= x}, which can be used for the loop invariant. A precondition for the **while** statement can be determined from the loop invariant. In fact, I can be used as the precondition, P.

We must ensure that our choice satisfies the four criteria for I for our example loop. First, because P = I, P => I. The second requirement is that the following must be true:

{ I and B } S { I }

In our example, we have

{y <= x and y <> x} y = y + 1 {y <= x}

Applying the assignment axiom to

y = y + 1 {y <= x}

we get {y + 1 <= x}, which is equivalent to {y < x}, which is implied by {y <= x and y <> x}. So, the earlier statement is proven.

Next, we must have

- {I and (not B)} => Q

In our example, we have

- {(y <= x) and not (y <> x)} => {y = x}
- {(y <= x) and (y = x)} => {y = x}

- `{y = x} => {y = x}`

So, this is obviously true. Next, loop termination must be considered. In this example, the question is whether the loop

```
{y <= x} while y <> x do y = y + 1 end {y = x}
```

terminates. Recalling that `x` and `y` are assumed to be integer variables, it is easy to see that this loop does terminate. The precondition guarantees that `y` initially is not larger than `x`. The loop body increments `y` with each iteration, until `y` is equal to `x`. No matter how much smaller `y` is than `x` initially, it will eventually become equal to `x`. So the loop will terminate. Because our choice of I satisfies all four criteria, it is a satisfactory loop invariant and loop precondition.

The previous process used to compute the invariant for a loop does not always produce an assertion that is the weakest precondition (although it does in the example).

As another example of finding a loop invariant using the approach used in mathematical induction, consider the following loop statement:

```
while  s > 1  do  s = s / 2  end  {s = 1}
```

As before, we use the assignment axiom to try to find a loop invariant and a precondition for the loop. For zero iterations, the weakest precondition is `{s = 1}`.

For one iteration, it is

- `wp(s = s / 2, {s = 1}) = {s / 2 = 1}`, or `{s = 2}`

For two iterations, it is

- `wp(s = s / 2, {s = 2}) = {s / 2 = 2}`, or `{s = 4}`

For three iterations, it is

- `wp(s = s / 2, {s = 4}) = {s / 2 = 4}`, or `{s = 8}`

From these cases, we can see clearly that the invariant is

- {s is a nonnegative power of 2}

Once again, the computed I can serve as P, and I passes the four requirements. Unlike our earlier example of finding a loop precondition, this one clearly is not a weakest precondition. Consider using the precondition {s > 1}. The logical statement

```
{s > 1} while  s > 1 do  s = s / 2 end {s = 1}
```

can easily be proven, and this precondition is significantly broader than the one computed earlier. The loop and precondition are satisfied for any positive value for s, not just powers of 2, as the process indicates. Because of the rule of consequence, using a precondition that is stronger than the weakest precondition does not invalidate a proof.

Finding loop invariants is not always easy. It is helpful to understand the nature of these invariants. First, a loop invariant is a weakened version of the loop postcondition and also a precondition for the loop. So, I must be weak enough to be satisfied prior to the beginning of loop execution, but when combined with the loop exit condition, it must be strong enough to force the truth of the postcondition.

Because of the difficulty of proving loop termination, that requirement is often ignored. If loop termination can be shown, the axiomatic description of the loop is called **total correctness**. If the other conditions can be met but termination is not guaranteed, it is called **partial correctness**.

In more complex loops, finding a suitable loop invariant, even for partial correctness, requires a good deal of ingenuity. Because computing the precondition for a **`while`** loop depends on finding a loop invariant, proving the correctness of programs with **`while`** loops using axiomatic semantics can be difficult.

3.5.3.7 Program Proofs

This section provides validations for two simple programs. The first example of a correctness proof is for a very short program, consisting of a sequence of three assignment statements that interchange the values of two variables.

```
{x = A AND y = B}
t = x;
x = y;
y = t;
{x = B AND y = A}
```

Because the program consists entirely of assignment statements in a sequence, the assignment axiom and the inference rule for sequences can be used to prove its correctness. The first step is to use the assignment axiom on the last statement and the postcondition for the whole program. This yields the precondition

```
{x = B AND t = A}
```

Next, we use this new precondition as a postcondition on the middle statement and compute its precondition, which is

```
{y = B AND t = A}
```

Next, we use this new assertion as the postcondition on the first statement and apply the assignment axiom, which yields

```
{y = B AND x = A}
```

which is the same as the precondition on the program, except for the order of operands on the `AND` operator. Because `AND` is a symmetric operator, our proof is complete.

The following example is a proof of correctness of a pseudocode program that computes the factorial function.

```
{n >= 0}
count = n;
fact = 1;
while count <> 0 do
    fact = fact * count;
    count = count - 1;
end
```

```
{fact = n!}
```

The method described earlier for finding the loop invariant does not work for the loop in this example. Some ingenuity is required here, which can be aided by a brief study of the code. The loop computes the factorial function in order of the last multiplication first; that is, `(n - 1) * n` is done first, assuming `n` is greater than `1`. So, part of the invariant can be the following:

```
fact = (count + 1) * (count + 2) * . . . * (n - 1) * n
```

But we also must ensure that `count` is always nonnegative, which we can do by adding that to the assertion above, to get the following:

```
I = (fact = (count + 1) * . . . * n) AND (count >= 0)
```

Next, we must confirm that this I meets the requirements for invariants. Once again we let I also be used for P, so P clearly implies I. The next question is

{ I and B } S { I }

I and B is the following:

```
((fact = (count + 1) * . . . * n) AND (count >= 0)) AND
    (count <> 0)
```

This reduces to

```
(fact = (count + 1) * . . . * n) AND (count > 0)
```

In our case, we must compute the precondition of the body of the loop, using the invariant for the postcondition. For

- {P} `count = count - 1 {I}`

we compute P to be

```
{(fact = count * (count + 1) * . . . * n) AND
    (count >= 1)}
```

Using this as the postcondition for the first assignment in the loop body,

- {P} `fact = fact * count {(fact = count * (count + 1)`
- `* . . . * n) AND (count >= 1)}`

In this case, P is

```
{(fact = (count + 1) * . . . * n) AND (count >= 1)}
```

It is clear that I and B implies this P, so by the rule of consequence,

- {I `AND` B} S {`I`}

is true. Finally, the last test of I is

- I `AND` (`NOT` B) => Q

For our example, this is

```
((fact = (count + 1) * . . . * n) AND (count >= 0)) AND
(count = 0)) => fact = n!
```

This is clearly true, for when `count = 0`, the first part is precisely the definition of factorial. So, our choice of I meets the requirements for a loop invariant. Now we can use our P (which is the same as I) from the **`while`** as the postcondition on the second assignment of the program

```
{P} fact = 1 {(fact = (count + 1) * . . . * n) AND
    (count >= 0)}
```

which yields for P

```
(1 = (count + 1) * . . . * n) AND (count >= 0))
```

Using this as the postcondition for the first assignment in the code

```
{P} count = n {(1 = (count + 1) * . . . * n) AND
    (count >= 0))}
```

produces for P

```
{(n + 1) * . . . * n = 1) AND (n >= 0)}
```

The left operand of the `AND` operator is true (because `1 = 1`) and the right operand is exactly the precondition of the whole code segment, `{n >= 0}`. Therefore, the program has been proven to be correct.

3.5.3.8 Evaluation

As stated previously, to define the semantics of a complete programming language using the axiomatic method, there must be an axiom or an inference rule for each statement type in the language. Defining axioms or inference rules for some of the statements of programming languages has proven to be a difficult task. An obvious solution to this problem is to design the language with the axiomatic method in mind, so that only statements for which axioms or inference rules can be written are included. Unfortunately, such a language would necessarily leave out some useful and powerful statements.

Axiomatic semantics is a powerful tool for research into program correctness proofs, and it provides an excellent framework in which to reason about programs, both during their construction and later. Its usefulness in describing the meaning of programming languages to language users and compiler writers is, however, highly limited.

SUMMARY

Backus-Naur Form and context-free grammars are equivalent metalanguages that are well suited for the task of describing the syntax of programming languages. Not only are they concise descriptive tools, but also the parse trees that can be associated with their generative actions give graphical evidence of the underlying syntactic structures. Furthermore, they are naturally related to recognition devices for the languages they generate, which leads to the relatively easy construction of syntax analyzers for compilers for these languages.

An attribute grammar is a descriptive formalism that can describe both the syntax and static semantics of a language. Attribute grammars are extensions to context-free grammars. An attribute grammar consists of a grammar, a set of attributes, a set of attribute computation functions, and a set of predicates that describe static semantics rules.

This chapter provides brief introductions to three methods of semantic description: operational, denotational, and axiomatic. Operational semantics is a method of describing the meaning of language constructs in terms of their effects on an ideal machine. In denotational semantics, mathematical objects are used to represent the meanings of language constructs. Language entities are converted to these mathematical objects with recursive functions. Axiomatic semantics, which is based on formal logic, was devised as a tool for proving the correctness of programs.

BIBLIOGRAPHIC NOTES

Syntax description using context-free grammars and BNF are thoroughly - discussed in Cleaveland and Uzgalis (1976).

Research in axiomatic semantics was begun by Floyd (1967) and further developed by Hoare (1969). The semantics of a large part of Pascal was described by Hoare and Wirth (1973) using this method. The parts they did not complete involved functional side effects and goto statements. These were found to be the most difficult to describe.

The technique of using preconditions and postconditions during the development of programs is described (and advocated) by Dijkstra (1976) and also discussed in detail in Gries (1981).

Good introductions to denotational semantics can be found in Gordon (1979) and Stoy (1977). Introductions to all of the semantics description methods discussed in this chapter can be found in Marcotty et al. (1976). Another good reference for much of the chapter material is Pagan (1981). The form of the denotational semantic functions in this chapter is similar to that found in Meyer (1990).

REVIEW QUESTIONS

1. Define *syntax* and *semantics*.
2. Who are language descriptions for?
3. Describe the operation of a general language generator.
4. Describe the operation of a general language recognizer.
5. What is the difference between a sentence and a sentential form?
6. Define a left-recursive grammar rule.
7. What three extensions are common to most EBNFs?
8. Distinguish between static and dynamic semantics.
9. What purpose do predicates serve in an attribute grammar?
10. What is the difference between a synthesized and an inherited attribute?
11. How is the order of evaluation of attributes determined for the trees of a given attribute grammar?
12. What is the primary use of attribute grammars?
13. Explain the primary uses of a methodology and notation for describing the semantics of programming languages.
14. Why can machine languages not be used to define statements in operational semantics?
15. Describe the two levels of uses of operational semantics.
16. In denotational semantics, what are the syntactic and semantic domains?

17. What is stored in the state of a program for denotational semantics?

18. Which semantics approach is most widely known?

19. What two things must be defined for each language entity in order to construct a denotational description of the language?

20. Which part of an inference rule is the antecedent?

21. What is a predicate transformer function?

22. What does partial correctness mean for a loop construct?

23. On what branch of mathematics is axiomatic semantics based?

24. On what branch of mathematics is denotational semantics based?

25. What is the problem with using a software pure interpreter for operational semantics?

26. Explain what the preconditions and postconditions of a given statement mean in axiomatic semantics.

27. Describe the approach of using axiomatic semantics to prove the correctness of a given program.

28. Describe the basic concept of denotational semantics.

29. In what fundamental way do operational semantics and denotational semantics differ?

PROBLEM SET

1. The two mathematical models of language description are generation and recognition. Describe how each can define the syntax of a programming language.

2. Write EBNF descriptions for the following:

    1. A Java class definition header statement

    2. A Java method call statement

    3. A C **`switch`** statement

    4. A C **`union`** definition

    5. C **`float`** literals

3. Rewrite the BNF of Example 3.4 to give `+` precedence over `*` and force `+` to be right associative.

4. Rewrite the BNF of Example 3.4 to add the `++` and `--` unary operators of Java.

5. Write a BNF description of the Boolean expressions of Java, including the three operators `&&`, `||`, and `!` and the relational expressions.

6. Using the grammar in Example 3.2, show a parse tree and a leftmost derivation for each of the following statements:

    1. `A = A * (B + (C * A))`

    2. `B = C * (A * C + B)`

    3. `A = A * (B + (C))`

7. Using the grammar in Example 3.4, show a parse tree and a leftmost derivation for each of the following statements:
    1. `A = (A + B) * C`
    2. `A = B + C + A`
    3. `A = A * (B + C)`
    4. `A = B * (C * (A + B))`
8. Prove that the following grammar is ambiguous:
    - <S> → <A>
    - <A> → <A> + <A> | <id>
    - <id> → a | b | c
9. Modify the grammar of Example 3.4 to add a unary minus operator that has higher precedence than either + or *.
10. Describe, in English, the language defined by the following grammar:
    - <S> → <A> <B> <C>
    - <A> → a <A> | a
    - <B> → b <B> | b
    - <C> → c <C> | c
11. Consider the following grammar:
    - <S> → <A> a <B> b
    - <A> → <A> b | b
    - <B> → a <B> | a

Which of the following sentences are in the language generated by this grammar?

1. baab
2. bbbab
3. bbaaaaas
4. bbaab

12. Consider the following grammar:

    - <S> → a <S> c <B> | <A> | b
    - <A> → c <A> | c
    - <B> → d | <A>

    Which of the following sentences are in the language generated by this grammar?

    1. abcd
    2. acccbd
    3. acccbcc
    4. acd
    5. accc

13. Write a grammar for the language consisting of strings that have *n* copies of the letter a followed by the same number of copies of the letter b, where $n > 0$. For example, the strings ab, aaaabbbb, and aaaaaaaabbbbbbbb are in the language but a, abb, ba, and aaabb are not.

14. Draw parse trees for the sentences aabb and aaaabbbb, as derived from the grammar of Problem 13.

15. Convert the BNF of Example 3.1 to EBNF.

16. Convert the BNF of Example 3.3 to EBNF.

17. Convert the following EBNF to BNF:

    - $S \rightarrow A\{bA\}$
    - $A \rightarrow a[b]A$

18. What is the difference between an intrinsic attribute and a nonintrinsic synthesized attribute?

19. Write an attribute grammar whose BNF basis is that of Example 3.6 in Section 3.4.5 but whose language rules are as follows: Data types cannot be mixed in expressions, but assignment statements need not have the same types on both sides of the assignment operator.

20. Write an attribute grammar whose base BNF is that of Example 3.2 and whose type rules are the same as for the assignment statement example of Section 3.4.5.

21. Using the virtual machine instructions given in Section 3.5.1.1, give an operational semantic definition of the following:

    1. Java **`do-while`**
    2. Ada **`for`**
    3. C++ **`if-then-else`**
    4. C **`for`**
    5. C **`switch`**

22. Write a denotational semantics mapping function for the following statements:

    1. Ada **`for`**

2. Java **`do-while`**
3. Java Boolean expressions
4. Java **`for`**
5. C **`switch`**

23. Compute the weakest precondition for each of the following assignment statements and postconditions:
    1. `a = 2 * (b - 1) - 1 {a > 0}`
    2. `b = (c + 10) / 3 {b > 6}`
    3. `a = a + 2 * b - 1 {a > 1}`
    4. `x = 2 * y + x - 1 {x > 11}`
24. Compute the weakest precondition for each of the following sequences of assignment statements and their postconditions:
    1. ```
       a = 2 * b + 1;
       b = a - 3
       {b < 0}
       ```
    2. ```
       a = 3 * (2 * b + a);
       b = 2 * a - 1
       {b > 5}
       ```
25. Compute the weakest precondition for each of the following selection constructs and their postconditions:
    1. ```
       if (a == b)
          b = 2 * a + 1
       ```
       ```

```
   else
    b = 2 * a;
   {b > 1}
```

2.
```
if (x < y)
  x = x + 1
else
  x = 3 * x
{x < 0}
```

3.
```
if (x > y)
   y = 2 * x + 1
else
   y = 3 * x - 1;
{y > 3}
```

26. Explain the four criteria for proving the correctness of a logical pretest loop construct of the form **`while`** B **`do`** S **`end`**.

27. Prove that (n+1) *…* n=1.

28. Prove the following program is correct:

```
{n > 0}
count = n;
sum = 0;
while  count <> 0  do
  sum = sum + count;
  count = count - 1;
end
{sum = 1 + 2 + . . . + n}
```

4 Lexical and Syntax Analysis

This chapter begins with an introduction to lexical analysis, along with a - simple example. Next, the general parsing problem is discussed, including the two primary approaches to parsing, and the complexity of parsing. Then, we - introduce the recursive-descent implementation technique for top-down parsers, including examples of parts of a recursive-descent parser and a trace of a parse using one. The last section discusses bottom-up parsing and the LR parsing algorithm. This section includes an example of a small LR parsing table and the parse of a string using the LR parsing process.

4.1 Introduction

A serious investigation of compiler design requires at least a semester of intensive study, including the design and implementation of a compiler for a small but realistic programming language. The first part of such a course is devoted to lexical and syntax analyses. The syntax analyzer is the heart of a compiler, because several other important components, including the semantic analyzer and the intermediate code generator, are driven by the actions of the syntax analyzer.

Some readers may wonder why a chapter on any part of a compiler would be included in a book on programming languages. There are at least two reasons to include a discussion of lexical and syntax analyses in this text: First, syntax analyzers are based directly on the grammars discussed in Chapter 3, so it is natural to discuss them as an application of grammars. Second, lexical and syntax analyzers are needed in numerous situations outside compiler design. Many applications, among them program listing formatters, programs that compute the complexity of programs, and programs that must analyze and react to the contents of a configuration file, all need to do lexical and syntax analyses. Therefore, lexical and syntax analyses are important topics for software developers, even if they never need to write a compiler. Furthermore, some computer science programs no longer require students to take a compiler design course, which leaves students with no instruction in lexical or syntax analysis. In those cases, this chapter can be covered in the programming language course. In degree programs that require a compiler design course, this chapter can be skipped.

Three different approaches to implementing programming languages are introduced in Chapter 1: compilation, pure interpretation, and hybrid implementation. The compilation approach uses a program called a compiler, which translates programs written in a high-level programming language into machine code. Compilation is typically used to implement programming languages that are used for large applications, often written in languages such as C++ and COBOL. Pure interpretation systems perform no translation; rather, programs are interpreted in their original form by a software

interpreter. Pure interpretation is usually used for smaller systems in which execution efficiency is not critical, such as scripts embedded in HTML documents, written in languages such as JavaScript. Hybrid implementation systems translate programs written in high-level languages into intermediate forms, which are interpreted. These systems are now more widely used than ever, thanks in large part to the popularity of scripting languages. Traditionally, hybrid systems have resulted in much slower program execution than compiler systems. However, in recent years the use of Just-in-Time (JIT) compilers has become widespread, particularly for Java programs and programs written for the Microsoft .NET system. A JIT compiler, which translates intermediate code to machine code, is used on methods at the time they are first called. In effect, a JIT compiler transforms a hybrid system to a delayed compiler system.

All three of the implementation approaches just discussed use both lexical and syntax analyzers.

Syntax analyzers, or parsers, are nearly always based on a formal description of the syntax of programs. The most commonly used syntax-description formalism is context-free grammars, or BNF, which is introduced in Chapter 3. Using BNF, as opposed to using some informal syntax description, has at least three compelling advantages. First, BNF descriptions of the syntax of programs are clear and concise, both for humans and for software systems that use them. Second, the BNF description can be used as the direct basis for the syntax analyzer. Third, implementations based on BNF are relatively easy to maintain because of their modularity.

Nearly all compilers separate the task of analyzing syntax into two distinct parts, lexical analysis and syntax analysis, although this terminology is confusing. The lexical analyzer deals with small-scale language constructs, such as names and numeric literals. The syntax analyzer deals with the large-scale constructs, such as expressions, statements, and program units. Section 4.2 introduces lexical analyzers. Sections 4.3, 4.4, and 4.5 discuss syntax analyzers.

There are three reasons why lexical analysis is separated from syntax analysis:

1. Simplicity—Techniques for lexical analysis are less complex than those required for syntax analysis, so the lexical-analysis process can be simpler if it is separate. Also, removing the low-level details of lexical analysis from the syntax analyzer makes the syntax analyzer both smaller and less complex.

2. Efficiency—Although it pays to optimize the lexical analyzer, because lexical analysis requires a significant portion of total compilation time, it is not fruitful to optimize the syntax analyzer. Separation facilitates this selective optimization.

3. Portability—Because the lexical analyzer reads input program files and often includes buffering of that input, it is somewhat platform dependent. However, the syntax analyzer can be platform independent. It is always good to isolate machine-dependent parts of any software system.

4.2 Lexical Analysis

A lexical analyzer is essentially a pattern matcher. A pattern matcher attempts to find a substring of a given string of characters that matches a given character pattern. Pattern matching is a traditional part of computing. One of the earliest uses of pattern matching was with text editors, such as the `ed` line editor, which was introduced in an early version of UNIX. Since then, pattern matching has found its way into some programming languages—for example, Perl and JavaScript. It is also available through the standard class libraries of Java, C++, and C#.

A lexical analyzer serves as the front end of a syntax analyzer. Technically, lexical analysis is a part of syntax analysis. A lexical analyzer performs syntax analysis at the lowest level of program structure. An input program appears to a compiler as a single string of characters. The lexical analyzer collects characters into logical groupings and assigns internal codes to the groupings according to their structure. In Chapter 3, these logical groupings are named **lexemes**, and the internal codes for categories of these groupings are named **tokens**. Lexemes are recognized by matching the input character string against character string patterns. Although tokens are usually represented as integer values, for the sake of readability of lexical and syntax analyzers, they are often referenced through named constants.

Consider the following example of an assignment statement:

```
result = oldsum - value / 100;
```

Following are the tokens and lexemes of this statement:

Token	*Lexeme*
IDENT	result
ASSIGN_OP	=
IDENT	oldsum
SUB_OP	-
IDENT	value
DIV_OP	/
INT_LIT	100
SEMICOLON	;

Lexical analyzers extract lexemes from a given input string and produce the corresponding tokens. In the early days of compilers, lexical analyzers often processed an entire source program file and produced a file of tokens and lexemes. Now, however, most lexical analyzers are subprograms that locate the next lexeme in the input, determine its associated token code, and return them to the caller, which is the syntax analyzer. So, each call to the lexical analyzer returns a single lexeme and its token. The only view of the input program seen by the syntax analyzer is the output of the lexical analyzer, one token at a time.

The lexical-analysis process includes skipping comments and white space outside lexemes, as they are not relevant to the meaning of the program. Also, the lexical analyzer inserts lexemes for user-defined names into the symbol table, which is used by later phases of the compiler. Finally, lexical analyzers detect syntactic errors in tokens, such as ill-formed floating-point literals, and report such errors to the user.

There are three approaches for building a lexical analyzer:

1. Write a formal description of the token patterns of the language using a

descriptive language related to regular expressions.1 These descriptions are used as input to a software tool that automatically generates a lexical analyzer. There are many such tools available for this. The oldest of these, named lex, is commonly included as part of UNIX systems.

1. These regular expressions are the basis for the pattern-matching facilities now part of many programming languages, either directly or through a class library.

2. Design a state transition diagram that describes the token patterns of the language and write a program that implements the diagram.

3. Design a state transition diagram that describes the token patterns of the language and hand-construct a table-driven implementation of the state diagram.

A state transition diagram, or just **state diagram**, is a directed graph. The nodes of a state diagram are labeled with state names. The arcs are labeled with the input characters that cause the transitions among the states. An arc may also include actions the lexical analyzer must perform when the transition is taken.

State diagrams of the form used for lexical analyzers are representations of a class of mathematical machines called **finite automata**. Finite automata can be designed to recognize members of a class of languages called **regular languages**. Regular grammars are generative devices for regular languages. The tokens of a programming language are a regular language, and a lexical analyzer is a finite automaton.

We now illustrate lexical-analyzer construction with a state diagram and the code that implements it. The state diagram could simply include states and transitions for each and every token pattern. However, that approach results in a very large and complex diagram, because every node in the state diagram would need a transition for every character in the character set of the language being analyzed. We therefore consider ways to simplify it.

Suppose we need a lexical analyzer that recognizes only arithmetic expressions, including variable names and integer literals as operands.

Assume that the variable names consist of strings of uppercase letters, lowercase letters, and digits but must begin with a letter. Names have no length limitation. The first thing to observe is that there are 52 different characters (any uppercase or lowercase letter) that can begin a name, which would require 52 transitions from the transition diagram's initial state. However, a lexical analyzer is interested only in determining that it is a name and is not concerned with which specific name it happens to be. Therefore, we define a character class named LETTER for all 52 letters and use a single transition on the first letter of any name.

Another opportunity for simplifying the transition diagram is with the integer literal tokens. There are 10 different characters that could begin an integer literal lexeme. This would require 10 transitions from the start state of the state diagram. Because specific digits are not a concern of the lexical analyzer, we can build a much more compact state diagram if we define a character class named DIGIT for digits and use a single transition on any character in this character class to a state that collects integer literals.

Because our names can include digits, the transition from the node following the first character of a name can use a single transition on LETTER or DIGIT to continue collecting the characters of a name.

Next, we define some utility subprograms for the common tasks inside the lexical analyzer. First, we need a subprogram, which we can name `getChar`, that has several duties. When called, `getChar` gets the next character of input from the input program and puts it in the global variable `nextChar`. `getChar` also must determine the character class of the input character and put it in the global variable `charClass`. The lexeme being built by the lexical analyzer, which could be implemented as a character string or an array, will be named `lexeme`.

We implement the process of putting the character in `nextChar` into the string array `lexeme` in a subprogram named `addChar`. This subprogram must be explicitly called because programs include some characters that need not be put in `lexeme`, for example the white-space characters between lexemes. In a more realistic lexical analyzer, comments also would not be placed in `lexeme`.

When the lexical analyzer is called, it is convenient if the next character of input is the first character of the next lexeme. Because of this, a function named `getNonBlank` is used to skip white space every time the analyzer is called.

Finally, a subprogram named `lookup` is needed to compute the token code for the single-character tokens. In our example, these are parentheses and the arithmetic operators. Token codes are numbers arbitrarily assigned to tokens by the compiler writer.

The state diagram in Figure 4.1 describes the patterns for our tokens. It includes the actions required on each transition of the state diagram.

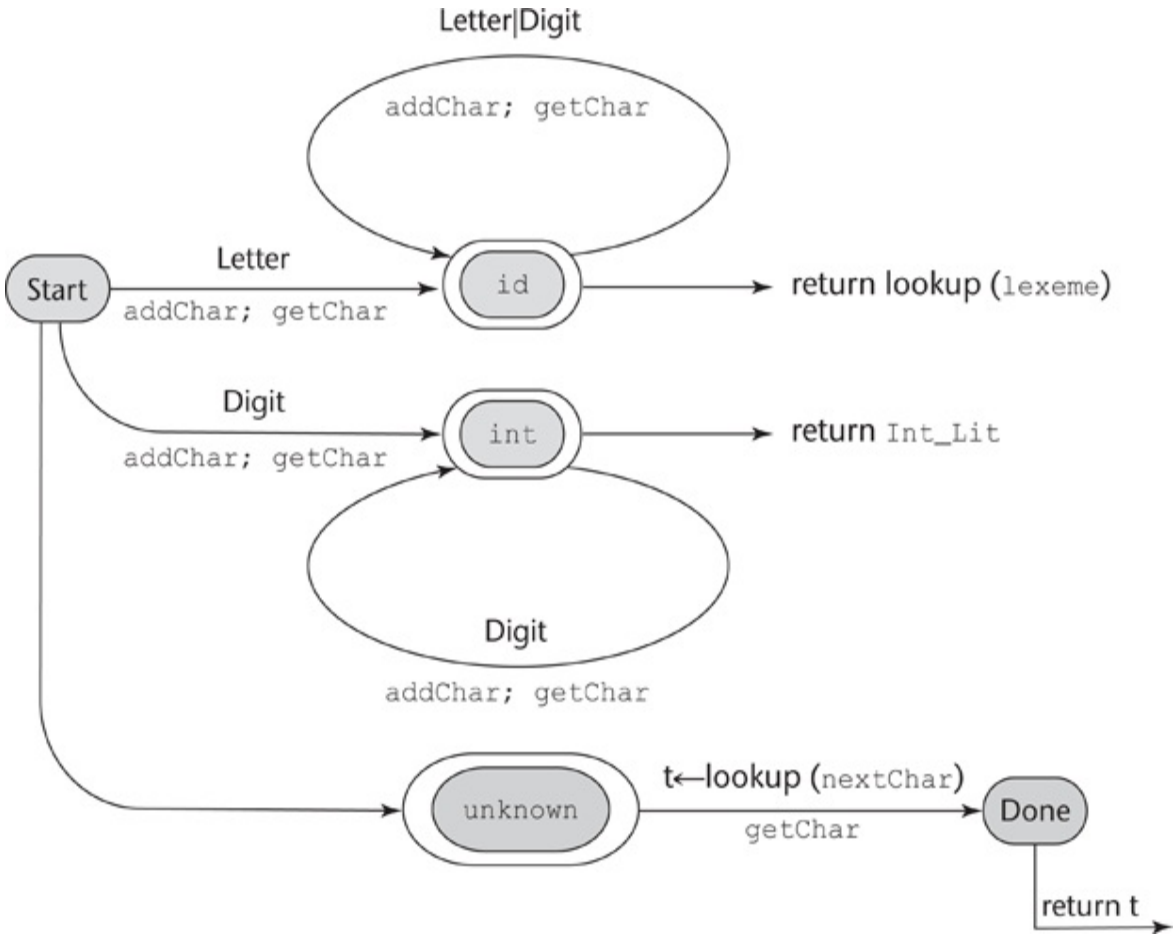


Figure 4.1 A state diagram to

recognize names, parentheses, and arithmetic operators

Figure 4.1 Full Alternative Text

The following is a C implementation of a lexical analyzer specified in the state diagram of Figure 4.1, including a main driver function for testing purposes:

```
/* front.c - a lexical analyzer system for simple
             arithmetic expressions */
#include <stdio.h>
#include <ctype.h>
/* Global declarations */
/* Variables */
int  charClass;
char  lexeme [100];
char  nextChar;
int  lexLen;
int  token;
int  nextToken;
FILE *in_fp, *fopen();
/* Function declarations */
void  addChar();
void  getChar();
void  getNonBlank();
int  lex();
/* Character classes */
#define LETTER 0
#define DIGIT 1
#define UNKNOWN 99
/* Token codes */
#define INT_LIT 10
#define IDENT 11
#define ASSIGN_OP 20
#define ADD_OP 21
#define SUB_OP 22
#define MULT_OP 23
#define DIV_OP 24
#define LEFT_PAREN 25
#define RIGHT_PAREN 26
/*****************************************************/
```

```
/* main driver */
main() {
/* Open the input data file and process its contents */
   if  ((in_fp = fopen("front.in", "r")) == NULL)
     printf("ERROR - cannot open front.in \n");
   else  {
     getChar();
     do   {
       lex();
    }  while  (nextToken! = EOF);
  }
}
/*****************************************************/
/* lookup - a function to lookup operators and parentheses
            and return the token */
int  lookup(char  ch) {
  switch  (ch) {
    case  '(':
      addChar();
      nextToken = LEFT_PAREN;
      break;
    case  ')':
      addChar();
      nextToken = RIGHT_PAREN;
      break;
    case  '+':
      addChar();
      nextToken = ADD_OP;
      break;
    case  '-':
      addChar();
      nextToken = SUB_OP;
      break;
    case  '*':
      addChar();
      nextToken = MULT_OP;
      break;
    case  '/':
      addChar();
      nextToken = DIV_OP;
      break;
    default:
      addChar();
      nextToken = EOF;
      break;
  }
  return  nextToken;
}
```

```
/*****************************************************/
/* addChar - a function to add nextChar to lexeme */
void  addChar() {
  if  (lexLen <= 98) {
    lexeme[lexLen++] = nextChar;
    lexeme[lexLen] = 0;
  }
  else
    printf("Error - lexeme is too long \n");
}
/*****************************************************/
/* getChar - a function to get the next character of
             input and determine its character class */
 void getChar() {
   if  ((nextChar = getc(in_fp)) = EOF) {
     if  (isalpha(nextChar))
      charClass = LETTER;
     else if  (isdigit(nextChar))
           charClass = DIGIT;
          else  charClass = UNKNOWN;
   }
   else
     charClass = EOF;
}
/*****************************************************/
/* getNonBlank - a function to call getChar until it
                 returns a non-whitespace character */
void getNonBlank() {
  while  (isspace(nextChar))
    getChar();
}
/
*****************************************************/
/* lex - a simple lexical analyzer for arithmetic
         expressions */
int  lex() {
  lexLen = 0;
  getNonBlank();
  switch  (charClass) {
/* Parse identifiers */
    case  LETTER:
      addChar();
      getChar();
      while  (charClass == LETTER || charClass == DIGIT) {
        addChar();
        getChar();
      }
```

```
    nextToken = IDENT;
    break;
/* Parse integer literals */
    case  DIGIT:
      addChar();
      getChar();
      while  (charClass == DIGIT) {
       addChar();
       getChar();
   }
   nextToken = INT_LIT;
   break;
/* Parentheses and operators */
    case  UNKNOWN:
      lookup(nextChar);
      getChar();
      break;
/* EOF */
    case  EOF:
      nextToken = EOF;
      lexeme[0] = 'E';
      lexeme[1] = 'O';
      lexeme[2] = 'F';
      lexeme[3] = 0;
      break;
 } /* End of switch */
 printf("Next token is: %d, Next lexeme is %s\n",
         nextToken, lexeme);
  return  nextToken;
} /* End of function lex */
```

This code illustrates the relative simplicity of lexical analyzers. Of course, we have left out input buffering, as well as some other important details. - Furthermore, we have dealt with a very small and simple input language.

Consider the following expression:

```
(sum + 47) / total
```

Following is the output of the lexical analyzer of `front.c` when used on this expression:

```
Next token is: 25 Next lexeme is (
Next token is: 11 Next lexeme is sum
Next token is: 21 Next lexeme is +
```

```
Next token is: 10 Next lexeme is 47
Next token is: 26 Next lexeme is )
Next token is: 24 Next lexeme is /
Next token is: 11 Next lexeme is total
Next token is: -1 Next lexeme is EOF
```

Names and reserved words in programs have similar patterns. Although it is possible to build a state diagram to recognize every specific reserved word of a programming language, that would result in a prohibitively large state diagram. It is much simpler and faster to have the lexical analyzer recognize names and reserved words with the same pattern and use a lookup in a table of reserved words to determine which names are reserved words. Using this approach considers reserved words to be exceptions in the names token category.

A lexical analyzer often is responsible for the initial construction of the symbol table, which acts as a database of names for the compiler. The entries in the symbol table store information about user-defined names, as well as the attributes of the names. For example, if the name is that of a variable, the variable’s type is one of its attributes that will be stored in the symbol table. Names are usually placed in the symbol table by the lexical analyzer. The attributes of a name are usually put in the symbol table by some part of the compiler that is subsequent to the actions of the lexical analyzer.

4.3 The Parsing Problem

The part of the process of analyzing syntax that is referred to as *syntax analysis* is often called **parsing**. We will use these two interchangeably.

This section discusses the general parsing problem and introduces the two main categories of parsing algorithms, top-down and bottom-up, as well as the complexity of the parsing process.

4.3.1 Introduction to Parsing

Parsers for programming languages construct parse trees for given programs. In some cases, the parse tree is only implicitly constructed, meaning that perhaps only a traversal of the tree is generated. But in all cases, the information required to build the parse tree is created during the parse. Both parse trees and derivations include all of the syntactic information needed by a language processor.

There are two distinct goals of syntax analysis: First, the syntax analyzer must check the input program to determine whether it is syntactically correct. When an error is found, the analyzer must produce a diagnostic message and recover. In this case, recovery means it must get back to a normal state and continue its analysis of the input program. This step is required so that the compiler finds as many errors as possible during a single analysis of the input program. If it is not done well, error recovery may create more errors, or at least more error messages. The second goal of syntax analysis is to produce a complete parse tree, or at least trace the structure of the complete parse tree, for syntactically correct input. The parse tree (or its trace) is used as the basis for translation.

Parsers are categorized according to the direction in which they build parse trees. The two broad classes of parsers are **top-down**, in which the tree is built from the root downward to the leaves, and **bottom-up**, in which the parse tree is built from the leaves upward to the root.

In this chapter, we use a small set of notational conventions for grammar symbols and strings to make the discussion less cluttered. For formal languages, they are as follows:

1. Terminal symbols—lowercase letters at the beginning of the alphabet (a, b, . . .)

2. Nonterminal symbols—uppercase letters at the beginning of the alphabet (A, B, . . .)

3. Terminals or nonterminals—uppercase letters at the end of the alphabet (W, X, Y, Z)

4. Strings of terminals—lowercase letters at the end of the alphabet (w, x, y, z)

5. Mixed strings (terminals and/or nonterminals)—lowercase Greek letters (α, β, δ, γ)

For programming languages, terminal symbols are the small-scale syntactic constructs of the language, what we have referred to as lexemes. The nonterminal symbols of programming languages are usually connotative names or abbreviations, surrounded by angle brackets—for example, <while_statement>, <expr>, and <function_def>. The sentences of a language (programs, in the case of a programming language) are strings of terminals. Mixed strings describe right-hand sides (RHSs) of grammar rules and are used in parsing algorithms.

4.3.2 Top-Down Parsers

A top-down parser traces or builds a parse tree in preorder. A preorder traversal of a parse tree begins with the root. Each node is visited before its branches are followed. Branches from a particular node are followed in left-to-right order. This corresponds to a leftmost derivation.

In terms of the derivation, a top-down parser can be described as follows: Given a sentential form that is part of a leftmost derivation, the parser’s task

is to find the next sentential form in that leftmost derivation. The general form of a left sentential form is xAα, whereby our notational conventions x is a string of terminal symbols, A is a nonterminal, and α is a mixed string. Because x contains only terminals, A is the leftmost nonterminal in the sentential form, so it is the one that must be expanded to get the next sentential form in a leftmost derivation. Determining the next sentential form is a matter of choosing the correct grammar rule that has A as its LHS. For example, if the current sentential form is xAα and the A-rules are $A \rightarrow bB$, $A \rightarrow cBb$, and $A \rightarrow a$, a top-down parser must choose among these three rules to get the next sentential form, which could be xbBα, xcBbα, or xaα. This is the parsing decision problem for top-down parsers.

Different top-down parsing algorithms use different information to make parsing decisions. The most common top-down parsers choose the correct RHS for the leftmost nonterminal in the current sentential form by comparing the next token of input with the first symbols that can be generated by the RHSs of those rules. Whichever RHS has that token at the left end of the string it generates is the correct one. So, in the sentential form xAα, the parser would use whatever token would be the first generated by A to determine which A-rule should be used to get the next sentential form. In the example above, the three RHSs of the A-rules all begin with different terminal symbols. The parser can easily choose the correct RHS based on the next token of input, which must be a, b, or c in this example. In general, choosing the correct RHS is not so straightforward, because some of the RHSs of the leftmost nonterminal in the current sentential form may begin with a nonterminal.

The most common top-down parsing algorithms are closely related. A **recursive-descent parser** is a coded version of a syntax analyzer based directly on the BNF description of the syntax of language. The most common alternative to recursive descent is to use a parsing table, rather than code, to implement the BNF rules. Both of these, which are called **LL algorithms**, are equally powerful, meaning they work on the same subset of all context-free grammars. The first L in LL specifies a left-to-right scan of the input; the second L specifies that a leftmost derivation is generated. Section 4.4 introduces the recursive-descent approach to implementing an LL parser.

4.3.3 Bottom-Up Parsers

A bottom-up parser constructs a parse tree by beginning at the leaves and progressing toward the root. This parse order corresponds to the reverse of a rightmost derivation. That is, the sentential forms of the derivation are produced in order of last to first. In terms of the derivation, a bottom-up parser can be described as follows: Given a right sentential form α, the parser must determine what substring of α is the RHS of the rule in the grammar that must be reduced to its LHS to produce the previous sentential form in the rightmost derivation. For example, the first step for a bottom-up parser is to determine which substring of the initial given sentence is the RHS to be reduced to its corresponding LHS to get the second last sentential form in the derivation. The process of finding the correct RHS to reduce is complicated by the fact that a given right sentential form may include more than one RHS from the grammar of the language being parsed. The correct RHS is called the **handle**. A right sentential form is a sentential form that appears in a rightmost derivation.

Consider the following grammar and derivation:

- S → aAc
- A → aA | b
- S => aAc => aaAc => aabc

A bottom-up parser of this sentence, aabc, starts with the sentence and must find the handle in it. In this example, this is an easy task, for the string contains only one RHS, b. When the parser replaces b with its LHS, A, it gets the second to last sentential form in the derivation, aaAc. In the general case, as stated previously, finding the handle is much more difficult, because a sentential form may include several different RHSs.

A bottom-up parser finds the handle of a given right sentential form by examining the symbols on one or both sides of a possible handle. Symbols to the right of the possible handle are usually tokens in the input that have not yet been analyzed.

The most common bottom-up parsing algorithms are in the LR family, where the L specifies a left-to-right scan of the input and the R specifies that a rightmost derivation is generated.

4.3.4 The Complexity of Parsing

Parsing algorithms that work for any unambiguous grammar are complicated and inefficient. In fact, the complexity of such algorithms is O(n3), which means the amount of time they take is on the order of the cube of the length of the string to be parsed. This relatively large amount of time is required because these algorithms frequently must back up and reparse part of the sentence being analyzed. Reparsing is required when the parser has made a mistake in the parsing process. Backing up the parser also requires that part of the parse tree being constructed (or its trace) must be dismantled and rebuilt. O(n3) algorithms are normally not useful for practical processes, such as syntax analysis for a compiler, because they are far too slow. In situations such as this, computer scientists often search for algorithms that are faster, though less general. Generality is traded for efficiency. In terms of parsing, faster algorithms have been found that work for only a subset of the set of all possible grammars. These algorithms are acceptable as long as the subset includes grammars that describe programming languages. (Actually, as discussed in Chapter 3, the whole class of context-free grammars is not adequate to describe all of the syntax of most programming languages.)

All algorithms used for the syntax analyzers of commercial compilers have complexity O(n), which means the time they take is linearly related to the length of the string to be parsed. This is vastly more efficient than O(n3) algorithms.

4.4 Recursive-Descent Parsing

This section introduces the recursive-descent top-down parser implementation process.

4.4.1 The Recursive-Descent Parsing Process

A recursive-descent parser is so named because it consists of a collection of subprograms, many of which are recursive, and it produces a parse tree in top-down order. This recursion is a reflection of the nature of programming languages, which include several different kinds of nested structures. For example, statements are often nested in other statements. Also, parentheses in expressions must be properly nested. The syntax of these structures is naturally described with recursive grammar rules.

EBNF is ideally suited for recursive-descent parsers. Recall from Chapter 3 that the primary EBNF extensions are braces, which specify that what they enclose can appear zero or more times, and brackets, which specify that what they enclose can appear once or not at all. Note that in both cases, the enclosed symbols are optional. Consider the following examples:

- <if_statement> → **`if`** <logic_expr> <statement> [**`else`** <statement>]
- <ident_list> → ident {, ident}

In the first rule, the **`else`** clause of an **`if`** statement is optional. In the second, an <ident_list> is an identifier, followed by zero or more repetitions of a comma and an identifier.

A recursive-descent parser has a subprogram for each nonterminal in its associated grammar. The responsibility of the subprogram associated with a particular nonterminal is as follows: When given an input string, it traces out

the parse tree that can be rooted at that nonterminal and whose leaves match the input string. In effect, a recursive-descent parsing subprogram is a parser for the language (set of strings) that is generated by its associated nonterminal.

Consider the following EBNF description of simple arithmetic expressions:

- <expr> → <term> {(+ | -) <term>}
- <term> → <factor> {(* | /) <factor>}
- <factor> → id | int_constant | (<expr>)

Recall from Chapter 3 that an EBNF grammar for arithmetic expressions, such as this one, does not force any associativity rule. Therefore, when using such a grammar as the basis for a compiler, one must take care to ensure that the code generation process, which is normally driven by syntax analysis, produces code that adheres to the associativity rules of the language. This can be done easily when recursive-descent parsing is used.

In the following recursive-descent function, `expr`, the lexical analyzer is the function that is implemented in Section 4.2. It gets the next lexeme and puts its token code in the global variable `nextToken`. The token codes are defined as named constants, as in Section 4.2.

A recursive-descent subprogram for a rule with a single RHS is relatively simple. For each terminal symbol in the RHS, that terminal symbol is compared with `nextToken`. If they do not match, it is a syntax error. If they match, the lexical analyzer is called to get the next input token. For each nonterminal, the parsing subprogram for that nonterminal is called.

The recursive-descent subprogram for the first rule in the previous example grammar, written in C, is

```
/* expr
   Parses strings in the language generated by the rule:
   <expr> -> <term> {(+ | -) <term>}
   */
void  expr() {
  printf("Enter <expr>\n");
```

```
/* Parse the first term */
  term();
/* As long as the next token is + or -, get
   the next token and parse the next term */
  while  (nextToken == ADD_OP || nextToken == SUB_OP) {
    lex();
    term();
  }
  printf("Exit <expr>\n");
}  /* End of function expr */
```

Notice that the `expr` function includes tracing output statements, which are included to produce the example output shown later in this section.

Recursive-descent parsing subprograms are written with the convention that each one leaves the next token of input in `nextToken`. So, whenever a parsing function begins, it assumes that `nextToken` has the code for the leftmost token of the input that has not yet been used in the parsing process.

The part of the language that the `expr` function parses consists of one or more terms, separated by either plus or minus operators. This is the language generated by the nonterminal <expr>. Therefore, first it calls the function that parses terms (`term`). Then it continues to call that function as long as it finds `ADD_OP` or `SUB_OP` tokens (which it passes over by calling `lex`). This recursive-descent function is simpler than most, because its associated rule has only one RHS. Furthermore, it does not include any code for syntax error detection or recovery, because there are no detectable errors associated with the grammar rule.

A recursive-descent parsing subprogram for a nonterminal whose rule has more than one RHS begins with code to determine which RHS is to be parsed. Each RHS is examined (at compiler construction time) to determine the set of terminal symbols that can appear at the beginning of sentences it can generate. By matching these sets against the next token of input, the parser can choose the correct RHS.

The parsing subprogram for <term> is similar to that for <expr>:

```
/* term
   Parses strings in the language generated by the rule:
   <term> -> <factor> {(* | /) <factor>)
```

```
   */
void  term() {
  printf("Enter <term>\n");
/* Parse the first factor */
  factor();
/* As long as the next token is * or /, get the
   next token and parse the next factor */
  while  (nextToken == MULT_OP || nextToken == DIV_OP) {
    lex();
    factor();
  }
  printf("Exit <term>\n");
}  /* End of function term */
```

The function for the <factor> nonterminal of our arithmetic expression grammar must choose between its two RHSs. It also includes error detection. In the function for <factor>, the reaction to detecting a syntax error is simply to call the `error` function. In a real parser, a diagnostic message must be produced when an error is detected. Furthermore, parsers must recover from the error so that the parsing process can continue.

```
/* factor
   Parses strings in the language generated by the rule:
   <factor> -> id | int_constant | ( <expr )
   */
void factor() {
  printf("Enter <factor>\n");
/* Determine which RHS */
  if (nextToken == IDENT || nextToken == INT_LIT)
/* Get the next token */
    lex();
/* If the RHS is ( <expr> ), call lex to pass over the
   left parenthesis, call expr, and check for the right
   parenthesis */
  else  {
    if  (nextToken == LEFT_PAREN) {
      lex();
      expr();
      if (nextToken == RIGHT_PAREN)
        lex();
      else
        error();
    }  /* End of if (nextToken == ... */
/* It was not an id, an integer literal, or a left
   parenthesis */
    else error();
```

```
  } /* End of else */
  printf("Exit <factor>\n");;
} /* End of function factor */
```

Following is the trace of the parse of the example expression `(sum + 47) / total`, using the parsing functions `expr`, `term`, and `factor`, and the function `lex` from Section 4.2. Note that the parse begins by calling `lex` and the start symbol routine, in this case, `expr`.

```
Next token is: 25 Next lexeme is (
Enter <expr>
Enter <term>
Enter <factor>
Next token is: 11 Next lexeme is sum
Enter <expr>
Enter <term>
Enter <factor>
Next token is: 21 Next lexeme is +
Exit <factor>
Exit <term>
Next token is: 10 Next lexeme is 47
Enter <term>
Enter <factor>
Next token is: 26 Next lexeme is )
Exit <factor>
Exit <term>
Exit <expr>
Next token is: 24 Next lexeme is /
Exit <factor>
Next token is: 11 Next lexeme is total
Enter <factor>
Next token is: -1 Next lexeme is EOF
Exit <factor>
Exit <term>
Exit <expr>
```

The parse tree traced by the parser for the preceding expression is shown in Figure 4.2.

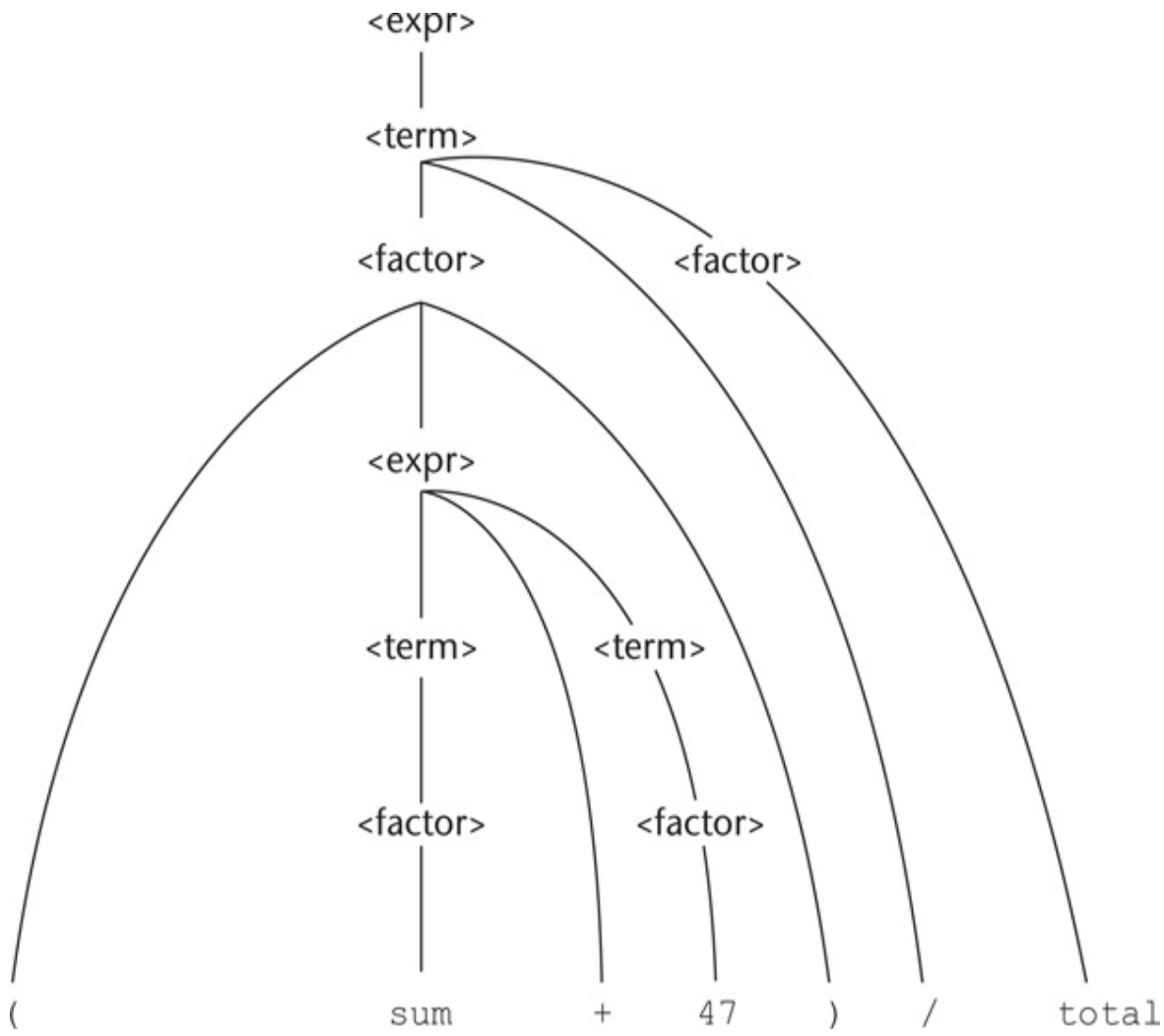


Figure 4.2 Parse tree for (sum + 47)/ total

Figure 4.2 Full Alternative Text

One more example grammar rule and parsing function should help solidify the reader's understanding of recursive-descent parsing. Following is a grammatical description of the Java **`if`** statement:

<ifstmt> → **`if`** (<boolexpr>) <statement> [**`else`** <statement>]

The recursive-descent subprogram for this rule follows:

```
/* Function ifstmt
    Parses strings in the language generated by the rule:
    <ifstmt> -> if (<boolexpr>)  <statement>
                [else <statement>]
    */
void  ifstmt() {
/* Be sure the first token is 'if' */
  if  (nextToken = IF_CODE)
    error();
  else  {
/* Call lex to get to the next token */
    lex();
/* Check for the left parenthesis */
     if  (nextToken = LEFT_PAREN)
       error();
     else  {
/* Parse the Boolean expression */
      boolexpr();
/* Check for the right parenthesis */
     if (nextToken = RIGHT_PAREN)
       error();
     else  {
/* Parse the then clause */
        statement();
/* If an else is next, parse the else clause */
        if  (nextToken == ELSE_CODE) {
/* Call lex to get over the else */
         lex();
         statement();
       } /* end of if (nextToken == ELSE_CODE ... */
     } /* end of else of if (nextToken != RIGHT ... */
    } /* end of else of if (nextToken != LEFT ... */
  } /* end of else of if (nextToken != IF_CODE ... */
} /* end of ifstmt */
```

Notice that this function uses parser functions for statements and Boolean expressions that are not given in this section.

The objective of these examples is to convince you that a recursive-descent parser can be easily written if an appropriate grammar is available for the language. The characteristics of a grammar that allows a recursive-descent parser to be built are discussed in the following subsection.

4.4.2 The LL Grammar Class

Before choosing to use recursive descent as a parsing strategy for a compiler or other program analysis tool, one must consider the limitations of the approach, in terms of grammar restrictions. This section discusses these restrictions and their possible solutions.

One simple grammar characteristic that causes a catastrophic problem for LL parsers is left recursion. For example, consider the following rule:

- $A \rightarrow A+B$

A recursive-descent parser subprogram for A immediately calls itself to parse the first symbol in its RHS. That activation of the A parser subprogram then immediately calls itself again, and again, and so forth. It is easy to see that this leads nowhere (except to stack overflow).

The left recursion in the rule $A \rightarrow A+B$ is called **direct left recursion**, because it occurs in one rule. Direct left recursion can be eliminated from a grammar by the following process:

For each nonterminal, A,

1. Group the A-rules as $A \rightarrow A\alpha 1, | \cdots | A\alpha m | \beta 1 | \beta 2 | \cdots | \beta n$ where none of the β's begins with A

2. Replace the original A-rules with

   $A \rightarrow \beta 1A' | \beta 2A' | \cdots | \beta nA'$ $A' \rightarrow \alpha 1A' | \alpha 2A' | \alpha mA' | \varepsilon$

Note that ε specifies the empty string. A rule that has ε as its RHS is called an *erasure rule*, because its use in a derivation effectively erases its LHS from the sentential form.

Consider the following example grammar and the application of the above process:

$E \rightarrow E+T \mid T \; T \rightarrow T * F \mid F \; F \rightarrow (E)+id$

For the E-rules, we have $\alpha 1 = \; + T$ and $\beta = T$, so we replace the E-rules with

$E \rightarrow T \; E' \; E' \rightarrow +T \; E' \mid \varepsilon$

For the T-rules, we have $\alpha 1 = *F$ and $\beta = F$, so we replace the T-rules with

$T \rightarrow F \; T' \; T \; ' \rightarrow *F \; T' \mid \varepsilon$

Because there is no left recursion in the F-rules, they remain the same, so the complete replacement grammar is

$E \rightarrow T \; E' \; E' \rightarrow +T \; E' \mid \varepsilon \; T \rightarrow F \; T' \; T' \rightarrow \; * \; F \; T' \mid \varepsilon \; F \rightarrow (E) \mid id$

This grammar generates the same language as the original grammar but is not left recursive.

As was the case with the expression grammar written using EBNF in Section 4.4.1, this grammar does not specify left associativity of operators. However, it is relatively easy to design the code generation based on this grammar so that the addition and multiplication operators will have left associativity.

Indirect left recursion poses the same problem as direct left recursion. For example, suppose we have

$A \; \rightarrow \; B \; a \; A \; B \; \rightarrow \; A \; b$

A recursive-descent parser for these rules would have the A subprogram immediately call the subprogram for B, which immediately calls the A subprogram. So, the problem is the same as for direct left recursion. The problem of left recursion is not confined to the recursive-descent approach to building top-down parsers. It is a problem for all top-down parsing algorithms. Fortunately, left recursion is not a problem for bottom-up parsing algorithms.

There is an algorithm to modify a given grammar to remove indirect left recursion (Aho et al., 2006), but it is not covered here. When writing a - grammar for a programming language, one can usually avoid including left

recursion, both direct and indirect.

Left recursion is not the only grammar trait that disallows top-down parsing. Another is whether the parser can always choose the correct RHS on the basis of the next token of input, using only the first token generated by the leftmost nonterminal in the current sentential form. There is a relatively simple test of a non-left recursive grammar that indicates whether this can be done, called the **pairwise disjointness test**. This test requires the ability to compute a set based on the RHSs of a given nonterminal symbol in a grammar. These sets, which are called FIRST, are defined as

- FIRST (α) = {a | α =>* aβ} (IF α =>* ε, ε is in FIRST (α))

in which =>* means 0 or more derivation steps.

An algorithm to compute FIRST for any mixed string α can be found in Aho et al. (2006). For our purposes, FIRST can usually be computed by inspection of the grammar.

The pairwise disjointness test is as follows:

For each nonterminal, A, in the grammar that has more than one RHS, for each pair of rules, $A \rightarrow \alpha i$ and $A \rightarrow \alpha j$, it must be true that

$FIRST(\alpha i) \cap FIRST(\alpha j) = \phi$

(The intersection of the two sets, FIRST(αi) and FIRST(αj), must be empty.)

In other words, if a nonterminal A has more than one RHS, the first terminal symbol that can be generated in a derivation for each of them must be unique to that RHS. Consider the following rules:

$A \rightarrow aB \mid bAb \mid Bb$ $B \rightarrow cB \mid d$

The FIRST sets for the RHSs of the A-rules are {a}, {b}, and {c}, {d}, which are clearly disjoint. Therefore, these rules pass the pairwise disjointness test. What this means, in terms of a recursive-descent parser, is that the code of the subprogram for parsing the nonterminal A can choose which RHS it is dealing with by seeing only the first terminal symbol of input (token) that is

generated by the nonterminal. Now consider the rules

A → aB | BAb B → aA | b

The FIRST sets for the RHSs in the A-rules are {a} and {a}, {b} which are clearly not disjoint. So, these rules fail the pairwise disjointness test. In terms of the parser, the subprogram for A could not determine which RHS was being parsed by looking at the next symbol of input, because if it were an a, it could be either RHS. This issue is of course more complex if one or more of the RHSs begin with nonterminals.

In many cases, a grammar that fails the pairwise disjointness test can be modified so that it will pass the test. For example, consider the rule

<variable> → identifier | identifier [<expression>]

This states that a <variable> is either an identifier or an identifier followed by an expression in brackets (a subscript). These rules clearly do not pass the pairwise disjointness test, because both RHSs begin with the same terminal, identifier. This problem can be alleviated through a process called **left factoring**.

We now take an informal look at left factoring. Consider our rules for <variable>. Both RHSs begin with identifier. The parts that follow identifier in the two RHSs are ε (the empty string) and [<expression>]. The two rules can be replaced by the following two rules:

<variable> → identifier <new>

<new> → ε | [<expression>]

It is not difficult to see that together, these two rules generate the same language as the two rules with which we began. However, these two pass the pairwise disjointness test.

If the grammar is being used as the basis for a recursive-descent parser, an alternative to left factoring is available. With an EBNF extension, the problem disappears in a way that is very similar to the left factoring solution.

Consider the original rules above for <variable>. The subscript can be made optional by placing it in square brackets, as in

<variable> → identifier [[<expression>]]

In this rule, the outer brackets are metasymbols that indicate that what is inside is optional. The inner brackets are terminal symbols of the programming language being described. The point is that we replaced two rules with a single rule that generates the same language but passes the pairwise disjointness test.

A formal algorithm for left factoring can be found in Aho et al. (2006). Left factoring cannot solve all pairwise disjointness problems of grammars. In some cases, rules must be rewritten in other ways to eliminate the problem.

4.5 Bottom-Up Parsing

This section introduces the general process of bottom-up parsing and includes a description of the LR parsing algorithm.

4.5.1 The Parsing Problem for Bottom-Up Parsers

Consider the following grammar for arithmetic expressions:

- E → E + T | T
- T → T * F | F
- F → (E) | id

Notice that this grammar generates the same arithmetic expressions as the example in Section 4.4. The difference is that this grammar is left recursive, which is acceptable to bottom-up parsers. Also note that grammars for bottom-up parsers normally do not include metasymbols such as those used to specify extensions to BNF. The following rightmost derivation illustrates this grammar:

- E => E+T
- => E + T * F
- => E + T * id
- => E + F * id
- => E + id * id

- => T + id * id
- => F + id * id
- => id + id * id

The underlined part of each sentential form in this derivation is the RHS that is rewritten as its corresponding LHS to get the previous sentential form. The process of bottom-up parsing produces the reverse of a rightmost derivation. So, in the example derivation, a bottom-up parser starts with the last sentential form (the input sentence) and produces the sequence of sentential forms from there until all that remains is the start symbol, which in this grammar is E. In each step, the task of the bottom-up parser is to find the specific RHS, the handle, in the sentential form that must be rewritten to get the next (previous) sentential form. As mentioned earlier, a right sentential form may include more than one RHS. For example, the right sentential form

- E + T * id

includes three RHSs, E+T, T and id. Only one of these is the handle. For example, if the RHS E+T were chosen to be rewritten in this sentential form, the resulting sentential form would be E * id, but E * id is not a legal right sentential form for the given grammar.

The handle of a right sentential form is unique. The task of a bottom-up parser is to find the handle of any given right sentential form that can be generated by its associated grammar. Formally, handle is defined as follows:

- Definition: β is the **handle** of the right sentential form γ=αβw if and only if S => *rm αAw =>rmαβw

In this definition, =>rm specifies a rightmost derivation step, and => *rm specifies zero or more rightmost derivation steps. Although the definition of a handle is mathematically concise, it provides little help in finding the handle of a given right sentential form. In the following, we provide the definitions of several substrings of sentential forms that are related to handles. The purpose of these is to provide some intuition about handles.

- Definition: β is a **phrase** of the right sentential form γ if and only if S =>* γ=α1Aα2 =>+α1βα2

In this definition, =>+ means one or more derivation steps.

- Definition: β is a **simple phrase** of the right sentential form γ if and only if S =>* γ=α1Aα2 =>+α1βα2

If these two definitions are compared carefully, it is clear that they differ only in the last derivation specification. The definition of phrase uses one or more steps, while the definition of simple phrase uses exactly one step.

The definitions of phrase and simple phrase may appear to have the same lack of practical value as that of a handle, but that is not true. Consider what a phrase is relative to a parse tree. It is the string of all of the leaves of the partial parse tree that is rooted at one particular internal node of the whole parse tree. A simple phrase is just a phrase that takes a single derivation step from its root nonterminal node. In terms of a parse tree, a phrase can be derived from a single nonterminal in one or more tree levels, but a simple phrase can be derived in just a single tree level. Consider the parse tree shown in Figure 4.3.

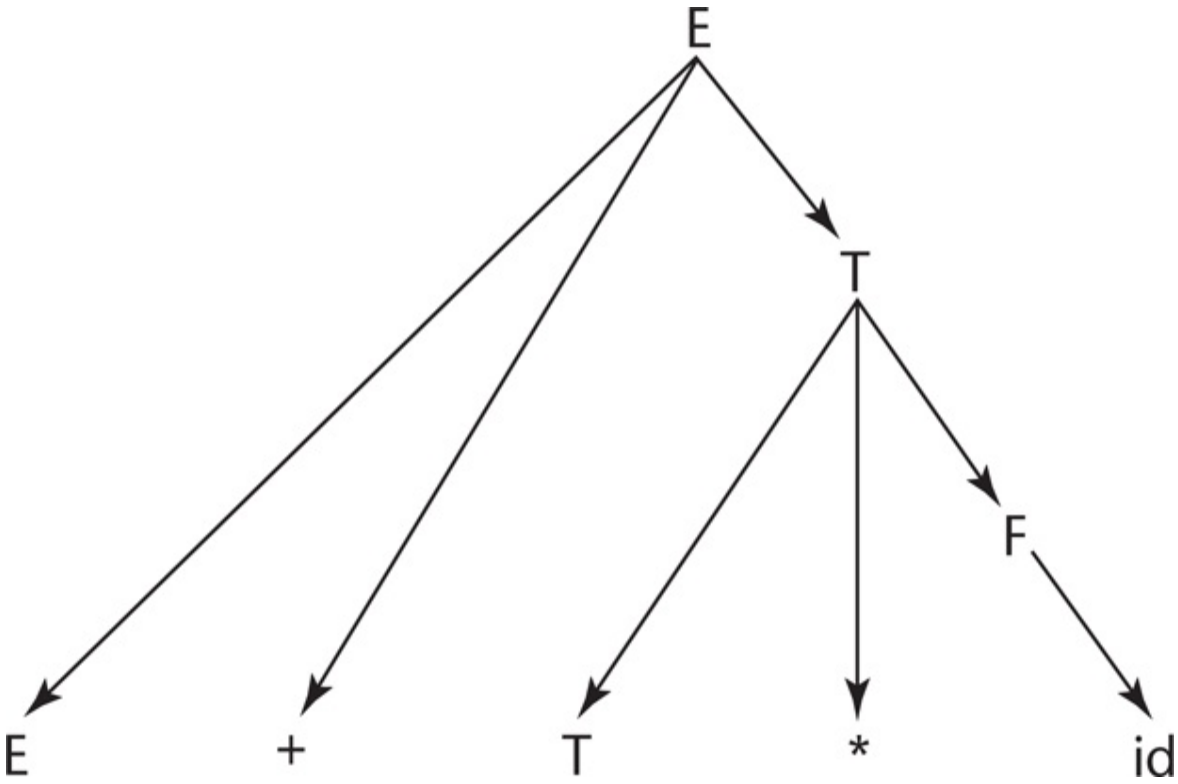


Figure 4.3 A parse tree for E+T * id

The leaves of the parse tree in Figure 4.3 comprise the sentential form E+T * id. Because there are three internal nodes, there are three phrases. Each internal node is the root of a subtree, whose leaves are a phrase. The root node of the whole parse tree, E, generates all of the resulting sentential form, E+T * id, which is a phrase. The internal node, T, generates the leaves T * id, which is another phrase. Finally, the internal node, F, generates id, which is also a phrase. So, the phrases of the sentential form E+T * id are E+T * id, and id. Notice that phrases are not necessarily RHSs in the underlying grammar.

The simple phrases are a subset of the phrases. In the previous example, the only simple phrase is id. A simple phrase is always a RHS in the grammar.

The reason for discussing phrases and simple phrases is this: The handle of

any rightmost sentential form is its leftmost simple phrase. So now we have a highly intuitive way to find the handle of any right sentential form, assuming we have the grammar and can draw a parse tree. This approach to finding handles is of course not practical for a parser. (If you already have a parse tree, why do you need a parser?) Its only purpose is to provide the reader with some intuitive feel for what a handle is, relative to a parse tree, which is easier than trying to think about handles in terms of sentential forms.

We can now consider bottom-up parsing in terms of parse trees, although the purpose of a parser is to produce a parse tree. Given the parse tree for an entire sentence, you easily can find the handle, which is the first thing to rewrite in the sentence to get the previous sentential form. Then the handle can be pruned from the parse tree and the process repeated. Continuing to the root of the parse tree, the entire rightmost derivation can be constructed.

4.5.2 Shift-Reduce Algorithms

Bottom-up parsers are often called **shift-reduce algorithms**, because shift and reduce are the two most common actions they specify. An integral part of every bottom-up parser is a stack. As with other parsers, the input to a bottom-up parser is the stream of tokens of a program and the output is a sequence of grammar rules. The shift action moves the next input token onto the parser's stack. A reduce action replaces an RHS (the handle) on top of the parser's stack by its corresponding LHS. Every parser for a programming language is a **pushdown automaton** (**PDA**), because a PDA is a recognizer for a context-free language. You need not be intimate with PDAs to understand how a bottom-up parser works, although it helps. A PDA is a very simple mathematical machine that scans strings of symbols from left to right. A PDA is so named because it uses a pushdown stack as its memory. PDAs can be used as recognizers for context-free languages. Given a string of symbols over the alphabet of a context-free language, a PDA that is designed for the purpose can determine whether the string is or is not a sentence in the language. In the process, the PDA can produce the information needed to construct a parse tree for the sentence.

With a PDA, the input string is examined, one symbol at a time, left to right.

The input is treated very much as if it were stored in another stack, because the PDA never sees more than the leftmost symbol of the input.

Note that a recursive-descent parser is also a PDA. In that case, the stack is that of the run-time system, which records subprogram calls (among other things), which correspond to the nonterminals of the grammar.

4.5.3 LR Parsers

Many different bottom-up parsing algorithms have been devised. Most of them are variations of a process called LR. LR parsers use a relatively small program and a parsing table that is built for a specific programming language. The original LR algorithm was designed by Donald Knuth (Knuth, 1965). This algorithm, which is sometimes called **canonical LR**, was not used in the years immediately following its publication because producing the required parsing table required large amounts of computer time and memory. Subsequently, several variations on the canonical LR table construction process were developed (DeRemer, 1971; DeRemer and Pennello, 1982). These are characterized by two properties: (1) They require far less computer resources to produce the required parsing table than the canonical LR algorithm, and (2) they work on smaller classes of grammars than the canonical LR algorithm.

There are three advantages to LR parsers:

1. They can be built for all programming languages.

2. They can detect syntax errors as soon as it is possible in a left-to-right scan.

3. The LR class of grammars is a proper superset of the class parsable by LL parsers (for example, many left recursive grammars are LR, but none are LL).

The only disadvantage of LR parsing is that it is difficult to produce by hand the parsing table for a given grammar for a complete programming language.

This is not a serious disadvantage, however, for there are several programs available that take a grammar as input and produce the parsing table, as - discussed later in this section.

Prior to the appearance of the LR parsing algorithm, there were a number of parsing algorithms that found handles of right sentential forms by looking both to the left and to the right of the substring of the sentential form that was suspected of being the handle. Knuth's insight was that one could effectively look to the left of the suspected handle all the way to the bottom of the parse stack to determine whether it was the handle. But all of the information in the parse stack that was relevant to the parsing process could be represented by a single state, which could be stored on the top of the stack. In other words, Knuth discovered that regardless of the length of the input string, the length of the sentential form, or the depth of the parse stack, there were only a relatively small number of different situations, as far as the parsing process is concerned. Each situation could be represented by a state and stored in the parse stack, one state symbol for each grammar symbol on the stack. At the top of the stack would always be a state symbol, which represented the relevant information from the entire history of the parse, up to the current time. We will use subscripted uppercase S's to represent the parser states.

Figure 4.4 shows the structure of an LR parser. The contents of the parse stack for an LR parser have the following form:

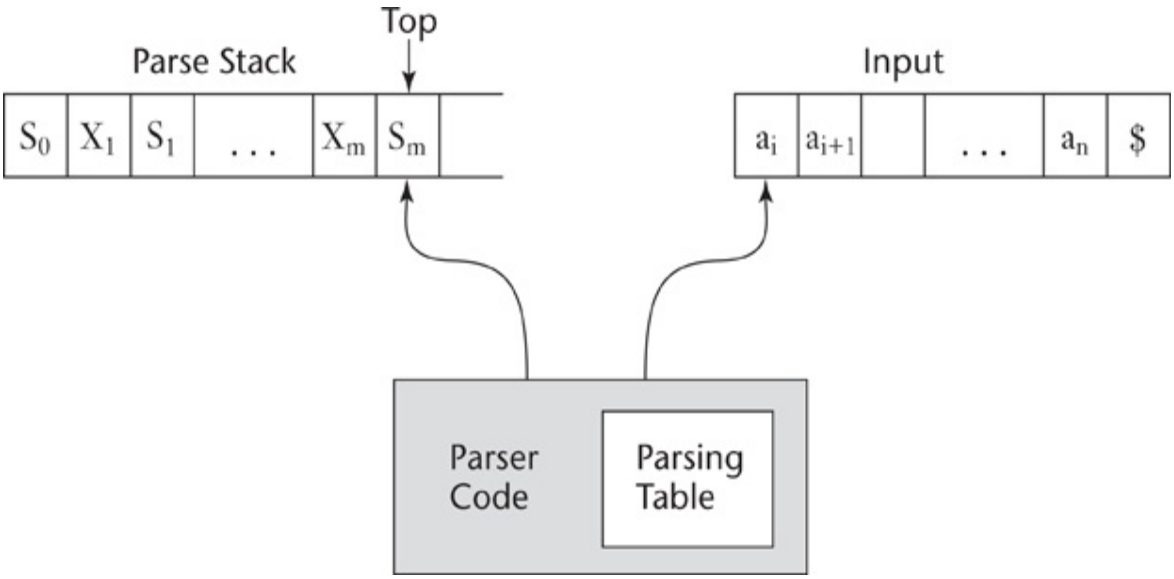


Figure 4.4 The structure of an LR parser

Figure 4.4 Full Alternative Text

S0X1S1X2…XmSm (top)

where the S's are state symbols and the X's are grammar symbols. An LR parser configuration is a pair of strings (stack, input), with the detailed form

(S0X1S1X2S2…XmSm, aiAi+1…an$)

Notice that the input string has a dollar sign at its right end. This sign is put there during initialization of the parser. It is used for normal termination of the parser. Using this parser configuration, we can formally define the LR parser process, which is based on the parsing table.

An LR parsing table has two parts, named ACTION and GOTO. The ACTION part of the table specifies most of what the parser does. It has state symbols as its row labels and the terminal symbols of the grammar as its column labels. Given a current parser state, which is represented by the state symbol on top of the parse stack, and the next symbol (token) of input, the parse table specifies what the parser should do. The two primary parser actions are shift and reduce. Either the parser shifts the next input symbol onto the parse stack, along with a state symbol, or it already has the handle on top of the stack, which it reduces to the LHS of the rule whose RHS is the same as the handle. Two other actions are possible: accept, which means the parser has successfully completed the parse of the input, and error, which means the parser has detected a syntax error.

The rows of the GOTO part of the LR parsing table have state symbols as labels. This part of the table has nonterminals as column labels. The values in the GOTO part of the table indicate which state symbol should be pushed onto the parse stack after a reduction has been completed, which means the handle has been removed from the parse stack and the new nonterminal has been pushed onto the parse stack. The specific symbol is found at the row

whose label is the state symbol on top of the parse stack after the handle and its associated state symbols have been removed. The column of the GOTO table that is used is the one with the label, that is the LHS of the rule used in the reduction.

Consider the traditional grammar for arithmetic expressions that follows:

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

The rules of this grammar are numbered to provide a simple way to reference them in a parsing table.

Figure 4.5 shows the LR parsing table for this grammar. Abbreviations are used for the actions: R for reduce and S for shift. R4 means reduce using rule 4; S6 means shift the next symbol of input onto the stack and push state 6 onto the stack. Empty positions in the ACTION table indicate syntax errors. In a complete parser, these could have calls to error-handling routines.

	Action						Goto		
State	id	+	*	(	)	$	E	T	F
0	S5			S4			1	2	3
1		S6				accept			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

Figure 4.5 The LR parsing table for an arithmetic expression grammar

Figure 4.5 Full Alternative Text

LR parsing tables can easily be constructed using a software tool, such as yacc (Johnson, 1975), which takes the grammar as input. Although LR parsing tables can be produced by hand, for a grammar of a real programming language, the task would be lengthy, tedious, and error prone. For real compilers, LR parsing tables are always generated with software

tools.

The initial configuration of an LR parser is

(S0, a1…an$)

The parser actions are informally defined as follows:

1. The Shift process is simple: The next symbol of input is pushed onto the stack, along with the state symbol that is part of the Shift specification in the ACTION table.

2. For a Reduce action, the handle must be removed from the stack. Because for every grammar symbol on the stack there is a state symbol, the number of symbols removed from the stack is twice the number of symbols in the handle. After removing the handle and its associated state symbols, the LHS of the rule is pushed onto the stack. Finally, the GOTO table is used, with the row label being the symbol that was exposed when the handle and its state symbols were removed from the stack, and the column label being the nonterminal that is the LHS of the rule used in the reduction.

3. When the action is Accept, the parse is complete and no errors were found.

4. When the action is Error, the parser calls an error-handling routine.

Although there are many parsing algorithms based on the LR concept, they differ only in the construction of the parsing table. All LR parsers use this same parsing algorithm.

Perhaps the best way to become familiar with the LR parsing process is through an example. Initially, the parse stack has the single symbol 0, which represents state 0 of the parser. The input contains the input string with an end marker, in this case a dollar sign, attached to its right end. At each step, the parser actions are dictated by the top (rightmost in Figure 4.4) symbol of the parse stack and the next (leftmost in Figure 4.4) token of input. The correct action is chosen from the corresponding cell of the ACTION part of

the parse table. The GOTO part of the parse table is used after a reduction action. Recall that GOTO is used to determine which state symbol is placed on the parse stack after a reduction.

Following is a trace of a parse of the string id + id * id, using the LR parsing algorithm and the parsing table shown in Figure 4.5.

Stack	*Input*	*Action*
0	id + id * id $	Shift 5
0id5	+ id * id $	Reduce 6 (use GOTO[0, F])
0F3	+ id * id $	Reduce 4 (use GOTO[0, T])
0T2	+ id * id $	Reduce 2 (use GOTO[0, E])
0E1	+ id * id $	Shift 6
0E1+6	id * id $	Shift 5
0E1+6id5	* id $	Reduce 6 (use GOTO[6, F])
0E1+6F3	* id $	Reduce 4 (use GOTO[6, T])
0E1+6T9	* id $	Shift 7
0E1+6T9*7	id $	Shift 5
0E1+6T9*7id5	$	Reduce 6 (use GOTO[7, F])
0E1+6T9*7F10	$	Reduce 3 (use GOTO[6, T])
0E1+6T9	$	Reduce 1 (use GOTO[0, E])
0E1	$	Accept

The algorithms to generate LR parsing tables from given grammars, which are described in Aho et al. (2006), are not overly complex but are beyond the scope of a book on programming languages. As stated previously, there are a number of different software systems available to generate LR parsing tables.

SUMMARY

Syntax analysis is a common part of language implementation, regardless of the implementation approach used. Syntax analysis is normally based on a formal syntax description of the language being implemented. A context-free grammar, which is also called BNF, is the most common approach for describing syntax. The task of syntax analysis is usually divided into two parts: lexical analysis and syntax analysis. There are several reasons for separating lexical analysis—namely, simplicity, efficiency, and portability.

A lexical analyzer is a pattern matcher that isolates the small-scale parts of a program, which are called lexemes. Lexemes occur in categories, such as integer literals and names. These categories are called tokens. Each token is assigned a numeric code, which along with the lexeme is what the lexical analyzer produces. There are three distinct approaches to constructing a lexical analyzer: using a software tool to generate a table for a table-driven analyzer, building such a table by hand, and writing code to implement a state diagram description of the tokens of the language being implemented. The state diagram for tokens can be reasonably small if character classes are used for transitions, rather than having transitions for every possible character from every state node. Also, the state diagram can be simplified by using a table lookup to recognize reserved words.

Syntax analyzers have two goals: to detect syntax errors in a given program and to produce a parse tree, or possibly only the information required to build such a tree, for a given program. Syntax analyzers are either top-down, meaning they construct leftmost derivations and a parse tree in top-down order, or bottom-up, in which case they construct the reverse of a rightmost derivation and a parse tree in bottom-up order. Parsers that work for all unambiguous grammars have complexity O(n3). However, parsers used for implementing syntax analyzers for programming languages work on subclasses of unambiguous grammars and have complexity O(n).

A recursive-descent parser is an LL parser that is implemented by writing code directly from the grammar of the source language. EBNF is ideal as the

basis for recursive-descent parsers. A recursive-descent parser has a subprogram for each nonterminal in the grammar. The code for a given grammar rule is simple if the rule has a single RHS. The RHS is examined left to right. For each nonterminal, the code calls the associated subprogram for that nonterminal, which parses whatever the nonterminal generates. For each terminal, the code compares the terminal with the next token of input. If they match, the code simply calls the lexical analyzer to get the next token. If they do not, the subprogram reports a syntax error. If a rule has more than one RHS, the subprogram must first determine which RHS it should parse. It must be possible to make this determination on the basis of the next token of input.

Two distinct grammar characteristics prevent the construction of a recursive-descent parser based on the grammar. One of these is left recursion. The process of eliminating direct left recursion from a grammar is relatively simple. Although we do not cover it, an algorithm exists to remove both direct and indirect left recursion from a grammar. The other problem is detected with the pairwise disjointness test, which tests whether a parsing subprogram can determine which RHS is being parsed on the basis of the next token of input. Some grammars that fail the pairwise disjointness test often can be modified to pass it, using left factoring.

The parsing problem for bottom-up parsers is to find the substring of the current sentential form that must be reduced to its associated LHS to get the next (previous) sentential form in the rightmost derivation. This substring is called the handle of the sentential form. A parse tree can provide an intuitive basis for recognizing a handle. A bottom-up parser is a shift-reduce algorithm, because in most cases it either shifts the next lexeme of input onto the parse stack or reduces the handle that is on top of the stack.

The LR family of shift-reduce parsers is the most commonly used bottom-up parsing approach for programming languages, because parsers in this family have several advantages over alternatives. An LR parser uses a parse stack, which contains grammar symbols and state symbols to maintain the state of the parser. The top symbol on the parse stack is always a state symbol that represents all of the information in the parse stack that is relevant to the parsing process. LR parsers use two parsing tables: ACTION and GOTO.

The ACTION part specifies what the parser should do, given the state symbol on top of the parse stack and the next token of input. The GOTO table is used to determine which state symbol should be placed on the parse stack after a reduction has been done.

REVIEW QUESTIONS

1. What are three reasons why syntax analyzers are based on grammars?
2. Explain the three reasons why lexical analysis is separated from syntax analysis.
3. Define *lexeme* and *token*.
4. What are the primary tasks of a lexical analyzer?
5. Describe briefly the three approaches to building a lexical analyzer.
6. What is a state transition diagram?
7. Why are character classes used, rather than individual characters, for the letter and digit transitions of a state diagram for a lexical analyzer?
8. What are the two distinct goals of syntax analysis?
9. Describe the differences between top-down and bottom-up parsers.
10. Describe the parsing problem for a top-down parser.
11. Describe the parsing problem for a bottom-up parser.
12. Explain why compilers use parsing algorithms that work on only a subset of all grammars.
13. Why are named constants used, rather than numbers, for token codes?
14. Describe how a recursive-descent parsing subprogram is written for a rule with a single RHS.
15. Explain the two grammar characteristics that prohibit them from being used as the basis for a top-down parser.

16. What is the FIRST set for a given grammar and sentential form?
17. Describe the pairwise disjointness test.
18. What is left factoring?
19. What is a phrase of a sentential form?
20. What is a simple phrase of a sentential form?
21. What is the handle of a sentential form?
22. What is the mathematical machine on which both top-down and bottom-up parsers are based?
23. Describe three advantages of LR parsers.
24. What was Knuth’s insight in developing the LR parsing technique?
25. Describe the purpose of the ACTION table of an LR parser.
26. Describe the purpose of the GOTO table of an LR parser.
27. Is left recursion a problem for LR parsers?

PROBLEM SET

1. Perform the pairwise disjointness test for the following grammar rules.

    1. A → aB | b | cBB

    2. B → aB | bA | aBb

    3. C → aaA | b | caB

2. Perform the pairwise disjointness test for the following grammar rules.

    1. S → aSb | bAA

    2. A → B{aB} | a

    3. B → aB | a

3. Show a trace of the recursive descent parser given in Section 4.4.1 for the string `a + b * c`.

4. Show a trace of the recursive descent parser given in Section 4.4.1 for the string `a * (b + c)`.

5. Given the following grammar and the right sentential form, draw a parse tree and show the phrases and simple phrases, as well as the handle.

    S → aAb | bBAA → ab | aABB → aB | b

    1. aaAbb

    2. bBab

    3. aaAbBb

6. Given the following grammar and the right sentential form, draw a parse tree and show the phrases and simple phrases, as well as the handle.

S → AbB | bAcA → Ab | aBBB → Ac | cBb | c

1. aAcccbbc
2. AbcaBccb
3. baBcBbbc

7. Show a complete parse, including the parse stack contents, input string, and action for the string id * (id + id), using the grammar and parse table in Section 4.5.3.

8. Show a complete parse, including the parse stack contents, input string, and action for the string (id + id) * id, using the grammar and parse table in Section 4.5.3.

9. Get the algorithm to remove the indirect left recursion from a grammar from Aho et al. (2006). Use this algorithm to remove all left recursion from the following grammar: S → Aa | BbA → Aa | Abc | c | SbB → bb

PROGRAMMING EXERCISES

1. Design a state diagram to recognize one form of the comments of the C-based programming languages, those that begin with `/*` and end with `*/`.

2. Design a state diagram to recognize the floating-point literals of your favorite programming language.

3. Write and test the code to implement the state diagram of Problem 1.

4. Write and test the code to implement the state diagram of Problem 2.

5. Modify the lexical analyzer given in Section 4.2 to recognize the following list of reserved words and return their respective token codes: **`for`** (`FOR_CODE`, `30`), **`if`** (`IF_CODE`, `31`), **`else`** (`ELSE_CODE`, `32`), **`while`** (`WHILE_CODE`, `33`), **`do`** (`DO_CODE`, `34`), **`int`** (`INT_CODE`, `35`), **`float`** (`FLOAT_CODE`, `36`), **`switch`** (`SWITCH_CODE`, `37`).

6. Convert the lexical analyzer (which is written in C) given in Section 4.2 to Java.

7. Convert the recursive descent parser routines for <expr>, <term>, and <factor> given in Section 4.4.1 to Java.

8. For those rules that pass the test in Problem 1, write a recursive-descent parsing subprogram that parses the language generated by the rules. Assume you have a lexical analyzer named `lex` and an error-handling subprogram named `error`, which is called whenever a syntax error is detected.

9. For those rules that pass the test in Problem 2, write a recursive-descent parsing subprogram that parses the language generated by the rules. Assume you have a lexical analyzer named `lex` and an error-handling subprogram named `error`, which is called whenever a syntax error is detected.

10. Implement and test the LR parsing algorithm given in Section 4.5.3.

11. Write an EBNF rule that describes the **`while`** statement of Java or C++. Write the recursive-descent subprogram in Java or C++ for this rule.

12. Write an EBNF rule that describes the **`for`** statement of Java or C++. Write the recursive-descent subprogram in Java or C++ for this rule.

5 Names, Bindings, and Scopes

This chapter introduces the fundamental semantic issues of variables. It begins by describing the nature of names and special words in programming languages. The attributes of variables, including type, address, and value, are then discussed, including the issue of aliases. The important concepts of binding and binding times are introduced next, including the different possible binding times for variable attributes and how they define four different categories of variables. Following that, two very different scoping rules for names, static and dynamic, are described, along with the concept of a referencing environment of a statement. Finally, named constants and variable initialization are discussed.

5.1 Introduction

Imperative programming languages are, to varying degrees, abstractions of the underlying von Neumann computer architecture. The architecture's two primary components are its memory, which stores both instructions and data, and its processor, which provides operations for modifying the contents of the memory. The abstractions in a language for the memory cells of the machine are variables. In some cases, the characteristics of the abstractions are very close to the characteristics of the cells; an example of this is an integer variable, which is usually represented directly in one or more bytes of memory. In other cases, the abstractions are far removed from the organization of the hardware memory, as with a three-dimensional array, which requires a software mapping function to support the abstraction.

A variable can be characterized by a collection of properties, or attributes, the most important of which is type, a fundamental concept in programming languages. Designing the data types of a language requires that a variety of issues be considered. (Data types are discussed in Chapter 6.) Among the most important of these issues are the scope and lifetime of variables.

Functional programming languages allow expressions to be named. These named expressions appear like assignments to variable names in imperative languages, but are fundamentally different in that they cannot be changed. So, they are like the named constants of the imperative languages. Pure functional languages do not have variables that are like those of the imperative languages. However, many functional languages do include such variables.

In the remainder of this book, families of languages will often be referred to as if they were single languages. For example, Fortran will mean all of the versions of Fortran. This is also the case for Ada. Likewise, a reference to C will mean the original version of C, as well as C89 and C99. When a specific version of a language is named, it is because it is different from the other family members within the topic being discussed. If we add a plus sign (+) to the name of a version of a language, we mean all versions of the language

beginning with the one named. For example, Fortran 95+ means all versions of Fortran beginning with Fortran 95. The phrase **C-based languages** will be used to refer to C, Objective-C, C++, Java, and C#.1

1. We were tempted to include the scripting languages JavaScript and PHP as C-based languages, but decided they were just a bit too different from their ancestors.

5.2 Names

Before beginning our discussion of variables, the design of one of the fundamental attributes of variables, names, must be covered. Names are also associated with subprograms, formal parameters, and other program constructs. The term *identifier* is often used interchangeably with *name*.

5.2.1 Design Issues

The following are the primary design issues for names:

- Are names case sensitive?
- Are the special words of the language reserved words or keywords?

These issues are discussed in the following two subsections, which also include examples of several design choices.

5.2.2 Name Forms

A **name** is a string of characters used to identify some entity in a program.

C99 has no length limitation on its internal names, but only the first 63 are significant. External names in C99 (those defined outside functions, which must be handled by the linker) are restricted to 31 characters. Names in Java and C# have no length limit, and all characters in them are significant. C++ does not specify a length limit on names, although implementors sometimes do.

Names in most programming languages have the same form: a letter followed by a string consisting of letters, digits, and underscore characters (_). Although the use of underscore characters to form names was widely used in

the 1970s and 1980s, that practice is now far less popular. In the C-based languages, it has to a large extent been replaced by the so-called *camel notation*, in which all of the words of a multiple-word name except the first are capitalized, as in `myStack`.[2] Note that the use of underscores and mixed case in names is a programming style issue, not a language design issue.

2. It is called “camel” because words written in it often have embedded uppercase letters, which look like a camel’s humps.

history note

The earliest programming languages used single-character names. This notation was natural because early programming was primarily mathematical, and mathematicians have long used single-character names for unknowns in their formal notations.

Fortran I broke with the tradition of the single-character name, allowing up to six characters in its names.

All variable names in PHP must begin with a dollar sign. In Perl, the special character at the beginning of a variable’s name, `$`, `@`, or `%`, specifies its type (although in a different sense than in other languages). In Ruby, special characters at the beginning of a variable’s name, `@` or `@@`, indicate that the variable is an instance or a class variable, respectively.

In many languages, notably the C-based languages, uppercase and lowercase letters in names are distinct; that is, names in these languages are **case sensitive**. For example, the following three names are distinct in C++: `rose`, `ROSE`, and `Rose`. To some people, this is a serious detriment to readability, because names that look very similar in fact denote different entities. In that sense, case sensitivity violates the design principle that language constructs that look similar should have similar meanings. But in languages whose variable names are case-sensitive, although `Rose` and `rose` look similar, there is no connection between them.

Obviously, not everyone agrees that case sensitivity is bad for names. In C,

the problems of case sensitivity are avoided by the convention that variable names do not include uppercase letters. In Java and C#, however, the problem cannot be escaped because many of the predefined names include both uppercase and lowercase letters. For example, the Java method for converting a string to an integer value is `parseInt`, and spellings such as `ParseInt` and `parseint` are not recognized. This is a problem of writability rather than readability, because the need to remember specific case usage makes it more difficult to write correct programs. It is a kind of intolerance on the part of the language designer, which is enforced by the compiler.

5.2.3 Special Words

Special words in programming languages are used to make programs more readable by naming actions to be performed. They also are used to separate the syntactic parts of statements and programs. In most languages, special words are classified as reserved words, which means they cannot be redefined by programmers. But in some, such as Fortran, they are only keywords, which means they can be redefined.

A **reserved word** is a special word of a programming language that cannot be used as a name. There is one potential problem with reserved words: If the language includes a large number of reserved words, the user may have difficulty making up names that are not reserved. The best example of this is COBOL, which has 300 reserved words. Unfortunately, some of the most commonly chosen names by programmers are in the list of reserved words—for example, `LENGTH`, `BOTTOM`, `DESTINATION`, and `COUNT`.

In program code examples in this book, reserved words are presented in boldface.

In most languages, names that are defined in other program units, such as Java packages and C and C++ libraries, can be made visible to a program. These names are predefined, but visible only if explicitly imported. Once imported, they cannot be redefined.

5.3 Variables

A program variable is an abstraction of a computer memory cell or collection of cells. Programmers often think of variables as names for memory locations, but there is much more to a variable than just a name.

The move from machine languages to assembly languages was largely one of replacing absolute numeric memory addresses for data with names, making programs far more readable and therefore easier to write and maintain. Assembly languages also provided an escape from the problem of manual absolute addressing, because the translator that converted the names to actual addresses also chose those addresses.

A variable can be characterized as a sextuple of attributes: (name, address, value, type, lifetime, and scope). Although this may seem too complicated for such an apparently simple concept, it provides the clearest way to explain the various aspects of variables.

Our discussion of variable attributes will lead to examinations of the important related concepts of aliases, binding, binding times, declarations, scoping rules, and referencing environments.

The name, address, type, and value attributes of variables are discussed in the following subsections. The lifetime and scope attributes are discussed in Sections 5.4.3 and 5.5, respectively.

5.3.1 Name

Variable names are the most common names in programs. They were discussed at length in Section 5.2 in the general context of entity names in programs. Most variables have names. The ones that do not are discussed in Section 5.4.3.3.

5.3.2 Address

The **address** of a variable is the machine memory address with which it is associated. This association is not as simple as it may appear. In many languages, it is possible for the same variable to be associated with different addresses at different times during the execution of the program. For example, if a subprogram has a local variable that is allocated from the run-time stack when the subprogram is called, different calls may result in that variable having different addresses. These are in a sense different instantiations of the same variable.

The process of associating variables with addresses is further discussed in Section 5.4.3. An implementation model for subprograms and their activations is discussed in Chapter 10.

The address of a variable is sometimes called its **l-value**, because the address is what is required when the name of a variable appears in the left side of an assignment.

It is possible to have multiple variables that have the same address. When more than one variable name can be used to access the same memory location, the variables are called **aliases**. Aliasing is a hindrance to readability because it allows a variable to have its value changed by an assignment to a different variable. For example, if variables named `total` and `sum` are aliases, any change to the value of `total` also changes the value of `sum` and vice versa. A reader of the program must always remember that `total` and `sum` are different names for the same memory cell. Because there can be any number of aliases in a program, this can be very difficult in practice. Aliasing also makes program verification more difficult.

Aliases can be created in programs in several different ways. One common way in C and C++ is with their union types. Unions are discussed at length in Chapter 6.

Two pointer variables are aliases when they point to the same memory location. The same is true for reference variables. This kind of aliasing is

simply a side effect of the nature of pointers and references. When a C++ pointer is set to point at a named variable, the pointer, when dereferenced, and the variable's name are aliases.

Aliasing can be created in many languages through subprogram parameters. These kinds of aliases are discussed in Chapter 9.

The time when a variable becomes associated with an address is very important to an understanding of programming languages. This subject is discussed in Section 5.4.3.

5.3.3 Type

The **type** of a variable determines the range of values the variable can store and the set of operations that are defined for values of the type. For example, the `int` type in Java specifies a value range of −2147483648 to 2147483647 and arithmetic operations for addition, subtraction, multiplication, division, and modulus.

5.3.4 Value

The **value** of a variable is the contents of the memory cell or cells associated with the variable. It is convenient to think of computer memory in terms of *abstract* cells, rather than physical cells. The physical cells, or individually addressable units, of most contemporary computer memories are eight-bit units called bytes. A byte size is too small for most program variables. An abstract memory cell has the size required by the variable with which it is associated. For example, although floating-point values may occupy four physical bytes in a particular implementation of a particular language, a floating-point value is thought of as occupying a single abstract memory cell. The value of each simple nonstructured type is considered to occupy a single abstract cell. Henceforth, the term *memory cell* will mean abstract memory cell.

A variable's value is sometimes called its **r-value** because it is what is

required when the name of the variable appears in the right side of an assignment statement. To access the *r*-value, the *l*-value must be determined first. Such determinations are not always simple. For example, scoping rules can greatly complicate matters, as is discussed in Section 5.5.

5.4 The Concept of Binding

A **binding** is an association between an attribute and an entity, such as between a variable and its type or value, or between an operation and a symbol. The time at which a binding takes place is called **binding time**. Binding and binding times are prominent concepts in the semantics of programming languages. Bindings can take place at language design time, language implementation time, compile time, load time, link time, or run time. For example, the asterisk symbol (`*`) is usually bound to the multiplication operation at language design time. A data type, such as **`int`** in C, is bound to a range of possible values at language implementation time. At compile time, a variable in a Java program is bound to a particular data type. A variable may be bound to a storage cell when the program is loaded into memory. That same binding does not happen until run time in some cases, as with variables declared in Java methods. A call to a library subprogram is bound to the subprogram code at link time.

Consider the following C++ assignment statement:

```
count = count + 5;
```

Some of the bindings and their binding times for the parts of this assignment statement are as follows:

- The type of `count` is bound at compile time.
- The set of possible values of `count` is bound at compiler design time.
- The meaning of the operator symbol + is bound at compile time, when the types of its operands have been determined.
- The internal representation of the literal `5` is bound at compiler design time.
- The value of `count` is bound at execution time with this statement.

A complete understanding of the binding times for the attributes of program entities is a prerequisite for understanding the semantics of a programming language. For example, to understand what a subprogram does, one must understand how the actual parameters in a call are bound to the formal parameters in its definition. To determine the current value of a variable, it may be necessary to know when the variable was bound to storage and with which statement or statements.

5.4.1 Binding of Attributes to Variables

A binding is **static** if it first occurs before run time begins and remains unchanged throughout program execution. If the binding first occurs during run time or can change in the course of program execution, it is called **dynamic**. The physical binding of a variable to a storage cell in a virtual memory environment is complex, because the page or segment of the address space in which the cell resides may be moved in and out of memory many times during program execution. In a sense, such variables are bound and unbound repeatedly. These bindings, however, are maintained by computer hardware, and the changes are invisible to the program and the user. Because they are not important to the discussion, we are not concerned with these hardware bindings. The essential point is to distinguish between static and dynamic bindings.

5.4.2 Type Bindings

Before a variable can be referenced in a program, it must be bound to a data type. The two important aspects of this binding are how the type is specified and when the binding takes place. Types can be specified statically through some form of explicit or implicit declaration.

5.4.2.1 Static Type Binding

An **explicit declaration** is a statement in a program that lists variable names and specifies that they are a particular type. An **implicit declaration** is a means of associating variables with types through default conventions, rather than declaration statements. In this case, the first appearance of a variable name in a program constitutes its implicit declaration. Both explicit and implicit declarations create static bindings to types.

Most widely used programming languages that use static type binding exclusively and were designed since the mid-1960s require explicit declarations of all variables (Visual Basic, ML, C#, and Swift are some exceptions).

Implicit variable type binding is done by the language processor, either a compiler or an interpreter. There are several different bases for implicit variable type bindings. The simplest of these is naming conventions. In this case, the compiler or interpreter binds a variable to a type based on the syntactic form of the variable's name.

Although they are a minor convenience to programmers, implicit declarations can be detrimental to reliability because they prevent the compilation process from detecting some typographical and programmer errors.

Some of the problems with implicit declarations can be avoided by requiring names for specific types to begin with particular special characters. For example, in Perl any name that begins with `$` is a scalar, which can store either a string or a numeric value. If a name begins with `@`, it is an array; if it begins with a `%`, it is a hash structure.3 This creates different namespaces for different type variables. In this scenario, the names `@apple` and `%apple` are unrelated, because each is from a different namespace. Furthermore, a program reader always knows the type of a variable when reading its name.

3. Both arrays and hashes are considered types—both can store any scalar in their elements.

Another kind of implicit type declarations uses context. This is sometimes called **type inference**. In the simpler case, the context is the type of the value assigned to the variable in a declaration statement. For example, in C# a **`var`** declaration of a variable must include an initial value, whose type is taken as

the type of the variable. Consider the following declarations:

```
var sum = 0;
var total = 0.0;
var name = "Fred";
```

The types of `sum`, `total`, and `name` are **`int`**, **`float`**, and **`string`**, respectively. Keep in mind that these are statically typed variables—their types are fixed for the lifetime of the unit in which they are declared.

Visual Basic, Swift, and the functional languages ML, Haskell, OCaml, and F# also use type inferencing.

5.4.2.2 Dynamic Type Binding

With dynamic type binding, the type of a variable is not specified by a declaration statement, nor can it be determined by the spelling of its name. Instead, the variable is bound to a type when it is assigned a value in an assignment statement. When the assignment statement is executed, the variable being assigned is bound to the type of the value of the expression on the right side of the assignment. Such an assignment may also bind the variable to an address and a memory cell, because different type values may require different amounts of storage. Any variable can be assigned any type value. Furthermore, a variable's type can change any number of times during program execution. It is important to realize that the type of a variable whose type is dynamically bound may be temporary.

When the type of a variable is statically bound, the name of the variable can be thought of being bound to a type, in the sense that the type and name of a variable are simultaneously bound. However, when a variable's type is dynamically bound, its name can be thought of as being only temporarily bound to a type. In reality, the names of variables are never bound to types. Names can be bound to variables and variables can be bound to types.

Languages in which types are dynamically bound are dramatically different from those in which types are statically bound. The primary advantage of dynamic binding of variables to types is that it provides more programming

flexibility. For example, a program to process numeric data in a language that uses dynamic type binding can be written as a generic program, meaning that it is capable of dealing with data of any numeric type. Whatever type data is input will be acceptable, because the variables in which the data are to be stored can be bound to the correct type when the data is assigned to the variables after input. By contrast, because of static binding of types, one cannot write a C program to process data without knowing the type of that data.

Before the mid-1990s, the most commonly used programming languages used static type binding, the primary exceptions being some functional languages such as Lisp. However, since then there has been a significant shift to languages that use dynamic type binding. In Python, Ruby, JavaScript, and PHP, type binding is dynamic. For example, a JavaScript script may contain the following statement:

```
list = [10.2, 3.5];
```

Regardless of the previous type of the variable named `list`, this assignment causes it to become the name of a single-dimensioned array of length 2. If the statement

```
list = 47;
```

followed the previous example assignment, `list` would become the name of a scalar variable.

The option of dynamic type binding was included in C# 2010. A variable can be declared to use dynamic type binding by including the **`dynamic`** reserved word in its declaration, as in the following example:

```
dynamic  any;
```

This is similar, although also different from declaring `any` to have type **`object`**. It is similar in that `any` can be assigned a value of any type, just as if it were declared **`object`**. It is different in that it is not useful for several different situations of interoperation; for example, with dynamically typed languages such as IronPython and IronRuby (.NET versions of Python and Ruby, respectively). However, it is useful when data of unknown type come

into a program from an external source. Class members, properties, method parameters, method return values, and local variables can all be declared **`dynamic`**.

In pure object-oriented languages—for example, Ruby—all variables are references and do not have types; all data are objects and any variable can reference any object. Variables in such languages are, in a sense, all the same type—they are references. However, unlike the references in Java, which are restricted to referencing one specific type of value, variables in Ruby can reference any object.

There are two disadvantages to dynamic type binding. First, it causes programs to be less reliable, because the error-detection capability of the compiler is diminished relative to a compiler for a language with static type bindings. Dynamic type binding allows any variable to be assigned a value of any type. Incorrect types of right sides of assignments are not detected as errors; rather, the type of the left side is simply changed to the incorrect type. For example, suppose that in a particular JavaScript program, `i` and `x` are currently the names of scalar numeric variables and `y` is currently the name of an array. Furthermore, suppose that the program needs the assignment statement

```
i = x;
```

but because of a keying error, it has the assignment statement

```
i = y;
```

In JavaScript (or any other language that uses dynamic type binding), no error is detected in this statement by the interpreter—the type of the variable named `i` is simply changed to an array. But later uses of `i` will expect it to be a scalar, and correct results will be impossible. In a language with static type binding, such as Java, the compiler would detect the error in the assignment `i = y`, and the program would not get to execution.

Note that this disadvantage is also present to some extent in some languages that use static type binding, such as C and C++, which in many cases automatically convert the type of the RHS of an assignment to the type of the

LHS.

Perhaps the greatest disadvantage of dynamic type binding is cost. The cost of implementing dynamic attribute binding is considerable, particularly in execution time. Type checking must be done at run time. Furthermore, every variable must have a run-time descriptor associated with it to maintain the current type. The storage used for the value of a variable must be of varying size, because different type values require different amounts of storage.

Finally, languages that have dynamic type binding for variables are usually implemented using pure interpreters rather than compilers. Computers do not have instructions whose operand types are not known at compile time. Therefore, a compiler cannot build machine instructions for the expression a + b if the types of a and b are not known at compile time. Pure interpretation typically takes at least 10 times as long as it does to execute equivalent machine code. Of course, if a language is implemented with a pure interpreter, the time to do dynamic type binding is hidden by the overall time of interpretation, so it seems less costly in that environment. On the other hand, languages with static type bindings are seldom implemented by pure interpretation, because programs in these languages can be easily translated to very efficient machine code versions.

5.4.3 Storage Bindings and Lifetime

The fundamental character of an imperative programming language is in large part determined by the design of the storage bindings for its variables. It is therefore important to have a clear understanding of these bindings.

The memory cell to which a variable is bound somehow must be taken from a pool of available memory. This process is called **allocation**. **Deallocation** is the process of placing a memory cell that has been unbound from a variable back into the pool of available memory.

The **lifetime** of a variable is the time during which the variable is bound to a

specific memory location. So, the lifetime of a variable begins when it is bound to a specific cell and ends when it is unbound from that cell. To investigate storage bindings of variables, it is convenient to separate scalar (unstructured) variables into four categories, according to their lifetimes. These categories are named static, stack-dynamic, explicit heap-dynamic, and implicit heap-dynamic. In the following sections, we discuss the definitions of these four categories, along with their purposes, advantages, and disadvantages.

5.4.3.1 Static Variables

A **static variable** is one that is bound to a memory cell before program execution begins and remains bound to that same memory cell until program execution terminates. Statically bound variables have several valuable applications in programming. Globally accessible variables are often used throughout the execution of a program, thus making it necessary to have them bound to the same storage during that execution. Sometimes it is convenient to have subprograms that are history sensitive. Such a subprogram must have local static variables.

One advantage of static variables is efficiency. All addressing of static variables can be direct;4 other kinds of variables often require indirect addressing, which is slower. Also, no run-time overhead is incurred for allocation and deallocation of static variables, although this time is often negligible.

4. In some implementations, static variables are addressed through a base register, making accesses to them as costly as for stack-allocated variables.

One disadvantage of static binding to storage is reduced flexibility; in particular, a language that has only static variables cannot support recursive subprograms. Another disadvantage is that storage cannot be shared among variables. For example, suppose a program has two subprograms, both of which require large arrays. Furthermore, suppose that the two subprograms are never active at the same time. If the arrays are static, they cannot share the same storage.

C and C++ allow programmers to include the **`static`** specifier on a variable definition in a function, making the variables it defines static. Note that when the **`static`** modifier appears in the declaration of a variable in a class definition in C++, Java, and C#, it also implies that the variable is a class variable, rather than an instance variable. Class variables are created statically some time before the class is first instantiated.

5.4.3.2 Stack-Dynamic Variables

Stack-dynamic variables are those whose storage bindings are created when their declaration statements are elaborated, but whose types are statically bound. **Elaboration** of such a declaration refers to the storage allocation and binding process indicated by the declaration, which takes place when execution reaches the code to which the declaration is attached. Therefore, elaboration occurs during run time. For example, the variable declarations that appear at the beginning of a Java method are elaborated when the method is called and the variables defined by those declarations are deallocated when the method completes its execution.

As their name indicates, stack-dynamic variables are allocated from the run-time stack.

Some languages—for example, C++ and Java—allow variable declarations to occur anywhere a statement can appear. In some implementations of these languages, all of the stack-dynamic variables declared in a function or method (not including those declared in nested blocks) may be bound to storage at the beginning of execution of the function or method, even though the declarations of some of these variables do not appear at the beginning. In such cases, the variable becomes visible at the declaration, but the storage binding (and initialization, if it is specified in the declaration) occurs when the function or method begins execution. The fact that storage binding of a variable takes place before it becomes visible does not affect the semantics of the language.

The advantages of stack-dynamic variables are as follows: To be useful, at least in most cases, recursive subprograms require some form of dynamic

local storage so that each active copy of the recursive subprogram has its own version of the local variables. These needs are conveniently met by stack-dynamic variables. Even in the absence of recursion, having stack-dynamic local storage for subprograms is not without merit, because all subprograms share the same memory space for their locals.

The disadvantages, relative to static variables, of stack-dynamic variables are the run-time overhead of allocation and deallocation, possibly slower accesses because indirect addressing is required, and the fact that subprograms cannot be history sensitive. The time required to allocate and deallocate stack-dynamic variables is not significant, because all of the stack-dynamic variables that are declared at the beginning of a subprogram are allocated and deallocated together, rather than by separate operations.

In Java, C++, and C#, variables defined in methods are by default stack dynamic.

All attributes other than storage are statically bound to stack-dynamic scalar variables. That is not the case for some structured types, as is discussed in Chapter 6. Implementation of allocation/deallocation processes for stack--dynamic variables is discussed in Chapter 10.

5.4.3.3 Explicit Heap-Dynamic Variables

Explicit heap-dynamic variables are nameless (abstract) memory cells that are allocated and deallocated by explicit run-time instructions written by the programmer. These variables, which are allocated from and deallocated to the heap, can only be referenced through pointer or reference variables. The heap is a collection of storage cells whose organization is highly disorganized due to the unpredictability of its use. The pointer or reference variable that is used to access an explicit heap-dynamic variable is created as any other scalar variable. An explicit heap-dynamic variable is created by either an operator (for example, in C++) or a call to a system subprogram provided for that purpose (for example, in C).

In C++, the allocation operator, named **new**, uses a type name as its operand. When executed, an explicit heap-dynamic variable of the operand type is created and its address is returned. Because an explicit heap-dynamic variable is bound to a type at compile time, that binding is static. However, such variables are bound to storage at the time they are created, which is during run time.

In addition to a subprogram or operator for creating explicit heap-dynamic variables, some languages include a subprogram or operator for explicitly destroying them.

As an example of explicit heap-dynamic variables, consider the following C++ code segment:

```
int  *intnode;      // Create a pointer
intnode =  new int; // Create the heap-dynamic variable
. . .
delete  intnode;   // Deallocate the heap-dynamic variable
                        // to which intnode points
```

In this example, an explicit heap-dynamic variable of **int** type is created by the **new** operator. This variable can then be referenced through the pointer, `intnode`. Later, the variable is deallocated by the **delete** operator. C++ requires the explicit deallocation operator **delete**, because it does not use implicit storage reclamation, such as garbage collection.

In Java, all data except the primitive scalars are objects. Java objects are explicitly heap dynamic and are accessed through reference variables. Java has no way of explicitly destroying a heap-dynamic variable; rather, implicit garbage collection is used. Garbage collection is discussed in Chapter 6.

C# has both explicit heap-dynamic and stack-dynamic objects, all of which are implicitly deallocated. In addition, C# supports C++-style pointers. Such pointers are used to reference heap, stack, and even static variables and objects. These pointers have the same dangers as those of C++, and the objects they reference on the heap are not implicitly deallocated. Pointers are included in C# to allow C# components to interoperate with C and C++ components. To discourage their use, and also to make clear to any program reader that the code uses pointers, the header of any method that defines a

pointer must include the reserved word **`unsafe`**.

Explicit heap-dynamic variables are often used to construct dynamic structures, such as linked lists and trees, that need to grow and/or shrink during execution. Such structures can be built conveniently using pointers or references and explicit heap-dynamic variables.

The disadvantages of explicit heap-dynamic variables are the difficulty of using pointer and reference variables correctly, the cost of references to the variables, and the complexity of the required storage management implementation. This is essentially the problem of heap management, which is costly and complicated. Implementation methods for explicit heap-dynamic variables are discussed at length in Chapter 6.

5.4.3.4 Implicit Heap-Dynamic Variables

Implicit heap-dynamic variables are bound to heap storage only when they are assigned values. In fact, all their attributes are bound every time they are assigned. For example, consider the following JavaScript assignment statement:

```
highs = [74, 84, 86, 90, 71];
```

Regardless of whether the variable named `highs` was previously used in the program or what it was used for, it is now an array of five numeric values.

The advantage of such variables is that they have the highest degree of flexibility, allowing highly generic code to be written. One disadvantage of implicit heap-dynamic variables is the run-time overhead of maintaining all the dynamic attributes, which could include array subscript types and ranges, among others. Another disadvantage is the loss of some error detection by the compiler, as discussed in Section 5.4.2.2.

5.5 Scope

One of the important factors in understanding variables is scope. The **scope** of a variable is the range of statements in which the variable is visible. A variable is **visible** in a statement if it can be referenced or assigned in that statement.

The scope rules of a language determine how a particular occurrence of a name is associated with a variable, or in the case of a functional language, how a name is associated with an expression. In particular, scope rules determine how references to variables declared outside the currently executing subprogram or block are associated with their declarations and thus their attributes (blocks are discussed in Section 5.5.2). A clear understanding of these rules for a language is therefore essential to the ability to write or read programs in that language.

A variable is **local** in a program unit or block if it is declared there. The nonlocal variables of a program unit or block are those that are visible within the program unit or block but are not declared there. Global variables are a special category of nonlocal variables, which are discussed in Section 5.5.4.

Scoping issues of classes, packages, and namespaces are discussed in Chapter 11.

5.5.1 Static Scope

ALGOL 60 introduced the method of binding names to nonlocal variables called **static scoping**,[5] which has been copied by many subsequent imperative languages and many nonimperative languages as well. Static scoping is so named because the scope of a variable can be statically determined—that is, prior to execution. This permits a human program reader (and a compiler) to determine the type of every variable in the program simply by examining its source code.

5. Static scoping is sometimes called *lexical scoping*.

There are two categories of static-scoped languages: those in which subprograms can be nested, which creates nested static scopes, and those in which subprograms cannot be nested. In the latter category, static scopes are also created by subprograms but nested scopes are created only by nested class definitions and blocks.

Ada, JavaScript, Common Lisp, Scheme, Fortran 2003+, F#, and Python allow nested subprograms, but the C-based languages do not.

Our discussion of static scoping in this section focuses on those languages that allow nested subprograms. Initially, we assume that *all* scopes are associated with program units and that all referenced nonlocal variables are declared in other program units.6 In this chapter, it is assumed that scoping is the only method of accessing nonlocal variables in the languages under discussion. This is not true for all languages. It is not even true for all languages that use static scoping, but the assumption simplifies the discussion here.

6. Nonlocal variables not defined in other program units are discussed in Section 5.5.4.

When the reader of a program finds a reference to a variable, the attributes of the variable can be determined by finding the statement in which it is declared (either explicitly or implicitly). In static-scoped languages with nested subprograms, this process can be thought of in the following way. Suppose a reference is made to a variable `x` in subprogram `sub1`. The correct declaration is found by first searching the declarations of subprogram `sub1`. If no declaration is found for the variable there, the search continues in the declarations of the subprogram that declared subprogram `sub1`, which is called its **static parent**. If a declaration of `x` is not found there, the search continues to the next-larger enclosing unit (the unit that declared `sub1`'s parent), and so forth, until a declaration for `x` is found or the largest unit's declarations have been searched without success. In that case, an undeclared variable error is reported. The static parent of subprogram `sub1`, and its static parent, and so forth up to and including the largest enclosing subprogram, are called the **static ancestors** of `sub1`. Actual implementation techniques for

static scoping, which are discussed in Chapter 10, are usually much more efficient than the process just described.

Consider the following JavaScript function, `big`, in which the two functions `sub1` and `sub2` are nested:

```
function  big() {
  function  sub1() {
    var  x = 7;
    sub2();
  }
  function  sub2() {
    var  y = x;
  }
  var  x = 3;
  sub1();
}
```

Under static scoping, the reference to the variable `x` in `sub2` is to the `x` declared in the procedure `big`. This is true because the search for `x` begins in the procedure in which the reference occurs, `sub2`, but no declaration for `x` is found there. The search continues in the static parent of `sub2`, `big`, where the declaration of `x` is found. The `x` declared in `sub1` is ignored, because it is not in the static ancestry of `sub2`.

In some languages that use static scoping, regardless of whether nested subprograms are allowed, some variable declarations can be hidden from some other code segments. For example, consider again the JavaScript function `big`. The variable `x` is declared in both `big` and in `sub1`, which is nested inside `big`. Within `sub1`, every simple reference to `x` is to the local `x`. Therefore, the outer `x` is hidden from `sub1`.

5.5.2 Blocks

Many languages allow new static scopes to be defined in the midst of executable code. This powerful concept, introduced in ALGOL 60, allows a section of code to have its own local variables whose scope is minimized. Such variables are typically stack dynamic, so their storage is allocated when the section is entered and deallocated when the section is exited. Such a

section of code is called a **block**. Blocks provide the origin of the phrase **block-structured language**.

The C-based languages allow any compound statement (a statement sequence surrounded by matched braces) to have declarations and thereby define a new scope. Such compound statements are called blocks. For example, if `list` were an integer array, one could write the following:

```
if (list[i] < list[j]) {
  int temp;
  temp = list[i];
  list[i] = list[j];
  list[j] = temp;
}
```

The scopes created by blocks, which could be nested in larger blocks, are treated exactly like those created by subprograms. References to variables in a block that are not declared there are connected to declarations by searching enclosing scopes (blocks or subprograms) in order of increasing size.

Consider the following skeletal C function:

```
void sub() {
  int count;
  . . .
  while (. . .) {
    int count;
    count++;
    . . .
  }
  . . .
}
```

The reference to `count` in the **`while`** loop is to that loop's local `count`. In this case, the `count` of `sub` is hidden from the code inside the **`while`** loop. In general, a declaration for a variable effectively hides any declaration of a variable with the same name in a larger enclosing scope.[7] Note that this code is legal in C and C++ but illegal in Java and C#. The designers of Java and C# believed that the reuse of names in nested blocks was too error prone to be allowed.

7. As discussed in Section 5.5.4, in C++, such hidden global variables can be accessed in the inner scope using the scope operator (::).

Although JavaScript uses static scoping for its nested functions, nonfunction blocks cannot be defined in the language.

Most functional programming languages include a construct that is related to the blocks of the imperative languages, usually named *let*. These constructs have two parts, the first of which is to bind names to values, usually specified as expressions. The second part is an expression that uses the names defined in the first part. Programs in functional languages are comprised of expressions, rather than statements. Therefore, the final part of a `let` construct is an expression, rather than a statement. In Scheme, a `let` construct is a call to the function `LET` with the following form:

- `(LET (`
- `( name1 expression1)`
- `. . .`
- `( namen expressionn))`
- expression
- `)`

The semantics of the call to `LET` is as follows: The first *n* expressions are evaluated and the values are assigned to the associated names. Then, the final expression is evaluated and the return value of `LET` is that value. This differs from a block in an imperative language in that the names are of values; they are not variables in the imperative sense. Once set, they cannot be changed. However, they are like local variables in a block in an imperative language in that their scope is local to the call to `LET`. Consider the following call to `LET`:

```
(LET (
  (top (+ a b))
  (bottom (- c d)))
  (/ top bottom)
```

`)`

This call computes and returns the value of the expression `(a + b) / (c - d)`.

In ML, the form of a **`let`** construct is as follows:

- **`let`**
- **`val`** name1 = expression1
- `...`
- **`val`** namen = expressionn
- **`in`**
- expression
- **`end`**;

Each **`val`** statement binds a name to an expression. As with Scheme, the names in the first part are like the named constants of imperative languages; once set, they cannot be changed.[8] Consider the following **`let`** construct:

8. In Chapter 15, we will see that they can be reset, but that the process actually creates a new name.

```
let
  val  top = a + b
  val bottom = c - d
in
  top / bottom
end;
```

The general form of a **`let`** construct in F# is as follows:

- **`let`** left_side = expression

The left_side of **`let`** can be a name or a tuple pattern (a sequence of names separated by commas).

The scope of a name defined with **let** inside a function definition is from the end of the defining expression to the end of the function. The scope of **let** can be limited by indenting the following code, which creates a new local scope. Although any indentation will work, the convention is that the indentation is four spaces. Consider the following code:

```
let n1 =
    let  n2 = 7
    let  n3 = n2 + 3
    n3;;
let n4 = n3 + n1;;
```

The scope of `n1` extends over all of the code. However, the scope of `n2` and `n3` ends when the indentation ends. So, the use of `n3` in the last **let** causes an error. The last line of the **let** `n1` scope is the value bound to `n1`; it could be any expression.

Chapter 15 includes more details of the let constructs in Scheme, ML, Haskell, and F#.

5.5.3 Declaration Order

In C89, as well as in some other languages, all data declarations in a function except those in nested blocks must appear at the beginning of the function. However, some languages—for example, C99, C++, Java, JavaScript, and C#—allow variable declarations to appear anywhere a statement can appear in a program unit. Declarations may create scopes that are not associated with compound statements or subprograms. For example, in C99, C++, and Java, the scope of all local variables is from their declarations to the ends of the blocks in which those declarations appear.

In the official documentation for C#, the scope of any variable declared in a block is said to be the whole block, regardless of the position of the declaration in the block, as long as it is not in a nested block. The same is true for methods. However, this is misleading, because the C# language definition requires that all variables be declared before they are used. Therefore, although the scope of a variable is said to extend from the declaration to the

top of the block or subprogram in which that declaration appears, the variable still cannot be used above its declaration.

Recall that C# does not allow the declaration of a variable in a nested block to have the same name as a variable in a nesting scope. This, together with the rule that the scope of a declaration is the whole block, makes the following nested declaration of `x` illegal:

```
{
  {int  x;  // Illegal
  ...
  }
  int  x;
}
```

Note that C# still requires that all be declared before they are used. Therefore, although the scope of a variable extends from the declaration to the top of the block or subprogram in which that declaration appears, the variable still cannot be used above its declaration.

In JavaScript, local variables can be declared anywhere in a function, but the scope of such a variable is always the entire function. If used before its declaration in the function, such a variable has the value `undefined`. The reference is not illegal.

The **`for`** statements of C++, Java, and C# allow variable definitions in their initialization expressions. In early versions of C++, the scope of such a variable was from its definition to the end of the smallest enclosing block. In the standard version, however, the scope is restricted to the **`for`** construct, as is the case with Java and C#. Consider the following skeletal method:

```
void  fun() {
   . . .
   for (int  count = 0; count < 10; count++){
     . . .
   }
   . . .
}
```

In later versions of C++, as well as in Java and C#, the scope of `count` is from the **`for`** statement to the end of its body (the right brace).

5.5.4 Global Scope

Some languages, including C, C++, PHP, JavaScript, and Python, allow a program structure that is a sequence of function definitions, in which variable definitions can appear outside the functions. Definitions outside functions in a file create global variables, which potentially can be visible to those functions.

C and C++ have both declarations and definitions of global data. Declarations specify types and other attributes but do not cause allocation of storage. Definitions specify attributes *and* cause storage allocation. For a specific global name, a C program can have any number of compatible declarations, but only a single definition.

A declaration of a variable outside function definitions specifies that the variable is defined in a different file. A global variable in C is implicitly visible in all subsequent functions in the file, except those that include a declaration of a local variable with the same name. A global variable that is defined after a function can be made visible in the function by declaring it to be external, as in the following:

```
extern int  sum;
```

In C99, definitions of global variables usually have initial values. Declarations of global variables never have initial values. If the declaration is outside function definitions, it need not include the **`extern`** qualifier.

This idea of declarations and definitions carries over to the functions of C and C++, where prototypes declare names and interfaces of functions but do not provide their code. Function definitions, on the other hand, are complete.

In C++, a global variable that is hidden by a local with the same name can be accessed using the scope operator (`::`). For example, if `x` is a global that is hidden in a function by a local named `x`, the global could be referenced as `::x`.

PHP statements can be interspersed with function definitions. Variables in

PHP are implicitly declared when they appear in statements. Any variable that is implicitly declared outside any function is a global variable; variables implicitly declared in functions are local variables. The scope of global variables extends from their declarations to the end of the program but skips over any subsequent function definitions. So, global variables are not implicitly visible in any function. Global variables can be made visible in functions in their scope in two ways: (1) If the function includes a local variable with the same name as a global, that global can be accessed through the `$GLOBALS` array, using the name of the global as a string literal subscript, and (2) if there is no local variable in the function with the same name as the global, the global can be made visible by including it in a `global` declaration statement. Consider the following example:

```
$day = "Monday";
$month = "January";
function calendar() {
  $day = "Tuesday";
  global $month;
  print "local day is $day  ";
  $gday = $GLOBALS['day'];
  print "global day is $gday <br \>";
  print "global month is $month ";
}
calendar();
```

Interpretation of this code produces the following:

```
local day is Tuesday
global day is Monday
global month is January
```

The global variables of JavaScript are very similar to those of PHP, except that there is no way to access a global variable in a function that has declared a local variable with the same name.

The visibility rules for global variables in Python are unusual. Variables are not normally declared, as in PHP. They are implicitly declared when they appear as the targets of assignment statements. A global variable can be referenced in a function, but a global variable can be assigned in a function only if it has been declared to be global in the function. Consider the following examples:

```
day = "Monday"
def tester():
print "The global day is:", day
tester()
```

The output of this script, because globals can be referenced directly in -functions, is as follows:

```
The global day is: Monday
```

The following script attempts to assign a new value to the global `day`:

```
day = "Monday"
def  tester():
  print "The global day is:", day
  day = "Tuesday"
  print "The new value of day is:", day
tester()
```

This script creates an `UnboundLocalError` error message, because the assignment to `day` in the second line of the body of the function makes `day` a local variable, which makes the reference to `day` in the first line of the body of the function an illegal forward reference to the local.

The assignment to `day` can be to the global variable if `day` is declared to be global at the beginning of the function. This prevents the assignment to `day` from creating a local variable. This is shown in the following script:

```
day = "Monday"
def tester():
  global  day
  print  "The global day is:", day
  day = "Tuesday"
  print  "The new value of day is:", day
tester()
```

The output of this script is as follows:

```
The global day is: Monday
The new value of day is: Tuesday
```

Functions can be nested in Python. Variables defined in nesting functions are accessible in a nested function through static scoping, but such variables must

be declared **`nonlocal`** in the nested function.9 An example skeletal program in Section 5.7 illustrates accesses to nonlocal variables.

9. The **`nonlocal`** reserved word was introduced in Python 3.

All names defined outside function definitions in F# are globals. Their scope extends from their definitions to the end of the file.

Declaration order and global variables are also issues in the class and member declarations in object-oriented languages. These are discussed in Chapter 12.

5.5.5 Evaluation of Static Scoping

Static scoping provides a method of nonlocal access that works well in many situations. However, it is not without its problems. First, in most cases it allows more access to both variables and subprograms than is necessary. It is simply too crude a tool for concisely specifying such restrictions. Second, and perhaps more important, is a problem related to program evolution. Software is highly dynamic—programs that are used regularly continually change. These changes often result in restructuring, thereby destroying the initial structure that restricted variable and subprogram access in a static-scoped language. To avoid the complexity of maintaining these access restrictions, developers often discard structure when it gets in the way. Thus, getting around the restrictions of static scoping can lead to program designs that bear little resemblance to the original, even in areas of the program in which changes have not been made. Designers are encouraged to use far more globals than are necessary. All subprograms can end up being nested at the same level, in the main program, using globals instead of deeper levels of nesting.10 Moreover, the final design may be awkward and contrived, and it may not reflect the underlying conceptual design. These and other defects of static scoping are discussed in detail in Clarke, Wileden, and Wolf (1980). An alternative to the use of static scoping to control access to variables and subprograms is an encapsulation construct, which is included in many newer languages. Encapsulation constructs are discussed in Chapter 11.

10. Sounds like the structure of a C program, doesn’t it?

5.5.6 Dynamic Scope

The scope of variables in APL, SNOBOL4, and the early versions of Lisp is dynamic. Perl and Common Lisp also allow variables to be declared to have dynamic scope, although the default scoping mechanism in these languages is static. **Dynamic scoping** is based on the calling sequence of subprograms, not on their spatial relationship to each other. Thus, the scope can be determined only at run time.

Consider again the function `big` from Section 5.5.1, which is reproduced here, minus the function calls:

```
function  big() {
  function  sub1() {
    var  x = 7;
  }
  function  sub2() {
    var  y = x;
    var  z = 3;
  }
  var  x = 3;
}
```

Assume that dynamic-scoping rules apply to nonlocal references. The meaning of the identifier `x` referenced in `sub2` is dynamic—it cannot be determined at compile time. It may reference the variable from either declaration of `x`, depending on the calling sequence.

One way the correct meaning of `x` can be determined during execution is to begin the search with the local declarations. This is also the way the process begins with static scoping, but that is where the similarity between the two techniques ends. When the search of local declarations fails, the declarations of the dynamic parent, or calling function, are searched. If a declaration for `x` is not found there, the search continues in that function's dynamic parent, and so forth, until a declaration for `x` is found. If none is found in any dynamic ancestor, it is a run-time error.

Consider the two different call sequences for `sub2` in the earlier example. First, `big` calls `sub1`, which calls `sub2`. In this case, the search proceeds from

the local procedure, `sub2`, to its caller, `sub1`, where a declaration for `x` is found. So, the reference to `x` in `sub2` in this case is to the `x` declared in `sub1`. Next, `sub2` is called directly from `big`. In this case, the dynamic parent of `sub2` is `big`, and the reference is to the `x` declared in `big`.

Note that if static scoping were used, in either calling sequence discussed, the reference to `x` in `sub2` would be to `big`'s `x`.

Perl's dynamic scoping is unusual—in fact, it is not exactly like that discussed in this section, although the semantics are often that of traditional dynamic scoping (see Programming Exercise 1).

5.5.7 Evaluation of Dynamic Scoping

The effect of dynamic scoping on programming is profound. When dynamic scoping is used, the correct attributes of nonlocal variables visible to a program statement cannot be determined statically. Furthermore, a reference to the name of such a variable is not always to the same variable. A statement in a subprogram that contains a reference to a nonlocal variable can refer to different nonlocal variables during different executions of the subprogam. Several kinds of programming problems follow directly from dynamic scoping.

First, during the time span beginning when a subprogram begins its execution and ending when that execution ends, the local variables of the subprogram are all visible to any other executing subprogram, regardless of its textual proximity or how execution got to the currently executing subprogram. There is no way to protect local variables from this accessibility. Subprograms are *always* executed in the environment of all previously called subprograms that have not yet completed their executions. As a result, dynamic scoping results in less reliable programs than static scoping.

A second problem with dynamic scoping is the inability to type check references to nonlocals statically. This problem results from the inability to

statically find the declaration for a variable referenced as a nonlocal.

Dynamic scoping also makes programs much more difficult to read, because the calling sequence of subprograms must be known to determine the meaning of references to nonlocal variables. This task can be virtually impossible for a human reader.

Finally, accesses to nonlocal variables in dynamic-scoped languages take far longer than accesses to nonlocals when static scoping is used. The reason for this is explained in Chapter 10.

On the other hand, dynamic scoping is not without merit. In many cases, the parameters passed from one subprogram to another are variables that are defined in the caller. None of these needs to be passed in a dynamically scoped language, because they are implicitly visible in the called subprogram.

It is not difficult to understand why dynamic scoping is not as widely used as static scoping. Programs in static-scoped languages are easier to read, are more reliable, and execute faster than equivalent programs in dynamic-scoped languages. It was precisely for these reasons that dynamic scoping was replaced by static scoping in most current dialects of Lisp. Implementation methods for both static and dynamic scoping are discussed in Chapter 10.

5.6 Scope and Lifetime

Sometimes the scope and lifetime of a variable appear to be related. For example, consider a variable that is declared in a Java method that contains no method calls. The scope of such a variable is from its declaration to the end of the method. The lifetime of that variable is the period of time beginning when the method is entered and ending when execution of the method terminates. Although the scope and lifetime of the variable are clearly not the same, because static scope is a textual, or spatial, concept whereas lifetime is a temporal concept, they at least appear to be related in this case.

This apparent relationship between scope and lifetime does not hold in other situations. In C and C++, for example, a variable that is declared in a function using the specifier **`static`** is statically bound to the scope of that function and is also statically bound to storage. So, its scope is static and local to the function, but its lifetime extends over the entire execution of the program of which it is a part.

Scope and lifetime are also unrelated when subprogram calls are involved. Consider the following C++ functions:

```
void printheader() {
  . . .
 } /* end of printheader */
void compute() {
  int sum;
  . . .
  printheader();
 } /* end of compute */
```

The scope of the variable `sum` is completely contained within the `compute` function. It does not extend to the body of the function `printheader`, although `printheader` executes in the midst of the execution of `compute`. However, the lifetime of `sum` extends over the time during which `printheader` executes. Whatever storage location `sum` is bound to before the call to `printheader`, that binding will continue during and after the execution of `printheader`.

5.7 Referencing Environments

The **referencing environment** of a statement is the collection of all variables that are visible in the statement. The referencing environment of a statement in a static-scoped language is the variables declared in its local scope plus the collection of all variables of its ancestor scopes that are visible. In such a language, the referencing environment of a statement is needed while that statement is being compiled, so code and data structures can be created to allow references to variables from other scopes during run time. Techniques for implementing references to nonlocal variables in both static- and dynamic-scoped languages are discussed in Chapter 10.

In Python, scopes can be created by function definitions. The referencing environment of a statement includes the local variables, plus all of the variables declared in the functions in which the statement is nested (excluding variables in nonlocal scopes that are hidden by declarations in nearer functions). Each function definition creates a new scope and thus a new environment. Consider the following Python skeletal program:

```
g = 3; # A global
def sub1():
    a = 5; # Creates a local
    b = 7; # Creates another local
    . . . <------------------------------ 1
  def  sub2():
  global g;  # Global g is now assignable here
    c = 9;  # Creates a new local
    . . . <------------------------------ 2
    def  sub3():
      nonlocal  c:  # Makes nonlocal c visible here
      g = 11;  # Creates a new local
      . . . <------------------------------ 3
```

The referencing environments of the indicated program points are as follows:

Point	*Referencing Environment*
1	local `a` and `b` (of `sub1`), global `g` for reference, but not for assignment
2	local `c` (of `sub2`), global `g` for both reference and for assignment
3	nonlocal `c` (of `sub2`), local `g` (of `sub3`)

Now consider the variable declarations of this skeletal program. First, note that, although the scope of `sub1` is at a higher level (it is less deeply nested) than `sub3`, the scope of `sub1` is not a static ancestor of `sub3`, so `sub3` does not have access to the variables declared in `sub1`. There is a good reason for this. The variables declared in `sub1` are stack dynamic, so they are not bound to storage if `sub1` is not in execution. Because `sub3` can be in execution when `sub1` is not, it cannot be allowed to access variables in `sub1`, which would not necessarily be bound to storage during the execution of `sub3`.

A subprogram is **active** if its execution has begun but has not yet terminated. The referencing environment of a statement in a dynamically scoped language is the locally declared variables, plus the variables of all other - subprograms that are currently active. Once again, some variables in active subprograms can be hidden from the referencing environment. Recent subprogram activations can have declarations for variables that hide variables with the same names in previous subprogram activations.

Consider the following example program. Assume that the only function calls are the following: `main` calls `sub2`, which calls `sub1`.

```
void sub1() {
  int a, b;
  . . . <------------ 1
}  /* end of sub1 */
void sub2() {
  int b, c;
  . . . . <------------ 2
  sub1();
}  /* end of sub2 */
void main() {
  int c, d;
```

```
  . . . <------------ 3
  sub2();
}  /* end of main */
```

The referencing environments of the indicated program points are as follows:

Point	*Referencing Environment*
1	`a` and `b` of `sub1`, `c` of `sub2`, `d` of `main`, (`c` of `main` and `b` of `sub2` are hidden)
2	`b` and `c` of `sub2`, `d` of `main`, (`c` of `main` is hidden)
3	`c` and `d` of `main`

5.8 Named Constants

A **named constant** is a variable that is bound to a value only once. Named constants are useful as aids to readability and program reliability. Readability can be improved, for example, using the name `pi` instead of the constant `3.14159265`.

Another important use of named constants is to parameterize a program. For example, consider a program that processes a fixed number of data values, say 100. Such a program usually uses the constant `100` in a number of locations for declaring array subscript ranges and for `loop` control limits. Consider the following skeletal Java program segment:

```
void example() {
  int[] intList = new int[100];
  String[] strList =  new  String[100];
  . . .
  for  (index = 0; index < 100; index++) {
    . . .
  }
  . . .
  for  (index = 0; index < 100; index++) {
    . . .
  }
  . . .
  average = sum / 100;
  . . .
}
```

When this program must be modified to deal with a different number of data values, all occurrences of `100` must be found and changed. On a large program, this can be tedious and error prone. An easier and more reliable method is to use a named constant as a program parameter, as follows:

```
void example() {
  final int  len = 100;
  int[] intList =  new int[len];
  String[] strList =  new  String[len];
  . . .
  for (index = 0; index < len; index++) {
```

```
    . . .
  }
  . . .
  for (index = 0; index < len; index++) {
    . . .
  }
  . . .
  average = sum / len;
  . . .
}
```

Now, when the length must be changed, only one line must be changed (the variable `len`), regardless of the number of times it is used in the program. This is another example of the benefits of abstraction. The name `len` is an abstraction for the number of elements in some arrays and the number of iterations in some loops. This illustrates how named constants can aid modifiability.

C++ allows dynamic binding of values to named constants. This allows expressions containing variables to be assigned to constants in the declarations. For example, the C++ statement

```
const int  result = 2 * width + 1;
```

declares `result` to be an integer type named constant whose value is set to the value of the expression `2 * width + 1`, where the value of the variable `width` must be visible when `result` is allocated and bound to its value.

Java also allows dynamic binding of values to named constants. In Java, named constants are defined with the **`final`** reserved word (as in the earlier example). The initial value can be given in the declaration statement or in a subsequent assignment statement. The assigned value can be specified with any expression.

C# has two kinds of named constants: those defined with **`const`** and those defined with **`readonly`**. The **`const`** named constants, which are implicitly **`static`**, are statically bound to values; that is, they are bound to values at compile time, which means those values can be specified only with literals or other **`const`** members. The **`readonly`** named constants, which are dynamically bound to values, can be assigned in the declaration or with a

static constructor.11 So, if a program needs a constant-valued object whose value is the same on every use of the program, a **`const`** constant is used. However, if a program needs a constant-valued object whose value is determined only when the object is created and can be different for different executions of the program, then a **`readonly`** constant is used.

11. Static constructors in C# run at some indeterminate time before the class is instantiated.

The discussion of binding values to named constants naturally leads to the topic of initialization, because binding a value to a named constant is the same process, except it is permanent.

In many instances, it is convenient for variables to have values before the code of the program or subprogram in which they are declared begins executing. The binding of a variable to a value at the time it is bound to storage is called **initialization**. If the variable is statically bound to storage, binding and initialization occur before run time. In these cases, the initial value must be specified as a literal or an expression whose only nonliteral operands are named constants that have already been defined. If the storage binding is dynamic, initialization is also dynamic and the initial values can be any expression.

In most languages, initialization is specified on the declaration that creates the variable. For example, in C++, we could have

```
int sum = 0;
int* ptrSum = &sum;
char name[] = "George Washington Carver";
```

SUMMARY

Case sensitivity and the use of underscores are the design issues for names.

Variables can be characterized by the sextuple of attributes: name, address, value, type, lifetime, and scope.

Aliases are two or more variables bound to the same storage address. They are regarded as detrimental to reliability but are difficult to eliminate entirely from a language.

Binding is the association of attributes with program entities. Knowledge of the binding times of attributes to entities is essential to understanding the semantics of programming languages. Binding can be static or dynamic. - Declarations, either explicit or implicit, provide a means of specifying the static binding of variables to types. In general, dynamic binding allows greater flexibility but at the expense of readability, efficiency, and reliability.

Scalar variables can be separated into four categories by considering their lifetimes: static, stack dynamic, explicit heap dynamic, and implicit heap dynamic.

Static scoping is a central feature of ALGOL 60 and some of its descendants. It provides a simple, reliable, and efficient method of allowing visibility of nonlocal variables in subprograms. Dynamic scoping provides more flexibility than static scoping but, again, at the expense of readability, reliability, and efficiency.

Most functional languages allow the user to create local scopes with `let` constructs, which limit the scope of their defined names.

The referencing environment of a statement is the collection of all of the variables that are visible to that statement.

Named constants are simply variables that are bound to values only once.

REVIEW QUESTIONS

1. What are the design issues for names?
2. What is the potential danger of case-sensitive names?
3. What is an alias?
4. Which category of C++ reference variables always produces aliases?
5. What is the *l*-value of a variable? What is the *r*-value?
6. Define *binding* and *binding time*.
7. After language design and implementation, what are the four times bindings can take place in a program?
8. Define *static binding* and *dynamic binding*.
9. What are the advantages and disadvantages of implicit declarations?
10. What are the advantages and disadvantages of dynamic type binding?
11. Define *static*, *stack-dynamic*, *explicit heap-dynamic*, and *implicit heap-dynamic variables*. What are their advantages and disadvantages?
12. Define *lifetime, scope, static scope,* and *dynamic scope*.
13. How is a reference to a nonlocal variable in a static-scoped program connected to its definition?
14. What is the general problem with static scoping?
15. What is the referencing environment of a statement?
16. What is a static ancestor of a subprogram? What is a dynamic ancestor of a subprogram?

17. What is a block?

18. What is the purpose of the `let` constructs in functional languages?

19. What is the difference between the names defined in an ML `let` construct from the variables declared in a C block?

20. Describe the encapsulation of an F# `let` inside a function and outside all functions.

21. What are the advantages and disadvantages of dynamic scoping?

22. What are the advantages of named constants?

PROBLEM SET

1. Which of the following identifier forms is most readable? Support your decision.

   ```
   SumOfSales
   sum_of_sales
   SUMOFSALES
   ```

2. Some programming languages are typeless. What are the obvious advantages and disadvantages of having no types in a language?

3. Write a simple assignment statement with one arithmetic operator in some language you know. For each component of the statement, list the various bindings that are required to determine the semantics when the statement is executed. For each binding, indicate the binding time used for the language.

4. Dynamic type binding is closely related to implicit heap-dynamic variables. Explain this relationship.

5. Describe a situation when a history-sensitive variable in a subprogram is useful.

6. Consider the following JavaScript skeletal program:

   ```
   // The main program
   var  x;
   function  sub1() {
     var  x;
     function  sub2() {
       . . .
     }
   }
   function  sub3() {
     . . .
   }
   ```

   Assume that the execution of this program is in the following unit order:

- `main` calls `sub1`
- `sub1` calls `sub2`
- `sub2` calls `sub3`

1. Assuming static scoping, in the following, which declaration of `x` is the correct one for a reference to `x`?
    1. `sub1`
    2. `sub2`
    3. `sub3`
2. Repeat part a, but assume dynamic scoping.

7. Assume the following JavaScript program was interpreted using static-scoping rules. What value of `x` is displayed in function `sub1`? Under dynamic-scoping rules, what value of `x` is displayed in function `sub1`?

```
var x;
function  sub1() {
  document.write("x = " + x + "");
}
function  sub2() {
   var x;
   x = 10;
   sub1();
}
x = 5;
sub2();
```

8. Consider the following JavaScript program:

```
var  x, y, z;
function  sub1() {
  var  a, y, z;
  function sub2() {
    var  a, b, z;
    . . .
  }
```

```
  . . .
}
function  sub3() {
  var  a, x, w;
  . . .
}
```

List all the variables, along with the program units where they are declared, that are visible in the bodies of `sub1`, `sub2`, and `sub3`, assuming static scoping is used.

9. Consider the following Python program:

```
x = 1;
y = 3;
z = 5;
def sub1():
  a = 7;
  y = 9;
  z = 11;
  . . .
def sub2():
  global  x;
  a = 13;
  x = 15;
  w = 17;
  . . .
  def sub3():
    nonlocal  a;
    a = 19;
    b = 21;
    z = 23;
    . . .
. . .
```

List all the variables, along with the program units where they are declared, that are visible in the bodies of `sub1`, `sub2`, and `sub3`, assuming static scoping is used.

10. Consider the following C program:

```
void fun(void) {
  int a, b, c; /* definition 1 */
  . . .
  while (. . .) {
```

```
    int b, c, d; /*definition 2 */
    . . . <-------------  1
    while (. . .) {
    int c, d, e; /* definition 3 */
    . . . <-------------  2
    }
   . . . <--------------  3
   }
  . . . <----------------  4
}
```

For each of the four marked points in this function, list each visible - variable, along with the number of the definition statement that defines it.

11. Consider the following skeletal C program:

```
void fun1(void);  /* prototype */
void fun2(void);  /* prototype */
void fun3(void);  /* prototype */
void main() {
  int a, b, c;
  . . .
 }
void  fun1(void) {
  int  b, c, d;
  . . .
 }
void  fun2(void) {
  int  c, d, e;
  . . .
 }
void  fun3(void) {
  int  d, e, f;
  . . .
 }
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.

1. `main` calls `fun1`; `fun1` calls `fun2`; `fun2` calls `fun3`.

2. `main` calls `fun1`; `fun1` calls `fun3`.

3. `main` calls `fun2`; `fun2` calls `fun3`; `fun3` calls `fun1`.

4. `main` calls `fun3`; `fun3` calls `fun1`.

5. `main` calls `fun1`; `fun1` calls `fun3`; `fun3` calls `fun2`.

6. `main` calls `fun3`; `fun3` calls `fun2`; `fun2` calls `fun1`.

12. Consider the following program, written in JavaScript-like syntax:

```
// main program
var  x, y, z;
function  sub1() {
var  a, y, z;
. . .
}
function  sub2() {
  var  a, b, z;
  . . .
}
function  sub3() {
  var  a, x, w;
  . . .
}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last subprogram activated? Include with each visible variable the name of the unit where it is declared.

1. `main` calls `sub1`; `sub1` calls `sub2`; `sub2` calls `sub3`.

2. `main` calls `sub1`; `sub1` calls `sub3`.

3. `main` calls `sub2`; `sub2` calls `sub3`; `sub3` calls `sub1`.

4. `main` calls `sub3`; `sub3` calls `sub1`.

5. `main` calls `sub1`; `sub1` calls `sub3`; `sub3` calls `sub2`.

6. `main` calls `sub3`; `sub3` calls `sub2`; `sub2` calls `sub1`.

PROGRAMMING EXERCISES

1. Perl allows both static and a kind of dynamic scoping. Write a Perl program that uses both and clearly shows the difference in effect of the two. Explain clearly the difference between the dynamic scoping described in this chapter and that implemented in Perl.

2. Write a Common Lisp program that clearly shows the difference between static and dynamic scoping.

3. Write a JavaScript script that has subprograms nested three deep and in which each nested subprogram references variables defined in all of its enclosing subprograms.

4. Repeat Programming Exercise 3 with Python.

5. Write a C99 function that includes the following sequence of statements:

```
x = 21;
int x;
x = 42;
```

   Run the program and explain the results. Rewrite the same code in C++ and Java and compare the results.

6. Write test programs in C++, Java, and C# to determine the scope of a variable declared in a **`for`** statement. Specifically, the code must determine whether such a variable is visible after the body of the **`for`** statement.

7. Write three functions in C or C++: one that declares a large array statically, one that declares the same large array on the stack, and one that creates the same large array from the heap. Call each of the subprograms a large number of times (at least 100,000) and output the time required by each. Explain the results.

6 Data Types

This chapter first introduces the concept of a data type and the characteristics of the common primitive data types. Then, the designs of enumeration and subrange types are discussed. Next, the details of structured data types—

specifically arrays, associative arrays, records, tuples, lists, and unions—are investigated. This section is followed by an in-depth look at pointers and references. The last category of types discussed are the optional types.

For each of the various categories of data types, the design issues are stated and the design choices made by the designers of some common languages are described. These designs are then evaluated.

The next three sections provide a thorough investigation of type checking, strong typing, and type equivalence rules. The last section of the chapter briefly introduces the fundamentals of the theory of data types.

Implementation methods for data types sometimes have a significant impact on their design. Therefore, implementation of the various data types is another important part of this chapter, especially arrays.

6.1 Introduction

A **data type** defines a collection of data values and a set of predefined operations on those values. Computer programs produce results by manipulating data. An important factor in determining the ease with which they can perform this task is how well the data types available in the language being used match the objects in the real world of the problem being addressed. Therefore, it is crucial that a language supports an appropriate collection of data types and structures.

The contemporary concepts of data typing have evolved over the last 60 years. In the earliest languages, all problem space data structures had to be modeled with only a few basic language-supported data structures. For example, in pre-90 Fortrans, linked lists and binary trees were implemented with arrays.

The data structures of COBOL took the first step away from the Fortran I model by allowing programmers to specify the accuracy of decimal data values, and also by providing a structured data type for records of information. PL/I extended the capability of accuracy specification to integer and floating-point types. The designers of PL/I included many data types, with the intent of supporting a large range of applications. A better approach, introduced in ALGOL 68, is to provide a few basic types and a few flexible structure-defining operators that allow a programmer to design a data structure for each need. Clearly, this was one of the most important advances in the evolution of data type design. User-defined types also provide improved readability through the use of meaningful names for types. They allow type checking of the variables of a special category of use, which would otherwise not be possible. User-defined types also aid modifiability: A programmer can change the type of a category of variables in a program by changing a type definition statement only.

Taking the concept of a user-defined type a step further, we arrive at abstract data types, which are supported by most programming languages designed since the mid-1980s. The fundamental idea of an abstract data type is that the

interface of a type, which is visible to the user, is separated from the representation and set of operations on values of that type, which are hidden from the user. All of the types provided by a high-level programming language are abstract data types. User-defined abstract data types are discussed in detail in Chapter 11.

There are a number of uses of the type system of a programming language. The most practical of these is error detection. The process and value of type checking, which is directed by the type system of the language, are discussed in Section 6.12. A second use of a type system is the assistance it provides for program modularization. This results from the cross-module type checking that ensures the consistency of the interfaces among modules. Another use of a type system is documentation. The type declarations in a program document information about its data, which provides clues about the program's behavior.

The type system of a programming language defines how a type is associated with each expression in the language and includes its rules for type equivalence and type compatibility. Certainly, one of the most important parts of understanding the semantics of a programming language is understanding its type system.

The two most common structured (nonscalar) data types in the imperative languages are arrays and records, although the popularity of associative arrays has increased significantly in recent years. Lists have been a central part of functional programming languages since the first such language appeared in 1959 (Lisp). Over the last decade, the increasing popularity of functional programming has led to lists being added to primarily imperative languages, such as Python and C#.

The structured data types are defined with type operators, or constructors, which are used to form type expressions. For example, C uses brackets and asterisks as type operators to specify arrays and pointers.

It is convenient, both logically and concretely, to think of variables in terms of descriptors. A **descriptor** is the collection of the attributes of a variable. In an implementation, a descriptor is an area of memory that stores the attributes of a variable. If the attributes are all static, descriptors are required only at

compile time. These descriptors are built by the compiler, usually as a part of the symbol table, and are used during compilation. For dynamic attributes, however, part or all of the descriptor must be maintained during execution. In this case, the descriptor is used by the run-time system. In all cases, descriptors are used for type checking and building the code for the allocation and deallocation operations.

Care must be taken when using the term *variable*. One who uses only traditional imperative languages may think of identifiers as variables, but that can lead to confusion when considering data types. Identifiers do not have data types in some programming languages. It is wise to remember that identifiers are just one of the attributes of a variable.

The word *object* is often associated with the value of a variable and the space it occupies. In this book, however, we reserve *object* exclusively for instances of user-defined and language-defined abstract data types, rather than for the values of all program variables of predefined types. Objects are discussed in detail in Chapters 11 and 12.

In the following sections, many common data types are discussed. For most, design issues particular to the type are stated. For all, one or more example designs are described. One design issue is fundamental to all data types: What operations are provided for variables of the type, and how are they specified?

6.2 Primitive Data Types

Data types that are not defined in terms of other types are called **primitive data types**. Nearly all programming languages provide a set of primitive data types. Some of the primitive types are merely reflections of the hardware—for example, most integer types. Others require only a little nonhardware support for their implementation.

To specify the structured types, the primitive data types of a language are used, along with one or more type constructors.

6.2.1 Numeric Types

Some early programming languages only had numeric primitive types. Numeric types still play a central role among the collections of types supported by contemporary languages.

6.2.1.1 Integer

The most common primitive numeric data type is `integer`. The hardware of many computers supports several sizes of integers. These sizes of integers, and often a few others, are supported by some programming languages. For example, Java includes four signed integer sizes: `byte`, `short`, `int`, and `long`. Some languages, for example, C++ and C#, include unsigned integer types, which are types for integer values without signs. Unsigned types are often used for binary data.

A signed integer value is represented in a computer by a string of bits, with one of the bits (typically the leftmost) representing the sign. Most integer types are supported directly by the hardware. One example of an integer type that is not supported directly by the hardware is the long integer type of Python (F# also provides such integers). Values of this type can have

unlimited length. Long integer values can be specified as literals, as in the following example:

```
243725839182756281923L
```

Integer arithmetic operations in Python that produce values too large to be represented with **`int`** type store them as long integer type values.

A negative integer could be stored in sign-magnitude notation, in which the sign bit is set to indicate negative and the remainder of the bit string represents the absolute value of the number. Sign-magnitude notation, however, does not lend itself to computer arithmetic. Most computers now use a notation called **twos complement** to store negative integers, which is convenient for addition and subtraction. In twos-complement notation, the representation of a negative integer is formed by taking the logical complement of the positive version of the number and adding one. Ones-complement notation is still used by some computers. In ones-complement notation, the negative of an integer is stored as the logical complement of its absolute value. Ones-complement notation has the disadvantage that it has two representations of zero. See any book on assembly language programming for details of integer representations.

6.2.1.2 Floating-Point

Floating-point data types model real numbers, but the representations are only approximations for many real values. For example, neither of the fundamental numbers π or *e* (the base for the natural logarithms) can be correctly represented in floating-point notation. Of course, neither of these numbers can be precisely represented in any finite amount of computer memory. On most computers, floating-point numbers are stored in binary, which exacerbates the problem. For example, even the value 0.1 in decimal cannot be represented by a finite number of binary digits.[1] Another problem with floating-point types is the loss of accuracy through arithmetic operations. For more information on the problems of floating-point notation, see any book on numerical analysis.

1. 0.1 in decimal is 0.0001100110011 . . . in binary.

Floating-point values are represented as fractions and exponents, a form that is borrowed from scientific notation. Older computers used a variety of different representations for floating-point values. However, most newer machines use the IEEE Floating-Point Standard 754 format. Language implementors use whatever representation is supported by the hardware. Most languages include two floating-point types, often called **float** and **double**. The float type is the standard size, usually stored in four bytes of memory. The double type is provided for situations where larger fractional parts and/or a larger range of exponents is needed. Double-precision variables usually occupy twice as much storage as float variables and provide at least twice the number of bits of fraction.

The collection of values that can be represented by a floating-point type is defined in terms of precision and range. **Precision** is the accuracy of the fractional part of a value, measured as the number of bits. **Range** is a combination of the range of fractions and, more important, the range of exponents.

Figure 6.1 shows the IEEE Floating-Point Standard 754 format for single- and double-precision representation (IEEE, 1985). Details of the IEEE formats can be found in Tanenbaum (2005).

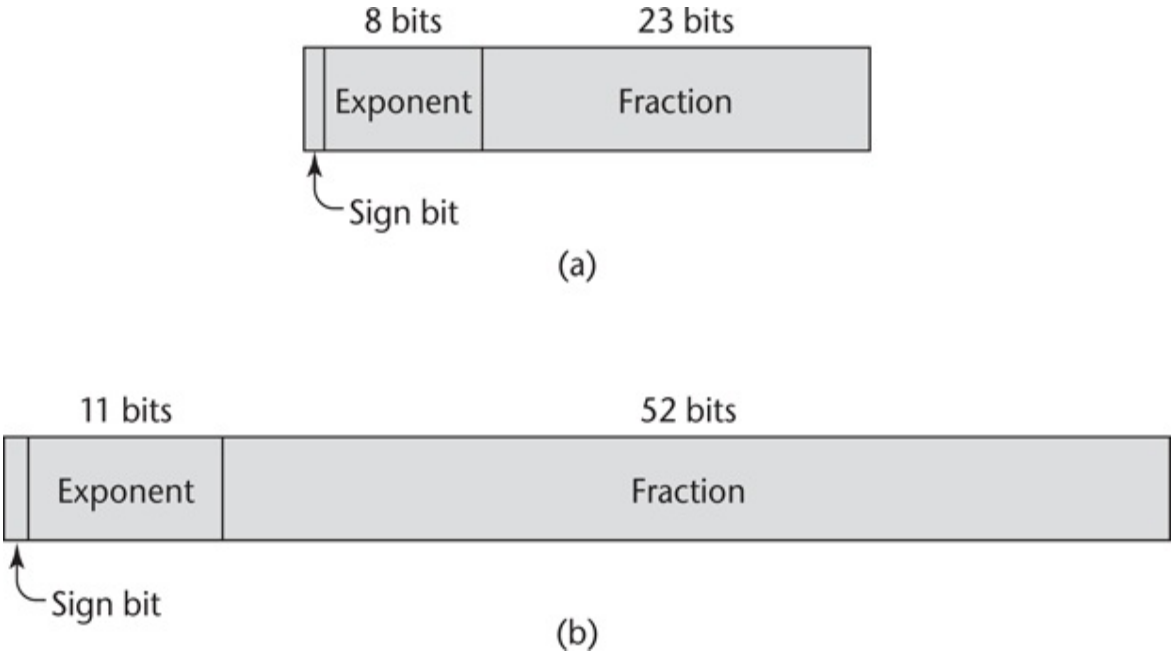


Figure 6.1 IEEE floating-point formats: (a) single precision, (b) double precision

Figure 6.1 Full Alternative Text

6.2.1.3 Complex

Some programming languages support a complex data type—for example, Fortran and Python. Complex values are represented as ordered pairs of floating-point values. In Python, the imaginary part of a complex literal is specified by following it with a `j` or `J`—for example,

```
(7 + 3j)
```

Languages that support a complex type include operations for arithmetic on complex values.

6.2.1.4 Decimal

Most larger computers that are designed to support business systems applications have hardware support for **decimal** data types. Decimal data types store a fixed number of decimal digits, with the implied decimal point at a fixed position in the value. These are the primary data types for business data processing and are therefore essential to COBOL. C# and F# also have decimal data types.

Decimal types have the advantage of being able to precisely store decimal values, at least those within a restricted range, which cannot be done with -floating-point. For example, the number 0.1 (in decimal) can be exactly represented in a decimal type, but not in a floating-point type, as is noted in

Section 6.2.1.2. The disadvantages of decimal types are that the range of values is restricted because no exponents are allowed, and their representation in memory is mildly wasteful, for reasons discussed in the following paragraph.

Decimal types are stored very much like character strings, using binary codes for the decimal digits. These representations are called **binary coded decimal (BCD)**. In some cases, they are stored one digit per byte, but in others, they are packed two digits per byte. Either way, they take more storage than binary representations. It takes at least four bits to code a decimal digit. Therefore, to store a six-digit coded decimal number requires 24 bits of memory. However, it takes only 20 bits to store the same number in binary.2 The operations on decimal values are done in hardware on machines that have such capabilities; otherwise, they are simulated in software.

2. Of course, unless a program needs to maintain a large number of large decimal values, the difference is insignificant.

6.2.2 Boolean Types

Boolean types are perhaps the simplest of all types. Their range of values has only two elements: one for true and one for false. They were introduced in ALGOL 60 and have been included in most general-purpose languages designed since 1960. One popular exception is C89, in which numeric expressions are used as conditionals. In such expressions, all operands with nonzero values are considered true, and zero is considered false. Although C99 and C++ have a Boolean type, they also allow numeric expressions to be used as if they were Boolean. This is not the case in the subsequent languages, Java and C#.

Boolean types are often used to represent switches or flags in programs. Although other types, such as integers, can be used for these purposes, the use of Boolean types is more readable.

A Boolean value could be represented by a single bit, but because a single bit of memory cannot be accessed efficiently on many machines, they are often

stored in the smallest efficiently addressable cell of memory, typically a byte.

6.2.3 Character Types

Character data are stored in computers as numeric codings. Traditionally, the most commonly used coding was the 8-bit code ASCII (American Standard Code for Information Interchange), which uses the values 0 to 127 to code 128 different characters. ISO 8859-1 is another 8-bit character code, but it allows 256 different characters.

Because of the globalization of business and the need for computers to communicate with other computers around the world, the ASCII character set became inadequate. In response, in 1991, the Unicode Consortium published the UCS-2 standard, a 16-bit character set. This character code is often called Unicode. Unicode includes the characters from most of the world's natural languages. For example, Unicode includes the Cyrillic alphabet, as used in Serbia, and the Thai digits. The first 128 characters of Unicode are identical to those of ASCII. Java was the first widely used language to use the Unicode character set. Since then, it has found its way into JavaScript, Python, Perl, C#, F#, and Swift.

After 1991, the Unicode Consortium, in cooperation with the International Standards Organization (ISO), developed a 4-byte character code named UCS-4, or UTF-32, which is described in the ISO/IEC 10646 Standard, published in 2000.

To provide the means of processing codings of single characters, most programming languages include a primitive type for them. However, Python supports single characters only as character strings of length 1.

6.3 Character String Types

A **character string type** is one in which the values consist of sequences of characters. Character string constants are used to label output, and the input and output of all kinds of data are often done in terms of strings. Of course, character strings also are an essential type for all programs that do character manipulation.

6.3.1 Design Issues

The two most important design issues that are specific to character string types are the following:

- Should strings be a special kind of character array or a primitive type?
- Should strings have static or dynamic length?

6.3.2 Strings and Their Operations

The most common string operations are assignment, catenation, substring reference, comparison, and pattern matching.

A **substring reference** is a reference to a substring of a given string. Substring references are discussed in the more general context of arrays, where the substring references are called **slices**.

In general, both assignment and comparison operations on character strings are complicated by the possibility of string operands of different lengths. For example, what happens when a longer string is assigned to a shorter string, or vice versa? Usually, simple and sensible choices are made for these situations, although programmers often have trouble remembering them.

In some languages, pattern matching is supported directly in the language. In others, it is provided by a function or class library.

If strings are not defined as a primitive type, string data is usually stored in arrays of single characters and referenced as such in the language. This is the approach taken by C and C++, which use **`char`** arrays to store character strings. These languages provide a collection of string operations through standard libraries. Many users of strings and many of the library functions use the convention that character strings are terminated with a special character, null, which is represented with zero. This is an alternative to maintaining the length of string variables. The library operations simply carry out their operations until the null character appears in the string being operated on. Library functions that produce strings often supply the null character. The character string literals that are built by the compiler also have the null character. For example, consider the following declaration:

```
char str[] = "apples";
```

In this example, `str` represents an array of **`char`** elements, specifically `apples0`, where `0` is the null character.

Some of the most commonly used library functions for character strings in C and C++ are `strcpy`, which moves strings; `strcat`, which catenates one given string onto another; `strcmp`, which lexicographically compares (by the order of their character codes) two given strings; and `strlen`, which returns the number of characters, not counting the null character, in the given string. The parameters and return values for most of the string manipulation functions are **`char`** pointers that point to arrays of **`char`**. Parameters can also be string literals.

The string manipulation functions of the C standard library, which are also available in C++, are inherently unsafe and have led to numerous programming errors. The problem is that the functions in this library that move string data do not guard against overflowing the destination. For example, consider the following call to `strcpy`:

```
strcpy(dest, src);
```

If the length of `dest` is 20 and the length of `src` is 50, `strcpy` will write over the 30 bytes that follow `dest`. The point is that `strcpy` does not know the length of `dest`, so it cannot ensure that the memory following it will not be overwritten. The same problem can occur with several of the other functions in the C string library. In addition to C-style strings, C++ also supports strings through its standard class library, which is also similar to that of Java. Because of the insecurities of the C string library, C++ programmers should use the `string` class from the standard library, rather than **`char`** arrays and the C string library.

In Java, strings are supported by the `String` class, whose values are constant strings, and the `StringBuffer` class, whose values are changeable and are more like arrays of single characters. These values are specified with methods of the `StringBuffer` class. C# and Ruby include string classes that are similar to those of Java.

Python includes strings as a primitive type and has operations for substring reference, catenation, indexing to access individual characters, as well as methods for searching and replacement. There is also an operation for character membership in a string. So, even though Python's strings are primitive types, for character and substring references, they act very much like arrays of characters. However, Python strings are immutable, similar to the `String` class objects of Java.

In F#, strings are a class. Individual characters, which are represented in Unicode UTF-16, can be accessed, but not changed. Strings can be catenated with the `+` operator. In ML, string is a primitive immutable type. It uses `^` for its catenation operator and includes functions for substring referencing and getting the size of a string.

In Swift, the `String` class supports its character strings. `String` objects can be either constants or variables. The binary `+` operator catenates `String` variables. The `append` method is used to add a `Character` object to a `String` object. The `characters` method of `String` is used to examine individual characters of a `String` object.

history note

SNOBOL 4 was the first widely known language to support pattern matching.

Perl, JavaScript, Ruby, and PHP include built-in pattern-matching operations. In these languages, the pattern-matching expressions are somewhat loosely based on mathematical regular expressions. In fact, they are often called **regular expressions**. They evolved from the early UNIX line editor, `ed`, to become part of the UNIX shell languages. Eventually, they grew to their current complex form. There is at least one complete book on this kind of -pattern-matching expressions (Friedl, 2006). In this section, we provide a brief look at the style of these expressions through two relatively simple examples.

Consider the following pattern expression:

```
/[A-Za-z][A-Za-z\d]+/
```

This pattern matches (or describes) the typical name form in programming languages. The brackets enclose character classes. The first character class specifies all letters; the second specifies all letters and digits (a digit is specified with the abbreviation `\d`). If only the second character class were included, we could not prevent a name from beginning with a digit. The plus operator following the second category specifies that there must be one or more of what is in the category. So, the whole pattern matches strings that begin with a letter, followed by one or more letters or digits.

Next, consider the following pattern expression:

```
/\d+\.?\d*|\.\d+/
```

This pattern matches numeric literals. The `\.` specifies a literal decimal point.3 The question mark quantifies what it follows to have zero or one appearance. The vertical bar (`|`) separates two alternatives in the whole pattern. The first alternative matches strings of one or more digits, possibly followed by a decimal point, followed by zero or more digits; the second

alternative matches strings that begin with a decimal point, followed by one or more digits.

3. The period must be “escaped” with the backslash because period has special meaning in a regular expression.

Pattern-matching capabilities using regular expressions are included in the class libraries of C++, Java, Python, C#, and F#.

6.3.3 String Length Options

There are several design choices regarding the length of string values. First, the length can be static and set when the string is created. Such a string is called a **static length string**. This is the choice for the strings of Python, the immutable objects of Java’s `String` class, as well as similar classes in the C++ standard class library, Ruby’s built-in `String` class, and the .NET class library available to C# and F#.

The second option is to allow strings to have varying length up to a declared and fixed maximum set by the variable’s definition, as exemplified by the strings in C and the C-style strings of C++. These are called **limited dynamic length strings**. Such string variables can store any number of characters between zero and the maximum. Recall that strings in C use a special character to indicate the end of the string’s characters, rather than maintaining the string length.

The third option is to allow strings to have varying length with no maximum, as in JavaScript, Perl, and the standard C++ library. These are called **dynamic length strings**. This option requires the overhead of dynamic storage allocation and deallocation but provides maximum flexibility.

6.3.4 Evaluation

String types are important to the writability of a language. Dealing with strings as arrays can be more cumbersome than dealing with a primitive

string type. For example, consider a language that treats strings as arrays of characters and does not have a predefined function that does what `strcpy` in C does. Then, a simple assignment of one string to another would require a loop. The addition of strings as a primitive type to a language is not costly in terms of either language or compiler complexity. Therefore, it is difficult to justify the omission of primitive string types in some contemporary languages. Of course, providing strings through a standard library is nearly as convenient as having them as a primitive type.

String operations such as simple pattern matching and catenation are essential and should be included for string type values. Although dynamic length strings are obviously the most flexible, the overhead of their implementation must be weighed against that additional flexibility.

6.3.5 Implementation of Character String Types

Character string types could be supported directly in hardware; but in most cases, software is used to implement string storage, retrieval, and manipulation. When character string types are represented as character arrays, the language often supplies few operations.

A descriptor for a static character string type, which is required only during compilation, has three fields. The first field of every descriptor is the name of the type. In the case of static character strings, the second field is the type's length (in characters). The third field is the address of the first character. This descriptor is shown in Figure 6.2. Limited dynamic strings require a run-time descriptor to store the fixed maximum length, the current length, and the address, as shown in Figure 6.3. Dynamic length strings require a simpler run-time descriptor because only the current length and the address need to be stored. Although we depict descriptors as independent blocks of storage, in most cases, they are stored in the symbol table.

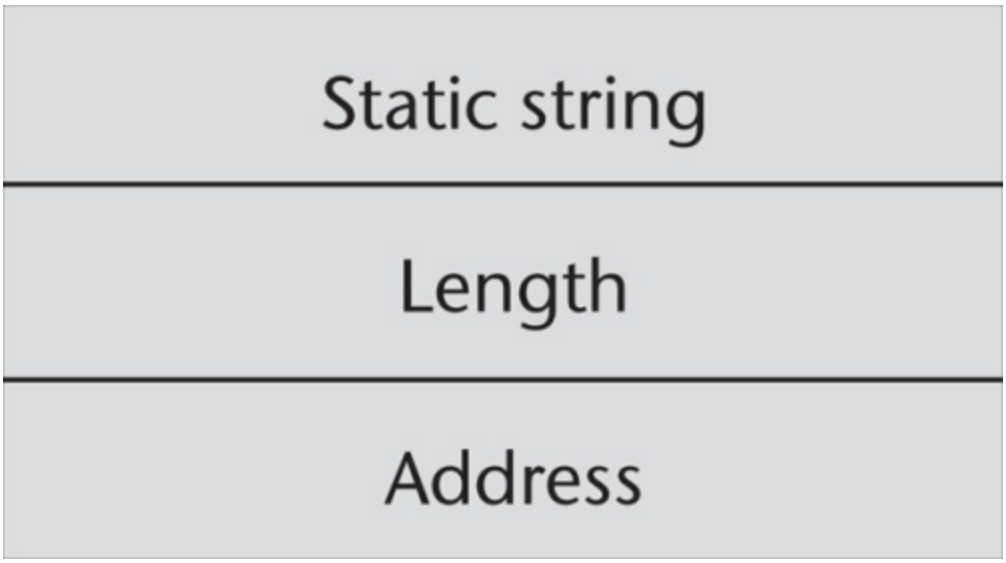


Figure 6.2 Compile-time descriptor for static strings

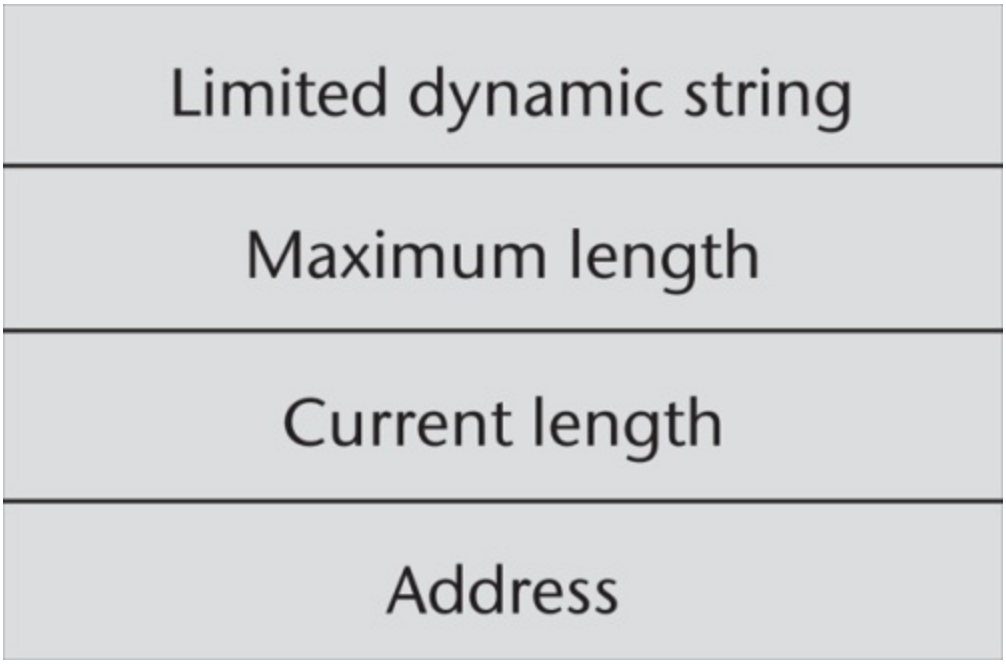


Figure 6.3 Run-time descriptor

for limited dynamic strings

The limited dynamic strings of C and C++ do not require run-time descriptors, because the end of a string is marked with the null character. They do not need the maximum length, because index values in array references are not range checked in these languages.

Static length and limited dynamic length strings require no special dynamic storage allocation. In the case of limited dynamic length strings, sufficient storage for the maximum length is allocated when the string variable is bound to storage, so only a single allocation process is involved.

Dynamic length strings require more complex storage management. The length of a string, and therefore the storage to which it is bound, must grow and shrink dynamically.

There are three approaches to supporting the dynamic allocation and deallocation that is required for dynamic length strings. First, strings can be stored in a linked list, so that when a string grows, the newly required cells can come from anywhere in the heap. The drawbacks to this method are the extra storage occupied by the links in the list representation and the necessary complexity of string operations.

The second approach is to store strings as arrays of pointers to individual characters allocated in the heap. This method still uses extra memory, but string processing can be faster than with the linked-list approach.

The third alternative is to store complete strings in adjacent storage cells. The problem with this method arises when a string grows: How can storage that is adjacent to the existing cells continue to be allocated for the string variable? Frequently, such storage is not available. Instead, a new area of memory is found that can store the complete new string, and the old part is moved to this area. Then, the memory cells used for the old string are deallocated. This latter approach is the one typically used. The general problem of managing allocation and deallocation of variable-size segments is discussed in Section 6.11.7.3.

Although the linked-list method requires more storage, the associated allocation and deallocation processes are simple. However, some string operations are slowed by the required pointer chasing. On the other hand, using adjacent memory for complete strings results in faster string operations and requires significantly less storage, but the allocation and deallocation processes are slower.

6.4 Enumeration Types

An **enumeration type** is one in which all of the possible values, which are named constants, are provided, or enumerated, in the definition. Enumeration types provide a way of defining and grouping collections of named constants, which are called **enumeration constants**. The definition of a typical enumeration type is shown in the following C# example:

```
enum  days {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
```

The enumeration constants are typically implicitly assigned the integer values, 0, 1, . . . but can be explicitly assigned any integer literal in the type's definition.

6.4.1 Design Issues

The design issues for enumeration types are as follows:

- Is an enumeration constant allowed to appear in more than one type definition, and if so, how is the type of an occurrence of that constant in the program checked?
- Are enumeration values coerced to integer?
- Are any other types coerced to an enumeration type?

All of these design issues are related to type checking. If an enumeration variable is coerced to a numeric type, then there is little control over its range of legal operations or its range of values. If an `int` type value is coerced to an enumeration type, then an enumeration type variable could be assigned any integer value, whether it represented an enumeration constant or not.

6.4.2 Designs

In languages that do not have enumeration types, programmers usually simulate them with integer values. For example, suppose we needed to represent colors in a C program and C did not have an enumeration type. We might use 0 to represent blue, 1 to represent red, and so forth. These values could be defined as follows:

```
int    red = 0, blue = 1;
```

Now, in the program, we could use `red` and `blue` as if they were of a color type. The problem with this approach is that because we have not defined a type for our colors, there is no type checking when they are used. For example, it would be legal to add the two together, although that would rarely be an intended operation. They could also be combined with any other numeric type operand using any arithmetic operator, which would also rarely be useful. Furthermore, because they are just variables, they could be assigned any integer value, thereby destroying the relationship with the colors. This latter problem could be prevented by making them named constants.

C and Pascal were the first widely used languages to include an enumeration data type. C++ includes C's enumeration types. In C++, we could have the following:

```
enum  colors {red, blue, green, yellow, black};
colors myColor = blue, yourColor = red;
```

The `colors` type uses the default internal values for the enumeration constants, 0, 1, . . . , although the constants could have been specifically assigned any integer literal (or any constant-valued expression) by the programmer. The enumeration values are coerced to **`int`** when they are put in integer context. This allows their use in any numeric expression. For example, if the current value of `myColor` is `blue`, then the expression

```
myColor++
```

would assign the integer code for `green` to `myColor`.

C++ also allows enumeration constants to be assigned to variables of any numeric type, though that would likely be an error. However, no other type

value is coerced to an enumeration type in C++. For example,

```
myColor = 4;
```

is illegal in C++. This assignment would be legal if the right side had been cast to `colors` type. This prevents some potential errors.

C++ enumeration constants can appear in only one enumeration type in the same referencing environment.

In 2004, an enumeration type was added to Java in Java 5.0. All enumeration types in Java are implicitly subclasses of the predefined class `Enum`. Because enumeration types are classes, they can have instance data fields, constructors, and methods. Syntactically, Java enumeration type definitions appear like those of C++, except that they can include fields, constructors, and methods. The possible values of an enumeration are the only possible instances of the class. All enumeration types inherit `toString`, as well as a few other methods. An array of the instances of an enumeration type can be fetched with the static method `values`. The internal numeric value of an enumeration variable can be fetched with the `ordinal` method. No expression of any other type can be assigned to an enumeration variable. Also, an enumeration variable is never coerced to any other type.

C# enumeration types are like those of C++, except that they are never coerced to integer. So, operations on enumeration types are restricted to those that make sense. Also, the range of values is restricted to that of the particular enumeration type.

In ML, enumeration types are defined as new types with **`datatype`** declarations. For example, we could have the following:

```
datatype weekdays = Monday | Tuesday | Wednesday |
Thursday | Friday
```

The type of the elements of `weekdays` is integer.

F# has enumeration types that are similar to those of ML, except the reserved word **`type`** is used instead of **`datatype`** and the first value is preceded by an OR operator (`|`).

Swift has an enumeration type in which the enumeration values are names, which represent themselves, rather than having internal integer values. An enumeration type is defined in a structure that is similar to a switch structure, as in:

```
enum fruit {
  case orange
  case apple
  case banana
}
```

Dot notation is used to reference enumeration values, so in our example, the value of `apple` is referenced as `fruit.apple`.

Interestingly, none of the relatively recent scripting languages include enumeration types. These include Perl, JavaScript, PHP, Python, and Ruby. Even Java was a decade old before enumeration types were added.

6.4.3 Evaluation

Enumeration types can provide advantages in both readability and reliability. Readability is enhanced very directly: Named values are easily recognized, whereas coded values are not.

In the area of reliability, the enumeration types of C#, F#, Java 5.0, and Swift provide two advantages: (1) No arithmetic operations are legal on enumeration types; this prevents adding days of the week, for example, and (2) second, no enumeration variable can be assigned a value outside its defined range.4 If the `colors` enumeration type has 10 enumeration constants and uses `0..9` as its internal values, no number greater than `9` can be assigned to a `colors` type variable.

4. In C# and F#, an integer value can be cast to an enumeration type and assigned to the name of an enumeration variable. Such values must be tested with `Enum.IsDefined` method before assigning them to the name of an enumeration variable.

Because C treats enumeration variables like integer variables, it does not

provide either of these two advantages.

C++ is a little better. Numeric values can be assigned to enumeration type variables only if they are cast to the type of the assigned variable. Numeric values assigned to enumeration type variables are checked to determine whether they are in the range of the internal values of the enumeration type. Unfortunately, if the user uses a wide range of explicitly assigned values, this checking is not effective. For example,

```
enum  colors {red = 1, blue = 1000, green = 100000}
```

In this example, a value assigned to a variable of `colors` type will only be checked to determine whether it is in the range of `1..100000`.

6.5 Array Types

An **array** is a homogeneous aggregate of data elements in which an individual element is identified by its position in the aggregate, relative to the first element. The individual data elements of an array are of the same type. References to individual array elements are specified using subscript expressions. If any of the subscript expressions in a reference include variables, then the reference will require an additional run-time calculation to determine the address of the memory location being referenced.

In many languages, such as C, C++, Java, and C#, all of the elements of an array are required to be of the same type. In these languages, pointers and references are restricted to point to or reference a single type. So the objects or data values being pointed to or referenced are also of a single type. In some other languages, such as JavaScript, Python, and Ruby, variables are typeless references to objects or data values. In these cases, arrays still consist of elements of a single type, but the elements can reference objects or data values of different types. Such arrays are still homogeneous, because the array elements are of the same type. In Swift, arrays can be typed, that is, they will contain values only of a single type, or untyped, which means they can contain values of any type.

C# and Java 5.0 provide generic arrays, that is, arrays whose elements are references to objects, through their class libraries. These are discussed in Section 6.5.3.

6.5.1 Design Issues

The primary design issues specific to arrays are the following:

- What types are legal for subscripts?
- Are subscripting expressions in element references range checked?

- When are subscript ranges bound?
- When does array allocation take place?
- Are ragged or rectangular multidimensioned arrays allowed, or both?
- Can arrays be initialized when they have their storage allocated?
- What kinds of slices are allowed, if any?

In the following sections, examples of the design choices made for the arrays of the most common programming languages are discussed.

6.5.2 Arrays and Indices

Specific elements of an array are referenced by means of a two-level syntactic mechanism, where the first part is the aggregate name, and the second part is a possibly dynamic selector consisting of one or more items known as **subscripts** or **indices**. If all of the subscripts in a reference are constants, the selector is static; otherwise, it is dynamic. The selection operation can be thought of as a mapping from the array name and the set of subscript values to an element in the aggregate. Indeed, arrays are sometimes called **finite mappings**. Symbolically, this mapping can be shown as

history note

The designers of pre-90 Fortrans and PL/I chose parentheses for array subscripts because no other suitable characters were available at the time. Card punches did not include bracket characters.

array_name(subscript_value_list) → element

The syntax of array references is fairly universal: The array name is followed by the list of subscripts, which is surrounded by either parentheses or brackets. In some languages that provide multidimensioned arrays as arrays

of arrays, each subscript appears in its own brackets. A problem with using parentheses to enclose subscript expressions is that they often are also used to enclose the parameters in subprogram calls; this use makes references to arrays appear exactly like those calls. For example, consider the following Ada assignment statement:

```
Sum := Sum + B(I);
```

Because parentheses are used for both subprogram parameters and array subscripts in Ada, both program readers and compilers are forced to use other information to determine whether `B(I)` in this assignment is a function call or a reference to an array element. This results in reduced readability.

history note

Fortran I limited the number of array subscripts to three, because at the time of the design, execution efficiency was a primary concern. Fortran I designers had developed a very fast method for accessing the elements of arrays of up to three dimensions, using the three index registers of the IBM 704. Fortran IV was first implemented on an IBM 7094, which had seven index registers. This allowed Fortran IV's designers to allow arrays with up to seven subscripts. Most other contemporary languages enforce no such limits.

The designers of Ada specifically chose parentheses to enclose subscripts so there would be uniformity between array references and function calls in expressions, in spite of potential readability problems. They made this choice in part because both array element references and function calls are mappings. Array element references map the subscripts to a particular element of the array. Function calls map the actual parameters to the function definition and, eventually, a functional value.

Most languages other than Fortran and Ada use brackets to delimit their array indices.

Two distinct types are involved in an array type: the element type and the type of the subscripts. The type of the subscripts is often integer.

Early programming languages did not specify that subscript ranges must be implicitly checked. Range errors in subscripts are common in programs, so requiring range checking is an important factor in the reliability of languages. Many contemporary languages also do not specify range checking of subscripts, but Java, ML, and C# do.

Subscripting in Perl is a bit unusual in that although the names of all arrays begin with at signs (@), because array elements are always scalars and the names of scalars always begin with dollar signs ($), references to array elements use dollar signs rather than at signs in their names. For example, for the array `@list`, the second element is referenced with `$list`[1].

One can reference an array element in Perl with a negative subscript, in which case the subscript value is an offset from the end of the array. For example, if the array `@list` has five elements with the subscripts 0..4, `$list` [−2] references the element with the subscript 3. A reference to a nonexistent element in Perl yields **`undef`**, but no error is reported.

6.5.3 Subscript Bindings and Array Categories

The binding of the subscript type to an array variable is usually static, but the subscript value ranges are sometimes dynamically bound.

In some languages, the lower bound of the subscript range is implicit. For example, in the C-based languages, the lower bound of all subscript ranges is fixed at `0`. In some other languages, the lower bounds of the subscript ranges must be specified by the programmer.

There are four categories of arrays, based on the binding to subscript ranges, the binding to storage, and from where the storage is allocated. The category names indicate the design choices of these three. In the first three of these categories, once the subscript ranges are bound and the storage is allocated, they remain fixed for the lifetime of the variable. Of course, when the subscript ranges are fixed, the array cannot change size.

A **static array** is one in which the subscript ranges are statically bound and storage allocation is static (done before run time). The advantage of static arrays is efficiency: No dynamic allocation or deallocation is required. The disadvantage is that the storage for the array is fixed for the entire execution time of the program.

A **fixed stack-dynamic array** is one in which the subscript ranges are statically bound, but the allocation is done at declaration elaboration time during execution. The advantage of fixed stack-dynamic arrays over static arrays is space efficiency. A large array in one subprogram can use the same space as a large array in a different subprogram, as long as both subprograms are not active at the same time. The same is true if the two arrays are in different blocks that are not active at the same time. The disadvantage is the required allocation and deallocation time.

A **fixed heap-dynamic array** is similar to a fixed stack-dynamic array, in that the subscript ranges and the storage binding are both fixed after storage is allocated. The differences are that both the subscript ranges and storage bindings are done when the user program requests them during execution, and the storage is allocated from the heap, rather than the stack. The advantage of fixed heap-dynamic arrays is flexibility—the array's size always fits the problem. The disadvantage is allocation time from the heap, which is longer than allocation time from the stack.

A **heap-dynamic array** is one in which the binding of subscript ranges and storage allocation is dynamic and can change any number of times during the array's lifetime. The advantage of heap-dynamic arrays over the others is flexibility: Arrays can grow and shrink during program execution as the need for space changes. The disadvantage is that allocation and deallocation take longer and may happen many times during execution of the program. Examples of the four categories are given in the following paragraphs.

Arrays declared in C and C++ functions that include the **`static`** modifier are static.

Arrays that are declared in C and C++ functions without the **`static`** specifier are examples of fixed stack-dynamic arrays.

C and C++ also provide fixed heap-dynamic arrays. The standard C library functions `malloc` and `free`, which are general heap allocation and deallocation operations, respectively, can be used for C arrays. C++ uses the operators **`new`** and **`delete`** to manage heap storage. An array is treated as a pointer to a collection of storage cells, where the pointer can be indexed, as discussed in Section 6.11.5.

In Java, all non-generic arrays are fixed heap-dynamic. Once created, these arrays keep the same subscript ranges and storage. C# also provides fixed heap-dynamic arrays.

Objects of the C# `List` class are generic heap-dynamic arrays. These array objects are created without any elements, as in

```
List<String> stringList =  new  List<String>();
```

Elements are added to this object with the `Add` method, as in

```
stringList.Add("Michael");
```

Access to elements of these arrays is through subscripting.

Java includes a generic class similar to C#'s `List`, named `ArrayList`. It is different from C#'s `List` in that subscripting is not supported—`get` and `set` methods must be used to access the elements.

A Perl array can be made to grow by using the `push` (puts one or more new elements on the end of the array) and `unshift` (puts one or more new elements on the beginning of the array), or by assigning a value to the array specifying a subscript beyond the highest current subscript of the array. An array can be made to shrink to no elements by assigning it the empty list, `()`. The length of an array is defined to be the largest subscript plus one.

Like Perl, JavaScript allows arrays to grow with the `push` and `unshift` methods and shrink by setting them to the empty list. However, negative subscripts are not supported.

JavaScript arrays can be sparse, meaning the subscript values need not be contiguous. For example, suppose we have an array named list that has 10

elements with the subscripts 0..9.[5] Consider the following assignment statement:

5. The subscript range could just as easily have been 1000 .. 1009.

```
list[50] = 42;
```

Now, `list` has 11 elements and length 51. The elements with subscripts `11..49` are not defined and therefore do not require storage. A reference to a nonexistent element in a JavaScript array yields **`undefined`**.

Arrays in Python and Ruby can be made to grow only through methods to add elements or catenate other arrays. Ruby and Perl support negative subscripts, but Python does not. In Python an element or slice of an array can be deleted. A reference to a nonexistent element in Python results in a run-time error, whereas a similar reference in Ruby yields **`nil`** and no error is reported.

Swift dynamic arrays are objects that use integer subscripts, beginning at zero, and include several useful methods. The `append` method adds an element to the end of an array. The `insert` method inserts a new element at any position in the array, but results in an error if the insertion is at a subscript beyond the current length of the array. Elements can be removed from an array with the `removeAtIndex` method. There are also `reverse` and `count` methods.

Although the ML definition does not include arrays, its widely used implementation, SML/NJ, does.

The only predefined collection type that is part of F# is the array (other collection types are provided through the .NET Framework Library). These arrays are like those of C#. A **`foreach`** statement is included in the language for array processing.

6.5.4 Array Initialization

Some languages provide the means to initialize arrays at the time their

storage is allocated. C, C++, Java, Swift, and C# allow initialization of their arrays. Consider the following C declaration:

```
int  list [] = {4, 5, 7, 83};
```

The array `list` is created and initialized with the values `4`, `5`, `7`, and `83`. The compiler also sets the length of the array. This is meant to be a convenience but is not without cost. It effectively removes the possibility that the system could detect some kinds of programmer errors, such as mistakenly leaving a value out of the list.

As discussed in Section 6.3.2, character strings in C and C++ are implemented as arrays of **`char`**. These arrays can be initialized to string constants, as in

```
char  name [] = "freddie";
```

The array `name` will have eight elements, because all strings are terminated with a null character (zero), which is implicitly supplied by the system for string constants.

Arrays of strings in C and C++ can also be initialized with string literals. For example,

```
char  *names [] = {"Bob", "Jake", "Darcie"};
```

This example illustrates the nature of character literals in C and C++. In the previous example of a string literal being used to initialize the **`char`** array `name`, the literal is taken to be a **`char`** array. But in the latter example (`names`), the literals are taken to be pointers to characters, so the array is an array of pointers to characters. For example, `names[0]` is a pointer to the letter `'B'` in the literal character array that contains the characters `'B'`, `'o'`, `'b'`, and the null character.

In Java, similar syntax is used to define and initialize an array of references to `String` objects. For example,

```
String[] names = ["Bob", "Jake", "Darcie"];
```

6.5.5 Array Operations

An array operation is one that operates on an array as a unit. The most common array operations are assignment, catenation, comparison for equality and inequality, and slices, which are discussed separately in Section 6.5.5.

The C-based languages do not provide any array operations, except through the methods of Java, C++, and C#. Perl supports array assignments but does not support comparisons.

Python's arrays are called lists, although they have all the characteristics of dynamic arrays. Because the objects can be of any types, these arrays are heterogeneous. Python provides array assignment, although it is only a reference change. Python also has operations for array catenation (+) and element membership (**`in`**). It includes two different comparison operators: one that determines whether the two variables reference the same object (**`is`**) and one that compares all corresponding objects in the referenced objects, regardless of how deeply they are nested, for equality (==).

Like Python, the elements of Ruby's arrays are references to objects. And like Python, when a == operator is used between two arrays, the result is true only if the two arrays have the same length and the corresponding elements are equal. Ruby's arrays can be catenated with an `Array` method.

F# includes many array operators in its `Array` module. Among these are `Array.append`, `Array.copy`, and `Array.length`.

Arrays and their operations are the heart of APL; it is the most powerful array-processing language ever devised. Because of its relative obscurity and its lack of effect on subsequent languages, however, we present here only a glimpse into its array operations.

In APL, the four basic arithmetic operations are defined for vectors (single-dimensioned arrays) and matrices, as well as scalar operands. For example,

```
A + B
```

is a valid expression, whether A and B are scalar variables, vectors, or matrices.

APL includes a collection of unary operators for vectors and matrices, some of which are as follows (where V is a vector and M is a matrix):

⌽V	reverses the elements of V
⌽M	reverses the columns of M
⊖M	reverses the rows of M
⍉M	transposes M (its rows become its columns and vice versa)
⌹M	inverts M

APL also includes several special operators that take other operators as operands. One of these is the inner product operator, which is specified with a period (.). It takes two operands, which are binary operators. For example,

```
+.×
```

is a new operator that takes two arguments, either vectors or matrices. It first multiplies the corresponding elements of two arguments, and then it sums the results. For example, if A and B are vectors,

```
A × B
```

is the mathematical inner product of A and B (a vector of the products of the corresponding elements of A and B). The statement

```
A +.× B
```

is the sum of the inner product of A and B. If A and B are matrices, this expression specifies the matrix multiplication of A and B.

The special operators of APL are actually functional forms, which are described in Chapter 15.

6.5.6 Rectangular and Jagged

Arrays

A **rectangular array** is a multidimensioned array in which all of the rows have the same number of elements and all of the columns have the same number of elements. Rectangular arrays model rectangular tables exactly.

A **jagged array** is one in which the lengths of the rows need not be the same. For example, a jagged matrix may consist of three rows, one with 5 elements, one with 7 elements, and one with 12 elements. This also applies to the columns and higher dimensions. So, if there is a third dimension (layers), each layer can have a different number of elements. Jagged arrays are made possible when multidimensioned arrays are actually arrays of arrays. For example, a matrix would appear as an array of single-dimensioned arrays.

C, C++, and Java support jagged arrays but not rectangular arrays. In those languages, a reference to an element of a multidimensioned array uses a separate pair of brackets for each dimension. For example,

```
myArray[3][7]
```

C# and F# support both rectangular and jagged arrays. For rectangular arrays, all subscript expressions in references to elements are placed in a single pair of brackets. For example,

```
myArray[3, 7]
```

6.5.7 Slices

A **slice** of an array is some substructure of that array. For example, if `A` is a matrix, then the first row of `A` is one possible slice, as are the last row and the first column. It is important to realize that a slice is not a new data type. Rather, it is a mechanism for referencing part of an array as a unit. If arrays cannot be manipulated as units in a language, that language has no use for slices.

Consider the following Python declarations:

```
vector = [2, 4, 6, 8, 10, 12, 14, 16]
mat = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]
```

Recall that the default lower bound for Python arrays is 0. The syntax of a Python slice reference is a pair of numeric expressions separated by a colon. The first is the first subscript of the slice; the second is the first subscript after the last subscript in the slice. Therefore, `vector[3:6]` is a three-element array with the fourth through sixth elements of `vector` (those elements with the subscripts `3`, `4`, and `5`). A row of a matrix is specified by giving just one subscript. For example, `mat[1]` refers to the second row of `mat`; a part of a row can be specified with the same syntax as a part of a single-dimensioned array. For example, `mat[0][0:2]` refers to the first and second element of the first row of mat, which is `[1, 2]`.

Python also supports more complex slices of arrays. For example, `vector[0:7:2]` references every other element of `vector`, up to but not including the element with the subscript 7, starting with the subscript `0`, which is `[2, 6, 10, 14]`.

Perl supports slices of two forms, a list of specific subscripts or a range of subscripts. For example,

```
@list[1..5] = @list2[3, 5, 7, 9, 13];
```

Notice that slice references use array names, not scalar names, because slices are arrays (not scalars).

Ruby supports slices with the `slice` method of its `Array` object, which can take three forms of parameters. A single integer expression parameter is interpreted as a subscript, in which case `slice` returns the element with the given subscript. If `slice` is given two integer expression parameters, the first is interpreted as a beginning subscript and the second is interpreted as the number of elements in the slice. For example, suppose `list` is defined as follows:

```
list = [2, 4, 6, 8, 10]
```

`list.slice(2, 2)` returns `[6, 8]`. The third parameter form for `slice` is a range, which has the form of an integer expression, two periods, and a second

integer expression. With a range parameter, `slice` returns an array of the element with the given range of subscripts. For example, `list.slice (1..3)` returns `[4, 6, 8]`.

6.5.8 Evaluation

Arrays have been included in virtually all programming languages. The primary advances since their introduction in Fortran I have been slices and dynamic arrays. As discussed in Section 6.6, the latest advances in arrays have been in associative arrays.

6.5.9 Implementation of Array Types

Implementing arrays requires considerably more compile-time effort than does implementing primitive types. The code to allow accessing of array elements must be generated at compile time. At run time, this code must be executed to produce element addresses. There is no way to precompute the address to be accessed by a reference such as

```
list[k]
```

A single-dimensioned array is implemented as a list of adjacent memory cells. Suppose the array `list` is defined to have a subscript range lower bound of `0`. The access function for `list` is often of the form

- address (`list[k]`) = address (`list[0]`) + `k` * element_size

where the first operand of the addition is the constant part of the access function, and the second is the variable part.

If the element type is statically bound and the array is statically bound to storage, then the value of the constant part can be computed before run time. However, the addition and multiplication operations must be done at run

time.

The generalization of this access function for an arbitrary lower bound is

- address (`list[k]`) = address (`list`[lower_bound]) +
- ((`k` −lower_bound) * element_size)

The compile-time descriptor for single-dimensioned arrays can have the form shown in Figure 6.4. The descriptor includes information required to construct the access function. If run-time checking of index ranges is not done and the attributes are all static, then only the access function is required during execution; no descriptor is needed. If run-time checking of index ranges is done, then those index ranges may need to be stored in a run-time descriptor. If the subscript ranges of a particular array type are static, then the ranges may be incorporated into the code that does the checking, thus eliminating the need for the run-time descriptor. If any of the descriptor entries are dynamically bound, then those parts of the descriptor must be maintained at run time.

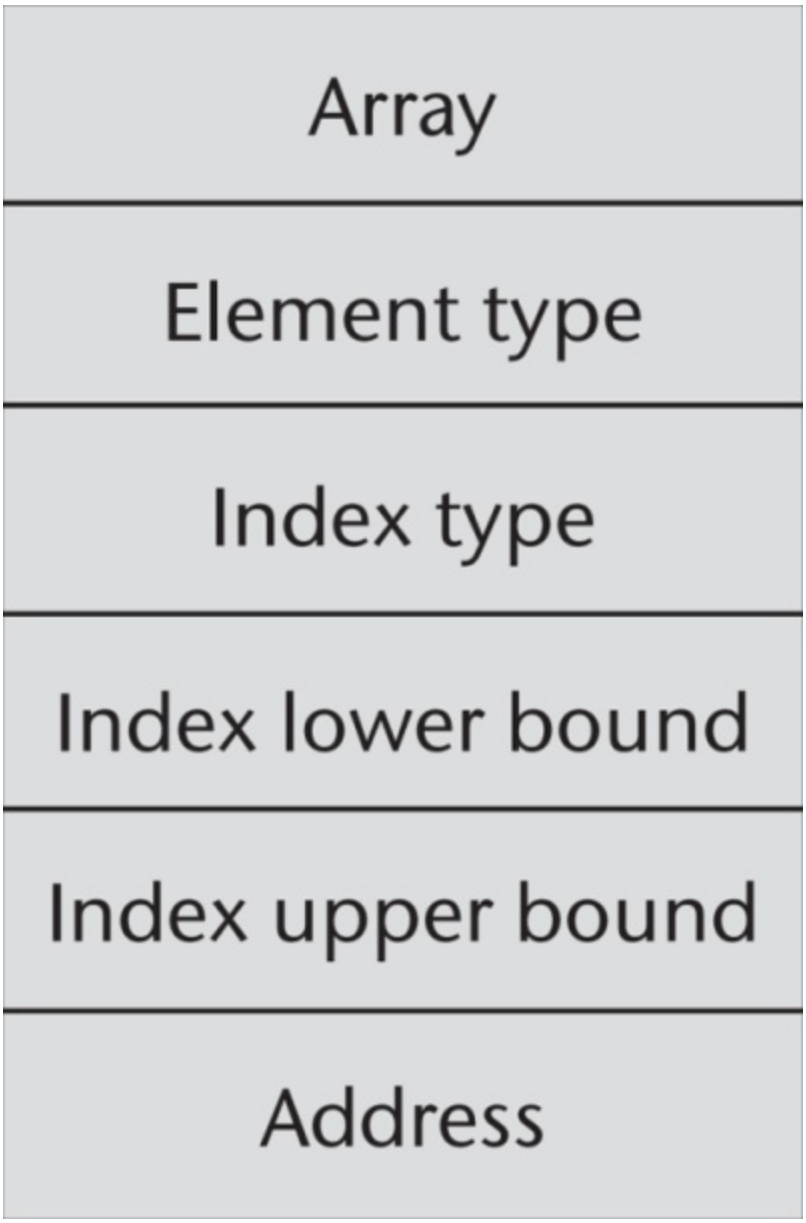


Figure 6.4 Compile-time descriptor for single-dimensioned arrays

True multidimensional arrays, that is, those that are not arrays of arrays, are more complex to implement than single-dimensioned arrays, although the

extension to more dimensions is straightforward. Hardware memory is linear—just a simple sequence of bytes. So values of data types that have two or more dimensions must be mapped onto the single-dimensioned memory. There are two ways in which multidimensional arrays can be mapped to one dimension: row major order and column major order (not used in any widely used language). In **row major order**, the elements of the array that have as their first subscript the lower bound value of that subscript are stored first, followed by the elements of the second value of the first subscript, and so forth. If the array is a matrix, it is stored by rows. For example, if the matrix had the values

3 4 7

6 2 5

1 3 8

it would be stored in row major order as

3, 4, 7, 6, 2, 5, 1, 3, 8

The access function for a multidimensional array is the mapping of its base address and a set of index values to the address in memory of the element specified by the index values. The access function for two-dimensional arrays stored in row major order can be developed as follows. In general, the address of an element is the base address of the structure plus the element size times the number of elements that precede it in the structure. For a matrix in row major order, the number of elements that precede an element is the number of rows above the element times the size of a row, plus the number of elements to the left of the element in its row. This is illustrated in Figure 6.5, in which we assume that subscript lower bounds are all zero.

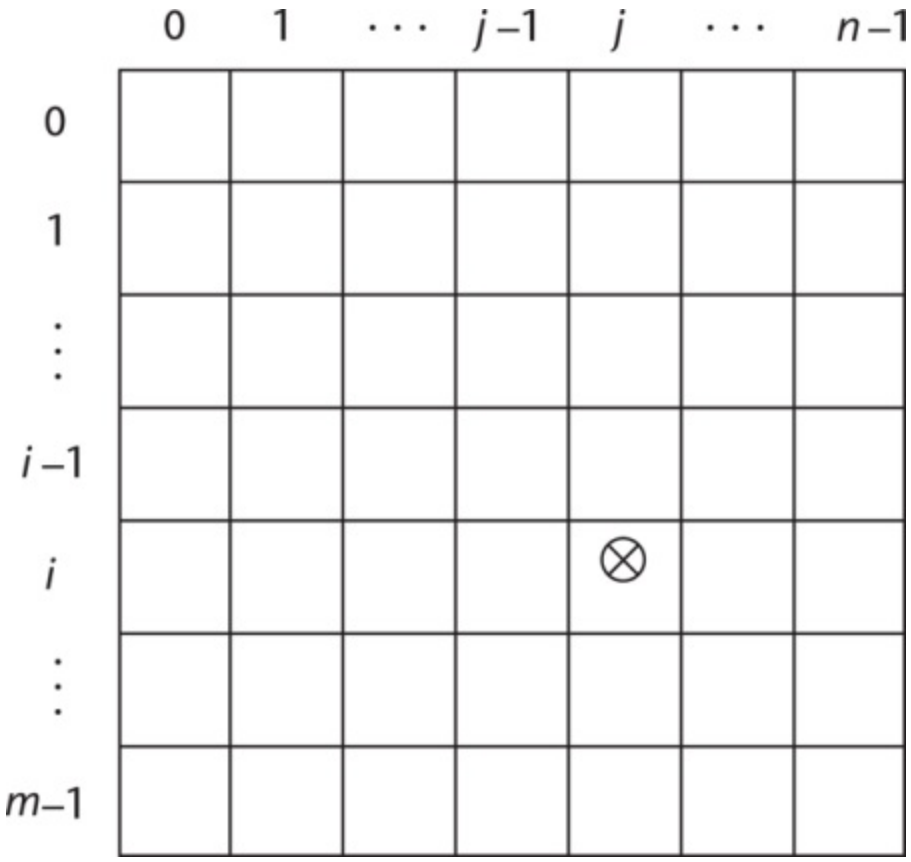


Figure 6.5 The location of the `[i,j]` element in a matrix

Figure 6.5 Full Alternative Text

To get an actual address value, the number of elements that precede the desired element must be multiplied by the element size. Now, the access function can be written as

- location(`a[i,j]`) = address of `a[0, 0]`
- + ((((number of rows above the `i`*th* row) * (size of a row))
- + (number of elements left of the `j`*th* column)) *

- element size)

Because the number of rows above the `i`*th* row is `i` and the number of elements to the left of the `j`*th* column is `j`, we have

- location(`a[i, j]`) = address of `a[0, 0]` + (((`i` * `n`) + `j`) *
- element_size)

where `n` is the number of elements per row. The first term is the constant part and the last is the variable part.

The generalization to arbitrary lower bounds results in the following access function:

- location(`a[i, j]`) = address of a[row_lb, col_lb]
- = (((`i` − row_lb) * n) + (`j` − col_lb)) * element_size

where row_lb is the lower bound of the rows and col_lb is the lower bound of the columns. This can be rearranged to the form

- location(`a[i, j]`) = address of a[row_lb, col_lb]
- − (((row_lb * n) + col_lb) * element_size)
- + (((`i` * `n`) + `j`) * element_size)

where the first two terms are the constant part and the last is the variable part. This can be generalized rather easily to an arbitrary number of dimensions.

For each dimension of an array, one add and one multiply instruction are required for the access function. Therefore, accesses to elements of arrays with several subscripts are costly. The compile-time descriptor for a multidimensional array is shown in Figure 6.6.

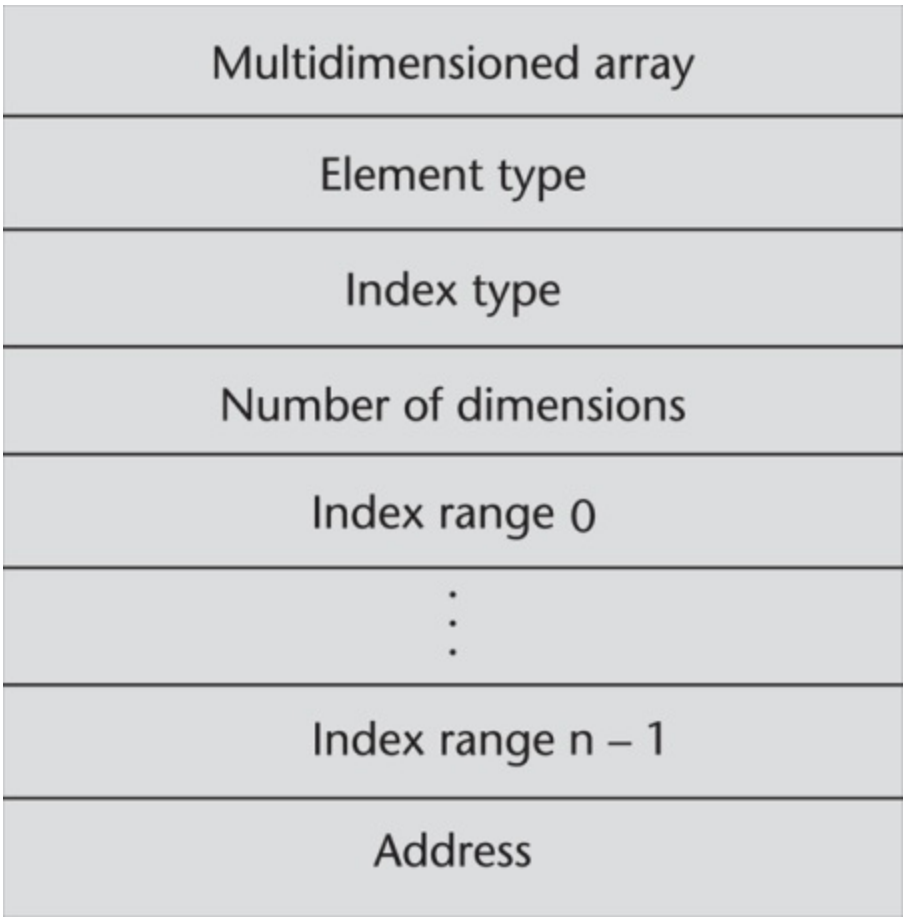


Figure 6.6 A compile-time - descriptor for a - multidimensional array

6.6 Associative Arrays

An **associative array** is an unordered collection of data elements that are indexed by an equal number of values called **keys**. In the case of non--associative arrays, the indices never need to be stored (because of their regularity). In an associative array, however, the user-defined keys must be stored in the structure. So each element of an associative array is in fact a pair of entities, a key and a value. We use Perl's design of associative arrays to illustrate this data structure. Associative arrays are also supported directly by Python, Ruby, and Swift and by the standard class libraries of Java, C++, C#, and F#.

The only design issue that is specific for associative arrays is the form of references to their elements.

6.6.1 Structure and Operations

In Perl, associative arrays are called **hashes**, because in the implementation their elements are stored and retrieved with hash functions. The namespace for Perl hashes is distinct: Every hash variable name must begin with a percent sign (`%`). Each hash element consists of two parts: a key, which is a string, and a value, which is a scalar (number, string, or reference). Hashes can be set to literal values with the assignment statement, as in

```
%salaries = ("Gary" => 75000, "Perry" => 57000,
             "Mary" => 55750, "Cedric" => 47850);
```

Individual element values are referenced using notation that is similar to that used for Perl arrays. The key value is placed in braces and the hash name is replaced by a scalar variable name that is the same except for the first character. Although hashes are not scalars, the value parts of hash elements are scalars, so references to hash element values use scalar names. Recall that scalar variable names begin with dollar signs (`$`). So, an assignment of `58850` to the element of `%salaries` with the key `"Perry"` would appear as follows:

```
$salaries{"Perry"} = 58850;
```

A new element is added using the same assignment statement form. An element can be removed from the hash with the **`delete`** operator, as in the following:

```
delete $salaries{"Gary"};
```

The entire hash can be emptied by assigning the empty literal to it, as in the following:

```
@salaries = ();
```

The size of a Perl hash is dynamic: It grows when an element is added and shrinks when an element is deleted, and also when it is emptied by assignment of the empty literal. The `exists` operator returns true or false, depending on whether its operand key is an element in the hash. For example,

```
if (exists $salaries{"Shelly"}) . . .
```

The `keys` operator, when applied to a hash, returns an array of the keys of the hash. The `values` operator does the same for the values of the hash. The `each` operator iterates over the element pairs of a hash.

Python's associative arrays, which are called **dictionaries**, are similar to those of Perl, except the values are all references to objects. The associative arrays supported by Ruby are similar to those of Python, except that the keys can be any object,[6] rather than just strings. There is a progression from Perl's hashes, in which the keys must be strings, to PHP's arrays, in which the keys can be integers or strings, to Ruby's hashes, in which any type object can be a key.

6. Objects that change do not make good keys, because the changes could change the hash function value. Therefore, arrays and hashes are never used as keys.

PHP's arrays are both normal arrays and associative arrays. They can be treated as either. The language provides functions that allow both indexed and hashed access to elements. An array can have elements that are created

with simple numeric indices and elements that are created with string hash keys.

Swift’s associative arrays are called dictionaries. The keys can be of one specific type, but the values can be of mixed types, in which case they are objects.

An associative array is much better than an array if searches of the elements are required, because the implicit hashing operation used to access elements is very efficient. Furthermore, associative arrays are ideal when the data to be stored is paired, as with employee names and their salaries. On the other hand, if every element of a list must be processed, it is more efficient to use an array.

6.6.2 Implementing Associative Arrays

The implementation of Perl’s associative arrays is optimized for fast lookups, but it also provides relatively fast reorganization when array growth requires it. A 32-bit hash value is computed for each entry and is stored with the entry, although an associative array initially uses only a small part of the hash value. When an associative array must be expanded beyond its initial size, the hash function need not be changed; rather, more bits of the hash value are used. Only half of the entries must be moved when this happens. So, although expansion of an associative array is not free, it is not as costly as might be expected.

The elements in PHP’s arrays are placed in memory through a hash function. However, all elements are linked together in the order in which they were created. The links are used to support iterative access to elements through the `current` and `next` functions.

6.7 Record Types

A **record** is an aggregate of data elements in which the individual elements are identified by names and accessed through offsets from the beginning of the structure.

There is frequently a need in programs to model a collection of data in which the individual elements are not of the same type or size. For example, information about a college student might include name, student number, grade point average, and so forth. A data type for such a collection might use a character string for the name, an integer for the student number, a floating-point for the grade point average, and so forth. Records are designed for this kind of need.

It may appear that records and heterogeneous arrays are the same, but that is not the case. The elements of a heterogeneous array are all references to data objects that reside in scattered locations, often on the heap. The elements of a record are of potentially different sizes and reside in adjacent memory locations.

Records have been part of all of the most popular programming languages, except pre-90 versions of Fortran, since the early 1960s, when they were introduced by COBOL. In some languages that support object-oriented programming, data classes serve as records.

In C, C++, C#, and Swift, records are supported with the `struct` data type. In C++, structures are a minor variation on classes. In C#, structs also are related to classes, but are quite different from them. C# structs are stack--allocated value types, as opposed to class objects, which are heap-allocated reference types. Structs in C++ and C# are normally used as encapsulation structures, rather than data structures. They are further discussed in this capacity in Chapter 11. Structs are also included in ML and F#.

In Python and Ruby, records can be implemented as hashes, which themselves can be elements of arrays.

The following sections describe how records are declared or defined, how references to fields within records are made, and the common record operations.

The following design issues are specific to records:

- What is the syntactic form of references to fields?
- Are elliptical references allowed?

6.7.1 Definitions of Records

The fundamental difference between a record and an array is that record elements, or **fields**, are not referenced by indices. Instead, the fields are named with identifiers, and references to the fields are made using these identifiers. Another difference between arrays and records is that records in some languages are allowed to include unions, which are discussed in Section 6.10.

The COBOL form of a record declaration, which is part of the data division of a COBOL program, is illustrated in the following example:

```
01  EMPLOYEE-RECORD.
    02  EMPLOYEE-NAME.
        05  FIRST   PICTURE IS X(20).
        05  Middle  PICTURE IS X(10).
        05  LAST    PICTURE IS X(20).
    02  HOURLY-RATE PICTURE IS 99V99.
```

The `EMPLOYEE-RECORD` record consists of the `EMPLOYEE-NAME` record and the `HOURLY-RATE` field. The numerals `01`, `02`, and `05` that begin the lines of the record declaration are **level numbers**, which indicate by their relative values the hierarchical structure of the record. Any line that is followed by a line with a higher-level number is itself a record. The `PICTURE` clauses show the formats of the field storage locations, with `X(20)` specifying 20 alphanumeric characters and `99V99` specifying four decimal digits with the decimal point in the middle.

In Java, records can be defined as data classes, with nested records defined as nested classes. Data members of such classes serve as the record fields.

6.7.2 References to Record Fields

References to the individual fields of records are syntactically specified by several different methods, two of which name the desired field and its enclosing records. COBOL field references have the form

- field_name `OF` record_name_1 `OF . . . OF` record_name_n

where the first record named is the smallest or innermost record that contains the field. The next record name in the sequence is that of the record that contains the previous record, and so forth. For example, the `Middle` field in the COBOL record example above can be referenced with

```
Middle OF EMPLOYEE-NAME OF EMPLOYEE-RECORD
```

Most of the other languages use **dot notation** for field references, where the components of the reference are connected with periods. Names in dot notation have the opposite order of COBOL references: They use the name of the largest enclosing record first and the field name last. For example, if `Middle` is a field in the `Employee_Name` record which is embedded in the `Employee_Record` record, it would be referenced with the following:

```
Employee_Record.Employee_Name.Middle
```

A **fully qualified reference** to a record field is one in which all intermediate record names, from the largest enclosing record to the specific field, are named in the reference. In the COBOL example above the field reference is fully qualified. As an alternative to fully qualified references, COBOL allows **elliptical references** to record fields. In an elliptical reference, the field is named, but any or all of the enclosing record names can be omitted, as long as the resulting reference is unambiguous in the referencing environment. For example, `FIRST`, `FIRST OF EMPLOYEE-NAME`, and `FIRST OF EMPLOYEE-RECORD` are elliptical references to the employee's first name in the COBOL record declared above. Although elliptical references are a programmer

convenience, they require a compiler to have elaborate data structures and procedures in order to correctly identify the referenced field. They are also somewhat detrimental to readability.

6.7.3 Evaluation

Records are frequently valuable data types in programming languages. The design of record types is straightforward, and their use is safe.

Records and arrays are closely related structural forms, and therefore it is interesting to compare them. Arrays are used when all the data values have the same type and/or are processed in the same way. This processing is easily done when there is a systematic way of sequencing through the structure. Such processing is well supported by using dynamic subscripting as the addressing method.

Records are used when the collection of data values is heterogeneous and the different fields are not processed in the same way. Also, the fields of a record often need not be processed in a particular order. Field names are like literal, or constant, subscripts. Because they are static, they provide very efficient access to the fields. Dynamic subscripts could be used to access record fields, but it would disallow type checking and would also be slower.

Records and arrays represent thoughtful and efficient methods of fulfilling two separate but related applications of data structures.

6.7.4 Implementation of Record Types

The fields of records are stored in adjacent memory locations. But because the sizes of the fields are not necessarily the same, the access method used for arrays is not used for records. Instead, the offset address, relative to the beginning of the record, is associated with each field. Field accesses are all handled using these offsets. The compile-time descriptor for a record has the

general form shown in Figure 6.7. Run-time descriptors for records are unnecessary.

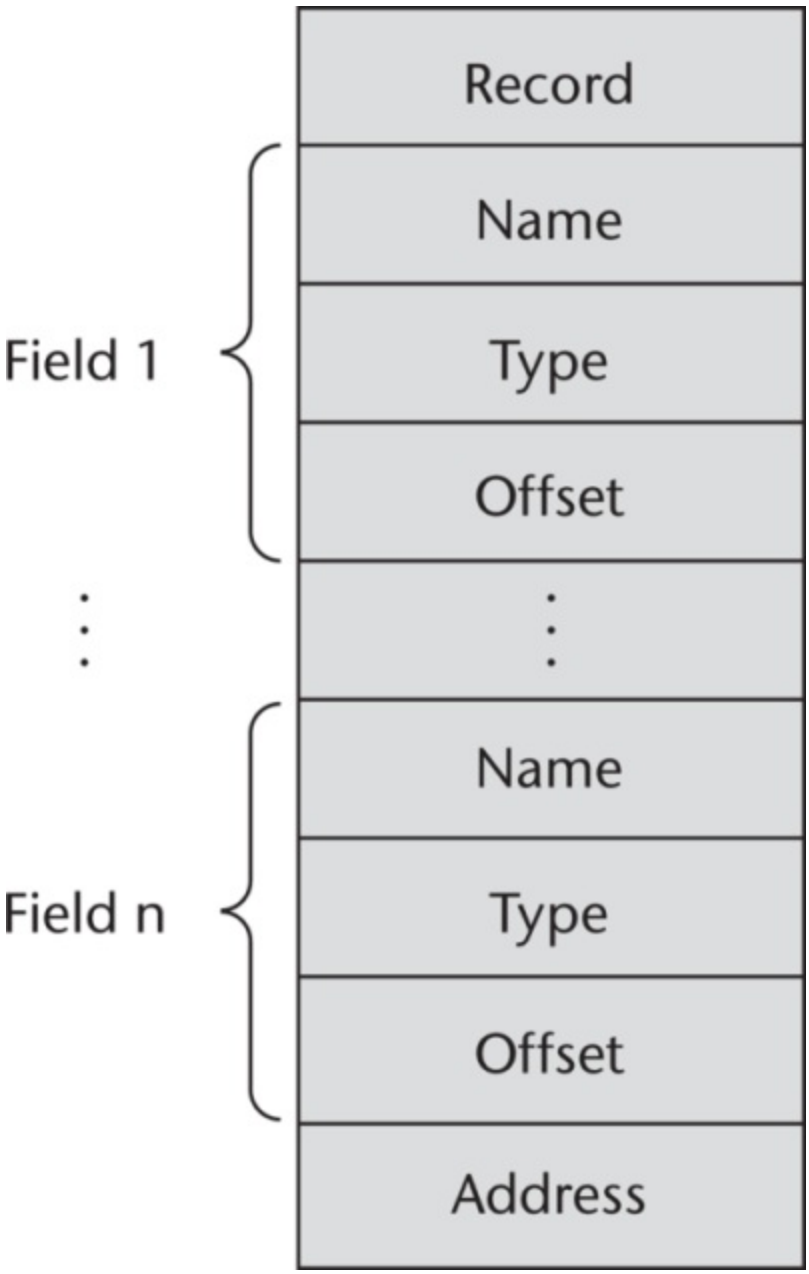


Figure 6.7 A compile-time - descriptor for a record

Figure 6.7 Full Alternative Text

6.8 Tuple Types

A tuple is a data type that is similar to a record, except that the elements are not named.

Python includes an immutable tuple type. If a tuple needs to be changed, it can be converted to an array with the `list` function. After the change, it can be converted back to a tuple with the `tuple` function. One use of tuples is when an array must be write protected, such as when it is sent as a parameter to an external function and the user does not want the function to be able to modify the parameter.

Python's tuples are closely related to its lists, except that tuples are immutable. A tuple is created by assigning a tuple literal, as in the following example:

```
myTuple = (3, 5.8, 'apple')
```

Notice that the elements of a tuple need not be of the same type.

The elements of a tuple can be referenced with indexing in brackets, as in the following:

```
myTuple[1]
```

This references the first element of the tuple, because tuple indexing begins at `1`.

Tuples can be catenated with the plus (+) operator. They can be deleted with the **`del`** statement. There are also other operators and functions that operate on tuples.

ML includes a tuple data type. An ML tuple must have at least two elements, whereas Python's tuples can be empty or contain one element. As in Python, an ML tuple can include elements of mixed types. The following statement creates a tuple:

```
val myTuple = (3, 5.8, 'apple');
```

The syntax of a tuple element access is as follows:

```
#1(myTuple);
```

This references the first element of the tuple.

A new tuple type can be defined in ML with a type declaration, such as the following:

```
type intReal = int * real;
```

Values of this type consist of an integer and a real. The asterisk can be misleading. It is used to separate the tuple components, indicating a type product, and has nothing to do with arithmetic.

F# also has tuples. A tuple is created by assigning a tuple value, which is a list of expressions separated by commas and delimited by parentheses, to a name in a **`let`** statement. If a tuple has two elements, they can be referenced with the functions `fst` and `snd`, respectively. The elements of a tuple with more than two elements are often referenced with a tuple pattern on the left side of a **`let`** statement. A tuple pattern is simply a sequence of names, one for each element of the tuple, with or without the delimiting parentheses. When a tuple pattern is the left side of a **`let`** construct, it is a multiple assignment. For example, consider the following **`let`** constructs:

```
let tup = (3, 5, 7);;
let a, b, c  = tup;;
```

This assigns `3` to `a`, `5` to `b`, and `7` to `c`.

Tuples are used in Python, ML, and F# to allow functions to return multiple values. In Swift, tuples are passed by value, so they are sometimes used to pass data to a function when the function is not to change that data.

6.9 List Types

Lists were first supported in the first functional programming language, Lisp. They have always been part of the functional languages, but in recent years they have found their way into some imperative languages.

Lists in Scheme and Common Lisp are delimited by parentheses and the elements are not separated by any punctuation. For example,

```
(A B C D)
```

Nested lists have the same form, so we could have

```
(A (B C) D)
```

In this list, `(B C)` is a list nested inside the outer list.

Data and code have the same syntactic form in Lisp and its descendants. If the list `(A B C)` is interpreted as code, it is a call to the function `A` with parameters `B` and `C`.

The fundamental list operations in Scheme are two functions that take lists apart and two that build lists. The `CAR` function returns the first element of its list parameter. For example, consider the following example:

```
(CAR '(A B C))
```

The quote before the parameter list is to prevent the interpreter from considering the list a call to the `A` function with the parameters `B` and `C`, in which case it would interpret it. This call to `CAR` returns `A`.

The `CDR` function returns its parameter list minus its first element. For example, consider the following example:

```
(CDR '(A B C))
```

This function call returns the list `(B C)`.

Common Lisp also has the functions `FIRST` (same as `CAR`), `SECOND`, `...`, `TENTH`, which return the element of their list parameters that is specified by their names.

In Scheme and Common Lisp, new lists are constructed with the `CONS` and `LIST` functions. The function `CONS` takes two parameters and returns a new list with its first parameter as the first element and its second parameter as the remainder of that list. For example, consider the following:

```
(CONS 'A '(B C))
```

This call returns the new list `(A B C)`.

The `LIST` function takes any number of parameters and returns a new list with the parameters as its elements. For example, consider the following call to `LIST`:

```
(LIST 'A 'B '(C D))
```

This call returns the new list `(A B (C D))`.

ML has lists and list operations, although their appearance is not like those of Scheme. Lists are specified in square brackets, with the elements separated by commas, as in the following list of integers:

```
[5, 7, 9]
```

`[]` is the empty list, which could also be specified with `nil`.

The Scheme `CONS` function is implemented as a binary infix operator in ML, represented as `::`. For example,

```
3 :: [5, 7, 9]
```

returns the following new list: `[3, 5, 7, 9]`.

The elements of a list must be of the same type, so the following list would be illegal:

```
[5, 7.3, 9]
```

ML has functions that correspond to Scheme's `CAR` and `CDR`, named `hd` (head) and `tl` (tail). For example,

```
hd [5, 7, 9] is 5
tl [5, 7, 9] is [7, 9]
```

Lists and list operations in Scheme and ML are more fully discussed in Chapter 15.

Lists in F# are related to those of ML with a few notable differences. - Elements of a list in F# are separated by semicolons, rather than the commas of ML. The operations `hd` and `tl` are the same, but they are called as methods of the `List` class, as in `List.hd [1; 3; 5; 7]`, which returns `1`. The `CONS` operation of F# is specified as two colons, as in ML.

Python includes a list data type, which also serves as Python's arrays. Unlike the lists of Scheme, Common Lisp, ML, and F#, the lists of Python are mutable. They can contain any data value or object. A Python list is created with an assignment of a list value to a name. A list value is a sequence of expressions that are separated by commas and delimited with brackets. For example, consider the following statement:

```
myList = [3, 5.8, "grape"]
```

The elements of a list are referenced with subscripts in brackets, as in the following example:

```
x = myList[1]
```

This statement assigns `5.8` to `x`. The elements of a list are indexed starting at zero. List elements also can be updated by assignment. A list element can be deleted with `del`, as in the following statement:

```
del myList[1]
```

This statement removes the second element of `myList`.

Python includes a powerful mechanism for creating arrays called **list comprehensions**. A list comprehension is an idea derived from set notation. It first appeared in the functional programming language Haskell (see

Chapter 15). The mechanics of a list comprehension is that a function is applied to each of the elements of a given array and a new array is constructed from the results. The syntax of a Python list comprehension is as follows:

[expression **for** iterate_var **in** array **if** condition]

Consider the following example:

```
[x * x  for  x  in range(12)  if  x % 3 == 0]
```

The **range** function creates the array `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]`. The conditional filters out all numbers in the array that are not evenly divisible by `3`. Then, the expression squares the remaining numbers. The results of the squaring are collected in an array, which is returned. This list comprehension returns the following array:

```
[0, 9, 36, 81]
```

Slices of lists are also supported in Python.

Haskell's list comprehensions have the following form:

- `[`body `|` qualifiers`]`

For example, consider the following definition of a list:

```
[n * n | n <- [1..10]]
```

This defines a list of the squares of the numbers from `1` to `10`.

F# includes list comprehensions, which in that language can also be used to create arrays. For example, consider the following statement:

```
let myArray = [|for i in 1 .. 5 -> (i * i) |];;
```

This statement creates the array `[1; 4; 9; 16; 25]` and names it `myArray`.

Recall from Section 6.5 that C# and Java support generic heap-dynamic collection classes, `List` and `ArrayList`, respectively. These structures are

actually lists.

6.10 Union Types

A **union** is a type whose variables may store different type values at different times during program execution. As an example of the need for a union type, consider a table of constants for a compiler, which is used to store the constants found in a program being compiled. One field of each table entry is for the value of the constant. Suppose that for a particular language being compiled, the types of constants were integer, floating-point, and Boolean. In terms of table management, it would be convenient if the same location, a table field, could store a value of any of these three types. Then all constant values could be addressed in the same way. The type of such a location is, in a sense, the union of the three value types it can store.

6.10.1 Design Issues

The problem of type checking union types, which is discussed in Section 6.12, is their major design issue.

6.10.2 Discriminated Versus Free Unions

C and C++ provide union constructs in which there is no language support for type checking. In C and C++, the **`union`** construct is used to specify union structures. The unions in these languages are called **free unions**, because programmers are allowed complete freedom from type checking in their use. For example, consider the following C union:

```
union flexType {
 int intEl;
 float  floatEl;
};
union flexType el1;
```

```
float x;
. . .
el1.intEl = 27;
x = el1.floatEl;
```

This last assignment is not type checked, because the system cannot determine the current type of the current value of `el1`, so it assigns the bit string representation of `27` to the **`float`** variable `x`, which, of course, is nonsense.

Type checking of unions requires that each union construct include a type indicator. Such an indicator is called a **tag**, or **discriminant**, and a union with a discriminant is called a **discriminated union**. The first language to provide discriminated unions was ALGOL 68. They are now supported by ML, Haskell, and F#.

6.10.3 Unions in F#

A union is declared in F# with a type statement using OR operators (`|`) to define the components. For example, we could have the following:

```
type intReal =
   | IntValue of int
   | RealValue of float;;
```

In this example, `intReal` is the union type. `IntValue` and `RealValue` are constructors. Values of type `intReal` can be created using the constructors as if they were a function, as in the following examples:7

7. The **`let`** statement is used to assign values to names and to create a static scope; the double semicolons are used to terminate statements when the F# interactive interpreter is being used.

```
let ir1 = IntValue 17;;
let ir2 = RealValue 3.4;;
```

Accessing the value of a union is done with a pattern-matching structure. Pattern matching in F# is specified with the **`match`** reserved word. The general form of the construct is as follows:

- **match** pattern **with**
- | expression_list1−>expression1
- | . . .
- | expression_listn−>expressionn

The pattern can be any data type. The expression list can include wild card characters (_) or be solely a wild card character. For example, consider the following match construct:

```
let a = 7;;
let b = "grape";;
let x = match (a, b) with
     | 4, "apple" -> apple
     | _, "grape" -> grape
     | _ -> fruit;;
```

To display the type of the `intReal` union, the following function could be used:

```
let printType value =
    match value with
        | IntValue value -> printfn "It is an integer"
        | RealValue value -> printfn "It is a float";;
```

The following lines show calls to this function and the output:

```
printType ir1;;
It is an integer
printType ir2;;
It is a float
```

6.10.4 Evaluation

Unions are potentially unsafe constructs in some languages. They are one of the reasons why C and C++ are not strongly typed: These languages do not allow type checking of references to their unions. On the other hand, unions can be safely used, as in their design in ML, Haskell, and F#.

Neither Java nor C# includes unions, which may be reflective of the growing concern for safety in some programming languages.

6.10.5 Implementation of Union Types

Unions are implemented by simply using the same address for every possible variant. Sufficient storage for the largest variant is allocated.

6.11 Pointer and Reference Types

A **pointer** type is one in which the variables have a range of values that consists of memory addresses and a special value, **nil**. The value nil is not a valid address and is used to indicate that a pointer cannot currently be used to reference a memory cell.

Pointers are designed for two distinct kinds of uses. First, pointers provide some of the power of indirect addressing, which is frequently used in assembly language programming. Second, pointers provide a way to manage dynamic storage. A pointer can be used to access a location in an area where storage is dynamically allocated called a **heap**.

Variables that are dynamically allocated from the heap are called **heap--dynamic variables**. They often do not have identifiers associated with them and thus can be referenced only by pointer or reference type variables. - Variables without names are called **anonymous variables**. It is in this latter application area of pointers that the most important design issues arise.

Pointers, unlike arrays and records, are not structured types, although they are defined using a type operator (`*` in C and C++). Furthermore, they are also different from scalar variables because they are used to reference some other variable, rather than being used to store data. These two categories of variables are called **reference types** and **value types**, respectively.

Both kinds of uses of pointers add writability to a language. For example, suppose it is necessary to implement a dynamic structure like a binary tree in a language that does not have pointers or dynamic storage. This would require the programmer to provide and maintain a pool of available tree nodes, which would probably be implemented in parallel arrays. Also, it would be necessary for the programmer to guess the maximum number of required nodes. This is clearly an awkward and error-prone way to deal with binary trees.

Reference variables, which are discussed in Section 6.11.6, are closely related

to pointers.

6.11.1 Design Issues

The primary design issues particular to pointers are the following:

- What are the scope and lifetime of a pointer variable?
- What is the lifetime of a heap-dynamic variable (the value a pointer references)?
- Are pointers restricted as to the type of value to which they can point?
- Are pointers used for dynamic storage management, indirect addressing, or both?
- Should the language support pointer types, reference types, or both?

6.11.2 Pointer Operations

Languages that provide a pointer type usually include two fundamental pointer operations: assignment and dereferencing. The first operation sets a pointer variable’s value to some useful address. If pointer variables are used only to manage dynamic storage, then the allocation mechanism, whether by operator or built-in subprogram, serves to initialize the pointer variable. If pointers are used for indirect addressing to variables that are not heap dynamic, then there must be an explicit operator or built-in subprogram for fetching the address of a variable, which can then be assigned to the pointer variable.

An occurrence of a pointer variable in an expression can be interpreted in two distinct ways. First, it could be interpreted as a reference to the contents of the memory cell to which it is bound, which in the case of a pointer is an address. This is exactly how a nonpointer variable in an expression would be interpreted, although in that case its value likely would not be an address.

However, the pointer also could be interpreted as a reference to the value in the memory cell pointed to by the memory cell to which the pointer variable is bound. In this case, the pointer is interpreted as an indirect reference. The former case is a normal pointer reference; the latter is the result of **dereferencing** the pointer. Dereferencing, which takes a reference through one level of indirection, is the second fundamental pointer operation.

Dereferencing of pointers can be either explicit or implicit. In many contemporary languages, it occurs only when explicitly specified. In C++, it is explicitly specified with the asterisk (`*`) as a prefix unary operator. Consider the following example of dereferencing: If `ptr` is a pointer variable with the value 7080 and the cell whose address is 7080 has the value 206, then the assignment

```
j = *ptr
```

sets `j` to 206. This process is shown in Figure 6.8.

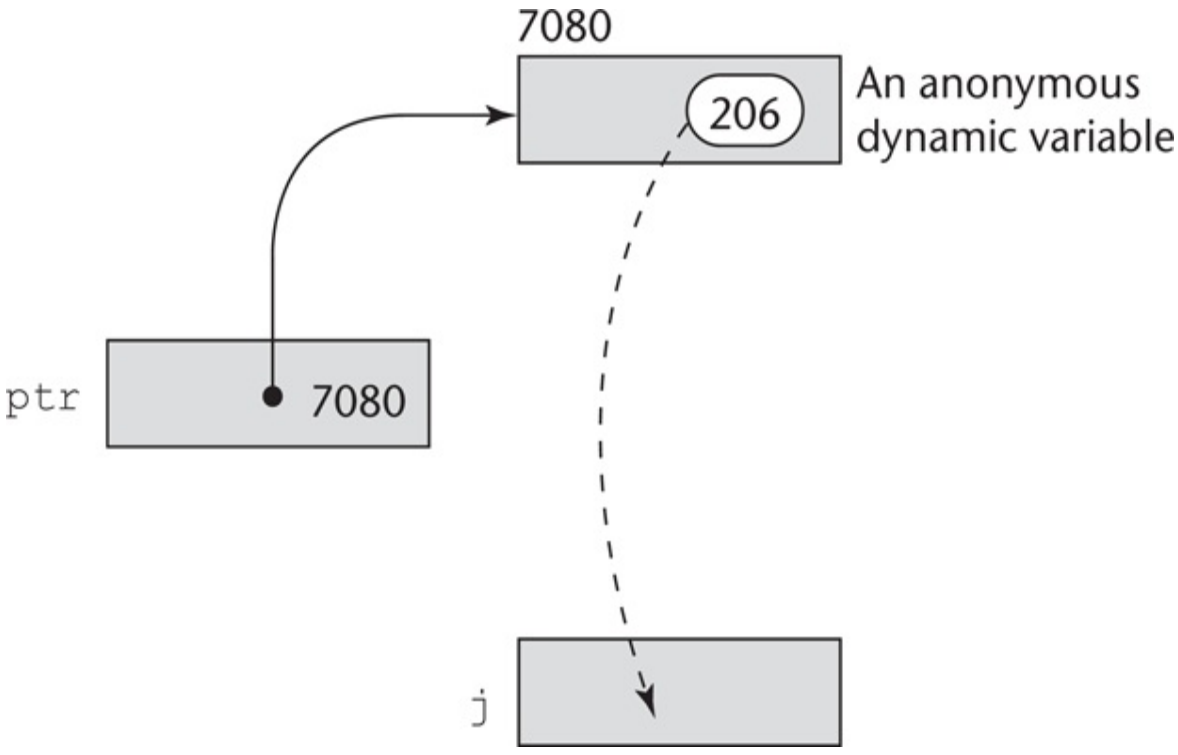


Figure 6.8 The assignment -

operation `j = *ptr`

Figure 6.8 Full Alternative Text

When pointers point to records, the syntax of the references to the fields of these records varies among languages. In C and C++, there are two ways a pointer to a record can be used to reference a field in that record. If a pointer variable `p` points to a record with a field named `age`, `(*p).age` can be used to refer to that field. The operator `->`, when used between a pointer to a struct and a field of that struct, combines dereferencing and field reference. For example, the expression `p -> age` is equivalent to `(*p).age`.

Languages that provide pointers for the management of a heap must include an explicit allocation operation. Allocation is sometimes specified with a subprogram, such as `malloc` in C. In languages that support object-oriented programming, allocation of heap objects is often specified with the **`new`** operator. C++, which does not provide implicit deallocation, uses **`delete`** as its deallocation operator.

6.11.3 Pointer Problems

The first high-level programming language to include pointer variables was PL/I, in which pointers could be used to refer to both heap-dynamic variables and other program variables. The pointers of PL/I were highly flexible, but their use could lead to several kinds of programming errors. Some of the problems of PL/I pointers are also present in the pointers of subsequent languages. Some recent languages, such as Java, have replaced pointers completely with reference types, which, along with implicit deallocation, minimize the primary problems with pointers. A reference type is really only a pointer with restricted operations.

6.11.3.1 Dangling Pointers

A **dangling pointer**, or **dangling reference**, is a pointer that contains the address of a heap-dynamic variable that has been deallocated. Dangling pointers are dangerous for several reasons. First, the location being pointed to may have been reallocated to some new heap-dynamic variable. If the new variable is not the same type as the old one, type checks of uses of the dangling pointer are invalid. Even if the new dynamic variable is the same type, its new value will have no relationship to the old pointer's dereferenced value. Furthermore, if the dangling pointer is used to change the heap-dynamic variable, the value of the new heap-dynamic variable will be destroyed. Finally, it is possible that the location now is being temporarily used by the storage management system, possibly as a pointer in a chain of available blocks of storage, thereby allowing a change to the location to cause the storage manager to fail.

The following sequence of operations creates a dangling pointer in many languages:

1. A new heap-dynamic variable is created and pointer `p1` is set to point to it.

2. Pointer `p2` is assigned `p1`'s value.

3. The heap-dynamic variable pointed to by `p1` is explicitly deallocated (possibly setting `p1` to `nil`), but `p2` is not changed by the operation. `p2` is now a dangling pointer. If the deallocation operation did not change `p1`, both `p1` and `p2` would be dangling. (Of course, this is a problem of aliasing—`p1` and `p2` are aliases.)

For example, in C++ we could have the following:

```
int * arrayPtr1;
int * arrayPtr2 = new int[100];
arrayPtr1 = arrayPtr2;
delete [] arrayPtr2;
// Now, arrayPtr1 is dangling, because the heap storage
// to which it was pointing has been deallocated.
```

In C++, both `arrayPtr1` and `arrayPtr2` are now dangling pointers, because the C++ **`delete`** operator has no effect on the value of its operand pointer. In

C++, it is common (and safe) to follow a **`delete`** operator with an assignment of zero, which represents null, to the pointer whose pointed-to value has been deallocated.

Notice that the explicit deallocation of dynamic variables is the cause of dangling pointers.

history note

Pascal included an explicit deallocate operator: dispose. Because of the problem of dangling pointers caused by dispose, some Pascal implementations simply ignored dispose when it appeared in a program. Although this effectively prevents dangling pointers, it also disallows the reuse of heap storage that the program no longer needs. Recall that Pascal initially was designed as a teaching language, rather than as an industrial tool.

6.11.3.2 Lost Heap-Dynamic Variables

A **lost heap-dynamic variable** is an allocated heap-dynamic variable that is no longer accessible to the user program. Such variables are often called **garbage**, because they are not useful for their original purpose, and they also cannot be reallocated for some new use in the program. Lost heap-dynamic variables are most often created by the following sequence of operations:

1. Pointer `p1` is set to point to a newly created heap-dynamic variable.

2. `p1` is later set to point to another newly created heap-dynamic variable.

The first heap-dynamic variable is now inaccessible, or lost. This is sometimes called **memory leakage**. Memory leakage is a problem, regardless of whether the language uses implicit or explicit deallocation. In the following sections, we investigate how language designers have dealt with the problems of dangling pointers and lost heap-dynamic variables.

6.11.4 Pointers in C and C++

In C and C++, pointers can be used in the same ways as addresses are used in assembly languages. This means they are extremely flexible but must be used with great care. This design offers no solutions to the dangling pointer or lost heap-dynamic variable problems. However, the fact that pointer arithmetic is possible in C and C++ makes their pointers more interesting than those of the other programming languages.

C and C++ pointers can point at any variable, regardless of where it is allocated. In fact, they can point anywhere in memory, whether there is a variable there or not, which is one of the dangers of such pointers.

In C and C++, the asterisk (`*`) denotes the dereferencing operation and the ampersand (`&`) denotes the operator for producing the address of a variable. For example, consider the following code:

```
int *ptr;
int count, init;
. . .
ptr = &init;
count = *ptr;
```

The assignment to the variable `ptr` sets it to the address of `init`. The assignment to `count` dereferences `ptr` to produce the value at `init`, which is then assigned to `count`. So, the effect of the two assignment statements is to assign the value of `init` to `count`. Notice that the declaration of a pointer specifies its domain type.

Pointers can be assigned the address value of any variable of the correct domain type, or they can be assigned the constant zero, which is used for `nil`.

Pointer arithmetic is also possible in some restricted forms. For example, if `ptr` is a pointer variable that is declared to point at some variable of some data type, then

```
ptr + index
```

is a legal expression. The semantics of such an expression is as follows. Instead of simply adding the value of `index` to `ptr`, the value of `index` is first scaled by the size of the memory cell (in memory units) to which `ptr` is pointing (its base type). For example, if `ptr` points to a memory cell for a type that is four memory units in size, then `index` is multiplied by 4, and the result is added to `ptr`. The primary purpose of this sort of address arithmetic is array manipulation. The following discussion is related to single--dimensioned arrays only.

In C and C++, all arrays use zero as the lower bound of their subscript ranges, and array names without subscripts always refer to the address of the first element. Consider the following declarations:

```
int list [10];
int *ptr;
```

Now consider the assignment

```
ptr = list;
```

This assigns the address of `list[0]` to `ptr`. Given this assignment, the following are true:

- `*(ptr + 1)` is equivalent to `list[1]`.
- `*(ptr + index)` is equivalent to `list[index]`.
- `ptr[index]` is equivalent to `list[index]`.

It is clear from these statements that the pointer operations include the same scaling that is used in indexing operations. Furthermore, pointers to arrays can be indexed as if they were array names.

Pointers in C and C++ can point to functions. This feature is used to pass functions as parameters to other functions. Pointers are also used for parameter passing, as discussed in Chapter 9.

C and C++ include pointers of type **`void *`**, which can point at values of any type. In effect they are generic pointers. However, type checking is not a

problem with **`void *`** pointers, because these languages disallow dereferencing them. One common use of **`void *`** pointers is as the types of parameters of functions that operate on memory. For example, suppose we wanted a function to move a sequence of bytes of data from one place in memory to another. It would be most general if it could be passed two pointers of any type. This would be legal if the corresponding formal parameters in the function were **`void *`** type. The function could then convert them to **`char *`** type and do the operation, regardless of what type pointers were sent as actual parameters.

6.11.5 Reference Types

A **reference type** variable is similar to a pointer, with one important and fundamental difference: A pointer refers to an address in memory, while a reference refers to an object or a value in memory. As a result, although it is natural to perform arithmetic on addresses, it is not sensible to do arithmetic on references.

C++ includes a special kind of reference type that is used primarily for the formal parameters in function definitions. A C++ reference type variable is a constant pointer that is always implicitly dereferenced. Because a C++ reference type variable is a constant, it must be initialized with the address of some variable in its definition, and after initialization a reference type variable can never be set to reference any other variable. The implicit dereference prevents assignment to the address value of a reference variable.

Reference type variables are specified in definitions by preceding their names with ampersands (`&`). For example,

```
int result = 0;
 int &ref_result = result;
. . .
ref_result = 100;
```

In this code segment, `result` and `ref_result` are aliases.

When used as formal parameters in function definitions, C++ reference types

provide for two-way communication between the caller function and the called function. This is not possible with nonpointer primitive parameter types, because C++ parameters are passed by value. Passing a pointer as a parameter accomplishes the same two-way communication, but pointer formal parameters require explicit dereferencing, making the code less readable and less safe. Reference parameters are referenced in the called function exactly as are other parameters. The calling function need not specify that a parameter whose corresponding formal parameter is a reference type is anything unusual. The compiler passes addresses, rather than values, to reference parameters.

In their quest for increased safety over C++, the designers of Java removed C++-style pointers altogether. Unlike C++ reference variables, Java reference variables can be assigned to refer to different class instances; they are not constants. All Java class instances are referenced by reference variables. That is, in fact, the only use of reference variables in Java. These issues are discussed further in Chapter 12.

In the following, `String` is a standard Java class:

```
String str1;
. . .
str1 = "This is a Java literal string";
```

In this code, `str1` is defined to be a reference to a `String` class instance or object. It is initially set to null. The subsequent assignment sets `str1` to reference the `String` object, `"This is a Java literal string"`.

Because Java class instances are implicitly deallocated (there is no explicit deallocation operator), there cannot be dangling references in Java.

C# includes both the references of Java and the pointers of C++. However, the use of pointers is strongly discouraged. In fact, any subprogram that uses pointers must include the **`unsafe`** modifier. Note that although objects pointed to by references are implicitly deallocated, that is not true for objects pointed to by pointers. Pointers were included in C# primarily to allow C# programs to interoperate with C and C++ code.

All variables in the object-oriented languages Smalltalk, Python, and Ruby

are references. They are always implicitly dereferenced. Furthermore, the direct values of these variables cannot be accessed.

6.11.6 Evaluation

The problems of dangling pointers and garbage have already been discussed at length. The problems of heap management are discussed in Section 6.11.7.3.

Pointers have been compared with the goto. The goto statement widens the range of statements that can be executed next. Pointer variables widen the range of memory cells that can be referenced by a variable. Perhaps the most damning statement about pointers was made by Hoare (1973): “Their introduction into high-level languages has been a step backward from which we may never recover.”

On the other hand, pointers are essential in some kinds of programming applications. For example, pointers are necessary to write device drivers, in which specific absolute addresses must be accessed.

The references of Java and C# provide some of the flexibility and the capabilities of pointers, without the hazards. It remains to be seen whether programmers will be willing to trade the full power of C and C++ pointers for the greater safety of references. The extent to which C# programs use pointers will be one measure of this.

6.11.7 Implementation of Pointer and Reference Types

In most languages, pointers are used in heap management. The same is true for Java and C# references, as well as the variables in Smalltalk and Ruby, so we cannot treat pointers and references separately. First, we briefly describe how pointers and references are represented internally. We then discuss two possible solutions to the dangling pointer problem. Finally, we describe the

major problems with heap-management techniques.

6.11.7.1 Representations of Pointers and References

In most larger computers, pointers and references are single values stored in memory cells. However, in early microcomputers based on Intel microprocessors, addresses have two parts: a segment and an offset. So, pointers and references are implemented in these systems as pairs of 16-bit cells, one for each of the two parts of an address.

6.11.7.2 Solutions to the Dangling-Pointer Problem

There have been several proposed solutions to the dangling-pointer problem. Among these are **tombstones** (Lomet, 1975), in which every heap-dynamic variable includes a special cell, called a tombstone, that is itself a pointer to the heap-dynamic variable. The actual pointer variable points only at tombstones and never to heap-dynamic variables. When a heap-dynamic variable is deallocated, the tombstone remains but is set to nil, indicating that the heap-dynamic variable no longer exists. This approach prevents a pointer from ever pointing to a deallocated variable. Any reference to any pointer that points to a `nil` tombstone can be detected as an error.

Tombstones are costly in both time and space. Because tombstones are never deallocated, their storage is never reclaimed. Every access to a heap-dynamic variable through a tombstone requires one more level of indirection, which requires an additional machine cycle on most computers. Apparently none of the designers of the more popular languages have found the additional safety to be worth this additional cost, because no widely used language uses tombstones.

An alternative to tombstones is the **locks-and-keys approach** used in the implementation of UW-Pascal (Fischer and LeBlanc, 1977, 1980). In this compiler, pointer values are represented as ordered pairs (key, address), where the key is an integer value. Heap-dynamic variables are represented as the storage for the variable plus a header cell that stores an integer lock value. When a heap-dynamic variable is allocated, a lock value is created and placed both in the lock cell of the heap-dynamic variable and in the key cell of the pointer that is specified in the call to **`new`**. Every access to the dereferenced pointer compares the key value of the pointer to the lock value in the heap-dynamic variable. If they match, the access is legal; otherwise the access is treated as a run-time error. Any copies of the pointer value to other pointers must copy the key value. Therefore, any number of pointers can reference a given heap-dynamic variable. When a heap-dynamic variable is deallocated with **`dispose`**, its lock value is cleared to an illegal lock value. Then, if a pointer other than the one specified in the **`dispose`** is dereferenced, its address value will still be intact, but its key value will no longer match the lock, so the access will not be allowed.

Of course, the best solution to the dangling-pointer problem is to take deallocation of heap-dynamic variables out of the hands of programmers. If programs cannot explicitly deallocate heap-dynamic variables, there will be no dangling pointers. To do this, the run-time system must implicitly deallocate heap-dynamic variables when they are no longer useful. Lisp systems have always done this. Both Java and C# also use this approach for their reference variables. Recall that C#'s pointers do not include implicit deallocation.

6.11.7.3 Heap Management

Heap management can be a very complex run-time process. We examine the process in two separate situations: one in which all heap storage is allocated and deallocated in units of a single size, and one in which variable-size segments are allocated and deallocated. Note that for deallocation, we discuss only implicit approaches. Our discussion will be brief and far from comprehensive, since a thorough analysis of these processes and their associated problems is not so much a language design issue as it is an

implementation issue.

Single-Size Cells

The simplest situation is when all allocation and deallocation is of single-size cells. It is further simplified when every cell already contains a pointer. This is the scenario of many implementations of Lisp, where the problems of dynamic storage allocation were first encountered on a large scale. All Lisp programs and most Lisp data consist of cells in linked lists.

In a single-size allocation heap, all available cells are linked together using the pointers in the cells, forming a list of available space. Allocation is a - simple matter of taking the required number of cells from this list when they are needed. Deallocation is a much more complex process. A heap-dynamic - variable can be pointed to by more than one pointer, making it difficult to determine when the variable is no longer useful to the program. Simply because one pointer is disconnected from a cell obviously does not make it garbage; there could be several other pointers still pointing to the cell.

In Lisp, several of the most frequent operations in programs create collections of cells that are no longer accessible to the program and therefore should be deallocated (put back on the list of available space). One of the fundamental design goals of Lisp was to ensure that reclamation of unused cells would not be the task of the programmer but rather that of the run-time system. This goal left Lisp implementors with the fundamental design question: When should deallocation be performed?

There are several different approaches to garbage collection. The two most common traditional techniques are in some ways opposite processes. These are named **reference counters**, in which reclamation is incremental and is done when inaccessible cells are created, and **mark-sweep**, in which reclamation occurs only when the list of available space becomes empty. These two methods are sometimes called the **eager approach** and the **lazy approach**, respectively. Many variations of these two approaches have been developed. In this section, however, we discuss only the basic processes.

The reference counter method of storage reclamation accomplishes its goal by maintaining in every cell a counter that stores the number of pointers that are currently pointing at the cell. Embedded in the decrement operation for the reference counters, which occurs when a pointer is disconnected from the cell, is a check for a zero value. If the reference counter reaches zero, it means that no program pointers are pointing at the cell, and it has thus become garbage and can be returned to the list of available space.

There are three distinct problems with the reference counter method. First, if storage cells are relatively small, the space required for the counters is significant. Second, some execution time is obviously required to maintain the counter values. Every time a pointer value is changed, the cell to which it was pointing must have its counter decremented, and the cell to which it is now pointing must have its counter incremented. In a language like Lisp, in which nearly every action involves changing pointers, that can be a significant portion of the total execution time of a program. Of course, if pointer changes are not too frequent, this is not a problem. Some of the inefficiency of reference counters can be eliminated by an approach named **deferred reference counting**, which avoids reference counters for some pointers. The third problem is that complications arise when a collection of cells is connected circularly. The problem here is that each cell in the circular list has a reference counter value of at least 1, which prevents it from being collected and placed back on the list of available space. A solution to this problem can be found in Friedman and Wise (1979).

The advantage of the reference counter approach is that it is intrinsically incremental. Its actions are interleaved with those of the application, so it never causes significant delays in the execution of the application.

The original mark-sweep process of garbage collection operates as follows: The run-time system allocates storage cells as requested and disconnects pointers from cells as necessary, without regard for storage reclamation (allowing garbage to accumulate), until it has allocated all available cells. At this point, a mark-sweep process is begun to gather all the garbage left floating around in the heap. To facilitate the process, every heap cell has an extra indicator bit or field that is used by the collection algorithm.

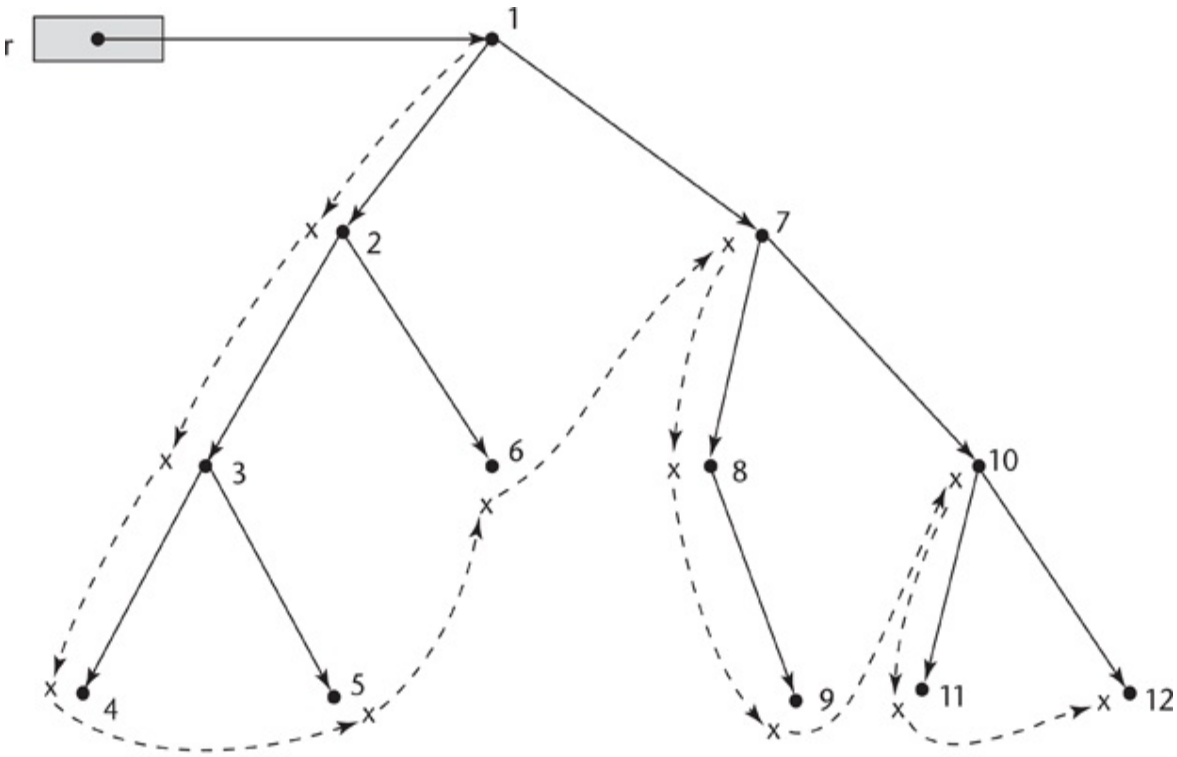
The mark-sweep process consists of three distinct phases. First, all cells in

the heap have their indicators set to indicate they are garbage. This is, of course, a correct assumption for only some of the cells. The second part, called the marking phase, is the most difficult. Every pointer in the program is traced into the heap, and all reachable cells are marked as not being garbage. After this, the third phase, called the sweep phase, is executed: All cells in the heap that have not been specifically marked as still being used are returned to the list of available space.

To illustrate the flavor of algorithms used to mark the cells that are currently in use, we provide the following simple version of a marking algorithm. We assume that all heap-dynamic variables, or heap cells, consist of an information part; a part for the mark, named `marker`; and two pointers named `llink` and `rlink`. These cells are used to build directed graphs with at most two edges leading from any node. The marking algorithm traverses all spanning trees of the graphs, marking all cells that are found. Like other graph traversals, the marking algorithm uses recursion.

```
for every pointer r  do
    mark(r)
void mark(void  * ptr) {
    if (ptr != 0)
      if (*ptr.marker is not marked) {
        set *ptr.marker
        mark(*ptr.llink)
        mark(*ptr.rlink)
       }
}
```

An example of the actions of this procedure on a given graph is shown in Figure 6.9. This simple marking algorithm requires a great deal of storage (for stack space to support recursion). A marking process that does not require additional stack space was developed by Schorr and Waite (1967). Their method reverses pointers as it traces out linked structures. Then, when the end of a list is reached, the process can follow the pointers back out of the structure.



Dashed lines show the order of node_marking

Figure 6.9 An example of the actions of the marking algorithm

Figure 6.9 Full Alternative Text

The most serious problem with the original version of mark-sweep was that it was done too infrequently—only when a program had used all or nearly all of the heap storage. Mark-sweep in that situation takes a good deal of time, because most of the cells must be traced and marked as being currently used. This causes a significant delay in the progress of the application. Furthermore, the process may yield only a small number of cells that can be placed on the list of available space. This problem has been addressed in a

variety of improvements. For example, **incremental mark-sweep** garbage collection occurs more frequently, long before memory is exhausted, making the process more effective in terms of the amount of storage that is reclaimed. Also, the time required for each run of the process is obviously shorter, thus reducing the delay in application execution. Another alternative is to perform the mark-sweep process on parts, rather than all of the memory associated with the application, at different times. This provides the same kinds of improvements as incremental mark-sweep.

Both the marking algorithms for the mark-sweep method and the processes required by the reference counter method can be made more efficient by use of the pointer rotation and slide operations that are described by Suzuki (1982).

Variable-Size Cells

Managing a heap from which variable-size cells[8] are allocated has all the difficulties of managing one for single-size cells, but also has additional problems. Unfortunately, variable-size cells are required by most programming languages. The additional problems posed by variable-size cell management depend on the method used. If mark-sweep is used, the following additional problems occur:

8. The cells have variable sizes because these are abstract cells, which store the values of variables, regardless of their types. Furthermore, a variable could be a structured type.

- The initial setting of the indicators of all cells in the heap to indicate that they are garbage is difficult. Because the cells are different sizes, scanning them is a problem. One solution is to require each cell to have the cell size as its first field. Then the scanning can be done, although it takes slightly more space and somewhat more time than its counterpart for fixed-size cells.

- The marking process is nontrivial. How can a chain be followed from a pointer if there is no predefined location for the pointer in the pointed-to

cell? Cells that do not contain pointers at all are also a problem. Adding an internal pointer to each cell, which is maintained in the background by the run-time system, will work. However, this background maintenance processing adds both space and execution time overhead to the cost of running the program.

- Maintaining the list of available space is another source of overhead. The list can begin with a single cell consisting of all available space. Requests for segments simply reduce the size of this block. Reclaimed cells are added to the list. The problem is that before long, the list becomes a long list of various-size segments, or blocks. This slows allocation because requests cause the list to be searched for sufficiently large blocks. Eventually, the list may consist of a large number of very small blocks, which are not large enough for most requests. At this point, adjacent blocks may need to be collapsed into larger blocks. Alternatives to using the first sufficiently large block on the list can shorten the search but require the list to be ordered by block size. In either case, maintaining the list is additional overhead.

If reference counters are used, the first two problems are avoided, but the available-space list-maintenance problem remains.

For a comprehensive study of memory management problems, see Wilson (2005).

6.12 Optional Types

There are situations in programming when there is a need to be able to indicate that a variable does not currently have a value. Some older languages use zero as a nonvalue for numeric variables. This approach has the disadvantage of not being able to distinguish between when the variable is supposed to have the zero value and when the zero indicates that it has no value. Some newer languages provide types that can have a normal value or a special value to indicate that their variables have no value. Variables that have this capability are called **optional types**. Optional types are now directly supported in C#, F#, and Swift, among others.

C# has two categories of variables, value and reference types. Reference types, which are classes, are optional types by their nature. The **`null`** value indicates that a reference type has no value. Value types, which are all struct types, can be declared to be optional types, which allows them to have the value **`null`**. A variable is declared to be an optional type by following its type name with a question mark `(?)`, as in

```
int? x;
```

To determine whether a variable has a normal value, it can be tested against **`null`**, as in

```
int? x;
. . .
if(x == null)
   Console.WriteLine("x has no value");
else
   Console.WriteLine("The value of x is: {0}", x);
```

Swift's optional types are similar to those of C#, except that the nonvalue is named **`nil`**, instead of **`null`**. The Swift version of the above code is:

```
var Int? x;
. . .
if x == nil
   print("x has no value")
```

```
else
   print("The value of x is: \(x)")
```

6.13 Type Checking

For our discussion of type checking, the concept of operands and operators is generalized to include subprograms and assignment statements. Subprograms will be thought of as operators whose operands are their parameters. The assignment symbol will be thought of as a binary operator, with its target variable and its expression being the operands.

Type checking is the activity of ensuring that the operands of an operator are of compatible types. A **compatible** type is one that either is legal for the operator or is allowed under language rules to be implicitly converted by compiler-generated code (or the interpreter) to a legal type. This automatic conversion is called a **coercion**. For example, if an `int` variable and a `float` variable are added in Java, the value of the `int` variable is coerced to `float` and a floating-point add is done.

A **type error** is the application of an operator to an operand of an inappropriate type. For example, in the original version of C, if an `int` value was passed to a function that expected a `float` value, a type error would occur (because compilers for that language did not check the types of parameters).

If all bindings of variables to types are static in a language, then type checking can nearly always be done statically. Dynamic type binding requires type checking at run time, which is called **dynamic type checking**.

Some languages, such as JavaScript and PHP, because of their dynamic type binding, allow only dynamic type checking. It is better to detect errors at compile time than at run time, because the earlier correction is usually less costly. The penalty for static checking is reduced programmer flexibility. Fewer shortcuts and tricks are possible. Such techniques, though, are now generally recognized to be error prone and detrimental to readability.

Type checking is complicated when a language allows a memory cell to store values of different types at different times during execution. Such memory

cells can be created with C and C++ unions and the discriminated unions of ML, Haskell, and F#. In these cases, type checking, if done, must be dynamic and requires the run-time system to maintain the type of the current value of such memory cells. So, even though all variables are statically bound to types in languages such as C++, not all type errors can be detected by static type checking.

6.14 Strong Typing

One of the ideas in language design that became prominent in the so-called structured-programming revolution of the 1970s was **strong typing**. Strong typing is widely acknowledged as being a highly valuable language characteristic. Unfortunately, it is often loosely defined, and it is sometimes used in computing literature without being defined at all.

A programming language is **strongly typed** if type errors are always detected. This requires that the types of all operands can be determined, either at compile time or at run time. The importance of strong typing lies in its ability to detect all misuses of variables that result in type errors. A strongly typed language also allows the detection, at run time, of uses of the incorrect type values in variables that can store values of more than one type.

C and C++ are not strongly typed languages because both include union types, which are not type checked.

ML is strongly typed, even though the types of some function parameters may not be known at compile time. F# is strongly typed.

Java and C#, although they are based on C++, are nearly strongly typed. Types can be explicitly cast, which could result in a type error. However, there are no implicit ways type errors can go undetected.

The coercion rules of a language have an important effect on the value of type checking. For example, expressions are strongly typed in Java. However, an arithmetic operator with one floating-point operand and one integer operand is legal. The value of the integer operand is coerced to floating-point, and a floating-point operation takes place. This is what is usually intended by the programmer. However, the coercion also results in a loss of one of the benefits of strong typing—error detection. For example, suppose a program had the `int` variables `a` and `b` and the `float` variable `d`. Now, if a programmer meant to type `a + b`, but mistakenly typed `a + d`, the error would not be detected by the compiler. The value of `a` would simply be

coerced to `float`. So, the value of strong typing is weakened by coercion. Languages with a great deal of coercion, like C, and C++, are less reliable than those with no coercion, such as ML and F#. Java and C# have half as many assignment type coercions as C++, so their error detection is better than that of C++, but still not nearly as effective as that of ML and F#. The issue of coercion is examined in detail in Chapter 7.

6.15 Type Equivalence

The idea of type compatibility was defined when the issue of type checking was introduced. The compatibility rules dictate the types of operands that are acceptable for each of the operators and thereby specify the possible type errors of the language.9 The rules are called compatibility because in some cases the type of an operand can be implicitly converted by the compiler or run-time system to make it acceptable for the operator.

9. Type compatibility is also an issue in the relationship between the actual parameters in a subprogram call and the formal parameters of the subprogram definition. This issue is discussed in Chapter 9.

The type compatibility rules are simple and rigid for the predefined scalar types. However, in the cases of structured types, such as arrays and records and some user-defined types, the rules are more complex. Coercion of these types is rare, so the issue is not type compatibility, but type equivalence. That is, two types are **equivalent** if an operand of one type in an expression can be substituted for one of the other type, without coercion. Type equivalence is a strict form of type compatibility—compatibility without coercion. The central issue here is how type equivalence is defined.

The design of the type equivalence rules of a language is important, because it influences the design of the data types and the operations provided for values of those types. With the types discussed here, there are very few predefined operations. Perhaps the most important result of two variables being of equivalent types is that either one can have its value assigned to the other.

There are two approaches to defining type equivalence: name type equivalence and structure type equivalence. **Name type equivalence** means that two variables have equivalent types if they are defined either in the same declaration or in declarations that use the same type name. **Structure type equivalence** means that two variables have equivalent types if their types have identical structures. There are some variations of these two approaches,

and many languages use combinations of them.

Name type equivalence is easy to implement but is more restrictive. Under a strict interpretation, a variable whose type is a subrange of the integers would not be equivalent to an integer type variable. For example, supposing Ada used strict name type equivalence, consider the following Ada code:

```
type Indextype is 1..100;
count : Integer;
index : Indextype;
```

The types of the variables `count` and `index` would not be equivalent; `count` could not be assigned to `index` or vice versa.

Another problem with name type equivalence arises when a structured or user-defined type is passed among subprograms through parameters. Such a type must be defined only once, globally. A subprogram cannot state the type of such formal parameters in local terms. This was the case with the original version of Pascal.

Note that to use name type equivalence, all types must have names. Most languages allow users to define types that are anonymous—they do not have names. For a language to use name type equivalence, such types must implicitly be given internal names by the compiler.

Structure type equivalence is more flexible than name type equivalence, but it is more difficult to implement. Under name type equivalence, only the two type names must be compared to determine equivalence. Under structure type equivalence, however, the entire structures of the two types must be compared. This comparison is not always simple. (Consider a data structure that refers to its own type, such as a linked list.) Other questions can also arise. For example, are two record (or **`struct`**) types equivalent if they have the same structure but different field names? Are two single-dimensioned array types in a language that allows lower bounds of array subscript ranges to be set in their declarations equivalent if they have the same element type but have subscript ranges of `0..10` and `1..11`? Are two enumeration types equivalent if they have the same number of components but spell the literals differently?

Another difficulty with structure type equivalence is that it disallows differentiating between types with the same structure. For example, consider the following Ada-like declarations:

```
type Celsius = Float;
     Fahrenheit = Float;
```

The types of variables of these two types are considered equivalent under structure type equivalence, allowing them to be mixed in expressions, which is surely undesirable in this case, considering the difference indicated by the type's names. In general, types with different names are likely to be abstractions of different categories of problem values and should not be considered equivalent.

Ada uses a restrictive form of name type equivalence but provides two type constructs, subtypes and derived types, that avoid the problems associated with name type equivalence. A **derived type** is a new type that is based on some previously defined type with which it is not equivalent, although it may have identical structure. Derived types inherit all the properties of their parent types. Consider the following example:

```
type Celsius is new Float;
type Fahrenheit is new Float;
```

The types of variables of these two derived types are not equivalent, although their structures are identical. Furthermore, variables of neither type is type equivalent with any other floating-point type. Literals are exempt from the rule. A literal such as `3.0` has the type universal real and is type equivalent to any floating-point type. Derived types can also include range constraints on the parent type, while still inheriting all of the parent's operations.

An Ada **subtype** is a possibly range-constrained version of an existing type. A subtype is type equivalent with its parent type. For example, consider the following declaration:

```
subtype Small_type is Integer range 0..99;
```

The type `Small_type` is equivalent to the type `Integer`.

Note that Ada's derived types are very different from Ada's subrange types. For example, consider the following type declarations:

```
type Derived_Small_Int is new Integer range 1..100;
subtype Subrange_Small_Int is Integer range 1..100;
```

Variables of both types, `Derived_Small_Int` and `Subrange_Small_Int`, have the same range of legal values and both inherit the operations of `Integer`. However, variables of type `Derived_Small_Int` are not compatible with any `Integer` type. On the other hand, variables of type `Subrange_Small_Int` are compatible with variables and constants of `Integer` type and any subtype of `Integer`.

For variables of an Ada unconstrained array type, structure type equivalence is used. For example, consider the following type declaration and two object declarations:

```
type Vector is array (Integer range <>) of Integer;
Vector_1: Vector (1..10);
Vector_2: Vector (11..20);
```

The types of these two objects are equivalent, even though they have different names and different subscript ranges, because for objects of unconstrained array types, structure type equivalence rather than name type equivalence is used. Because both types have 10 elements and the elements of both are of type `Integer`, they are type equivalent.

For constrained anonymous types, Ada uses a highly restrictive form of name type equivalence. Consider the following Ada declarations of constrained anonymous types:

```
A : array (1..10) of Integer;
```

In this case, `A` has an anonymous but unique type assigned by the compiler and unavailable to the program. If we also had

```
B : array (1..10) of Integer;
```

`A` and `B` would be of anonymous but distinct and not equivalent types, though they are structurally identical. The multiple declaration

```
C, D : array (1..10) of Integer;
```

creates two anonymous types, one for `C` and one for `D`, which are not equivalent. This declaration is actually treated as if it were the following two declarations:

```
C : array (1..10) of Integer;
D : array (1..10) of Integer;
```

Note that Ada's form of name type equivalence is more restrictive than the name type equivalence that is defined at the beginning of this section. If we had written instead

```
type List_10 is array (1..10) of Integer;
C, D : List_10;
```

then the types of `C` and `D` would be equivalent.

Name type equivalence works well for Ada, in part because all types, except anonymous arrays, are required to have type names (and anonymous types are given internal names by the compiler).

Type equivalence rules for Ada are more rigid than those for languages that have many coercions among types. For example, the two operands of an addition operator in Java can have virtually any combination of numeric types in the language. One of the operands will simply be coerced to the type of the other. But in Ada, there are no coercions of the operands of an arithmetic operator.

C uses both name and structure type equivalence. Every **`struct`**, **`enum`**, and **`union`** declaration creates a new type that is not equivalent to any other type. So, name type equivalence is used for structure, enumeration, and union types. Other nonscalar types use structure type equivalence. Array types are equivalent if they have the same type components. Also, if an array type has a constant size, it is equivalent either to other arrays with the same constant size or to with those without a constant size. Note that **`typedef`** in C and C++ does not introduce a new type; it simply defines a new name for an existing type. So, any type defined with **`typedef`** is type equivalent to its parent type. One exception to C using name type equivalence for structures,

enumerations, and unions is if two structures, enumerations, or unions are defined in different files, in which case structural type equivalence is used. This is a loophole in the name type equivalence rule to allow equivalence of structures, enumerations, and unions that are defined in different files.

C++ is like C except there is no exception for structures and unions defined in different files.

In languages that do not allow users to define and name types, such as Fortran and COBOL, name equivalence obviously cannot be used.

Object-oriented languages such as Java and C++ bring another kind of type compatibility issue with them. The issue is object compatibility and its relationship to the inheritance hierarchy, which is discussed in Chapter 12.

Type compatibility in expressions is discussed in Chapter 7; type compatibility for subprogram parameters is discussed in Chapter 9.

6.16 Theory and Data Types

Type theory is a broad area of study in mathematics, logic, computer science, and philosophy. It began in mathematics in the early 1900s and later became a standard tool in logic. Any general discussion of type theory is necessarily complex, lengthy, and highly abstract. Even when restricted to computer science, type theory includes such diverse and complex subjects as typed lambda calculus, combinators, the metatheory of bounded quantification, existential types, and higher-order polymorphism. All these topics are far beyond the scope of this book.

In computer science there are two branches of type theory: practical and abstract. The practical branch is concerned with data types in commercial programming languages; the abstract branch primarily focuses on typed lambda calculus, an area of extensive research by theoretical computer scientists over the past half century. This section is restricted to a brief description of some of the mathematical formalisms that underlie data types in programming languages.

A data type defines a set of values and a collection of operations on those values. A **type system** is a set of types and the rules that govern their use in programs. Obviously, every typed programming language defines a type system. The formal model of a type system of a programming language consists of a set of types and a collection of functions that define the type rules of the language, which are used to determine the type of any expression. A formal system that describes the rules of a type system, attribute grammars, is introduced in Chapter 3.

An alternative model to attribute grammars uses a type map and a collection of functions, not associated with grammar rules, that specify the type rules. A type map is similar to the state of a program used in denotational semantics, consisting of a set of ordered pairs, with the first element of each pair being a variable’s name and the second element being its type. A type map is constructed using the type declarations in the program. In a static typed language, the type map need only be maintained during compilation,

although it changes as the program is analyzed by the compiler. If any type checking is done dynamically, the type map must be maintained during execution. The concrete version of a type map in a compilation system is the symbol table, constructed primarily by the lexical and syntax analyzers. Dynamic types sometimes are maintained with tags attached to values or objects.

As stated previously, a data type is a set of values, although in a data type the elements are often ordered. For example, the elements in all enumeration types are ordered. However, in a mathematical set the elements are not ordered. Despite this difference, set operations can be used on data types to describe new data types. The structured data types of programming languages are defined by type operators, or constructors that correspond to set operations. These set operations/type constructors are briefly introduced in the following paragraphs.

A finite mapping is a function from a finite set of values, the domain set, onto values in the range set. Finite mappings model two different categories of types in programming languages, functions and arrays, although in some languages functions are not types. All languages include arrays, which are defined in terms of a mapping function that maps indices to elements in the array. For traditional arrays, the mapping is simple—integer values are mapped to the addresses of array elements; for associative arrays, the mapping is defined by a function that describes a hashing operation. The hashing function maps the keys of the associate arrays, usually character strings,10 to the addresses of the array elements.

10. In Ruby, the associative array keys need not be character strings—they can be any type.

A Cartesian, or cross product of n sets, S1, S2, …, Sn, is a set denoted $S1 \times S2 \times \ldots \times Sn$. Each element of the Cartesian product set has one element from each of the constituent sets. So, $S1 \times S2 = \{(x, y) \mid x \text{ is in } S1 \text{ and } y \text{ is in } S2\}$. For example, if $S1 = \{1, 2\}$ and $S2 = \{a, b\}$, $S1 \times S2 = \{(1, a), (1, b), (2, a), (2, b)\}$. A Cartesian product defines tuples in mathematics, which appear in Python, ML, Swift, and F# as a data type (see Section 6.5). Cartesian products also model records, or structs, although not exactly. Cartesian products do not have element names, but records require

them. For example, consider the following C struct:

```
struct intFloat {
  int myInt;
  float myFloat;
};
```

This struct defines the Cartesian product type **`int`** × **`float`**. The names of the elements are `myInt` and `myFloat`.

The union of two sets, S1 and S2, is defined as S1 ∪ S2={x | x is in S1 or x is in S2}. Set union models the union data types, as described in Section 6.10.

Mathematical subsets are defined by providing a rule that elements must follow. These sets model the subtypes of Ada, although not exactly, because subtypes must consist of contiguous elements of their parent sets. Elements of mathematical sets are unordered, so the model is not perfect.

Notice that pointers, defined with type operators, such as the asterisk in C, are not defined in terms of a set operation.

This concludes our discussion of formalisms in data types, as well as our whole discussion of data types.

SUMMARY

The data types of a language are a large part of what determines that language's style and usefulness. Along with control structures, they form the heart of a language.

The primitive data types of most imperative languages include numeric, character, and Boolean types. The numeric types are often directly supported by hardware.

The user-defined enumeration and subrange types are convenient and add to the readability and reliability of programs.

Arrays are part of most programming languages. The relationship between a reference to an array element and the address of that element is given in an access function, which is an implementation of a mapping. Arrays can be either static, as in C++ arrays whose definition includes the **`static`** specifier; fixed stack-dynamic, as in C functions (without the **`static`** specifier); fixed heap-dynamic, as with Java's objects; or heap dynamic, as in Perl's arrays. Most languages allow only a few operations on complete arrays.

Records are now included in most languages. Fields of records are specified in a variety of ways. In the case of COBOL, they can be referenced without naming all of the enclosing records, although this is messy to implement and harmful to readability. In several languages that support object-oriented programming, records are supported with objects.

Tuples are similar to records, but do not have names for their constituent. parts. They are part of Python, ML, and F#.

Lists are staples of the functional programming languages, but are now also included in Python and C#.

Unions are locations that can store different type values at different times. Discriminated unions include a tag to record the current type value. A free union is one without the tag. Most languages with unions do not have safe

designs for them, the exceptions being ML, Swift, and F#.

Pointers are used for addressing flexibility and to control dynamic storage management. Pointers have some inherent dangers: Dangling pointers are difficult to avoid, and memory leakage can occur.

Reference types, such as those in Java and C#, provide heap management without the dangers of pointers.

Enumeration and record types are relatively easy to implement. Arrays are also straightforward, although array element access is an expensive process when the array has several subscripts. The access function requires one addition and one multiplication for each subscript.

Pointers are relatively easy to implement, if heap management is not considered. Heap management is easy if all cells have the same size but is complicated for variable-size cell allocation and deallocation.

Optional type variables are variables that allow a nonvalue to be stored. This allows a program to indicate when a variable currently has no value.

Strong typing is the concept of requiring that all type errors be detected. The value of strong typing is increased reliability.

The type equivalence rules of a language determine what operations are legal among the structured types of a language. Name type equivalence and structure type equivalence are the two fundamental approaches to defining type equivalence.

Type theories have been developed in many areas. In computer science, the practical branch of type theory defines the types and type rules of programming languages. Set theory can be used to model most of the structured data types in programming languages.

BIBLIOGRAPHIC NOTES

A wealth of literature exists that is concerned with data type design, use, and implementation. Hoare gives one of the earliest systematic definitions of structured types in Dahl et al. (1972). A general discussion of a wide variety of data types is given in Cleaveland (1986).

Implementing run-time checks on the possible insecurities of Pascal data types is discussed in Fischer and LeBlanc (1980). Most compiler design books, such as Fischer and LeBlanc (1991) and Aho et al. (1986), describe implementation methods for data types, as do the other programming language texts, such as Pratt and Zelkowitz (2001) and Scott (2009). A detailed discussion of the problems of heap management can be found in Tenenbaum et al. (1990). Garbage-collection methods are developed by Schorr and Waite (1967) and Deutsch and Bobrow (1976). A comprehensive discussion of garbage-collection algorithms can be found in Cohen (1981) and Wilson (2005).

REVIEW QUESTIONS

1. What is a descriptor?
2. What are the advantages and disadvantages of decimal data types?
3. What are the design issues for character string types?
4. Describe the three string length options.
5. Define *ordinal*, *enumeration*, and *subrange types*.
6. What are the advantages of user-defined enumeration types?
7. In what ways are the user-defined enumeration types of C# more reliable than those of C++?
8. What are the design issues for arrays?
9. Define *static*, *fixed stack-dynamic*, *fixed heap-dynamic*, and *heap-dynamic arrays*. What are the advantages of each?
10. What happens when a nonexistent element of an array is referenced in Perl?
11. How does JavaScript support sparse arrays?
12. What languages support negative subscripts?
13. What languages support array slices with stepsizes?
14. What is an aggregate constant?
15. Define *row major order* and *column major order*.
16. What is an access function for an array?

17. What are the required entries in a Java array descriptor, and when must they be stored (at compile time or run time)?

18. What is the structure of an associative array?

19. What is the purpose of level numbers in COBOL records?

20. Define *fully qualified* and *elliptical references* to fields in records.

21. What is the primary difference between a record and a tuple?

22. Are the tuples of Python mutable?

23. What is the purpose of an F# tuple pattern?

24. In what primarily imperative language do lists serve as arrays?

25. What is the action of the Scheme function `CAR`?

26. What is the action of the F# function `tl`?

27. In what way does Scheme's `CDR` function modify its parameter?

28. On what are Python's list comprehensions based?

29. Define *union*, *free union*, and *discriminated union*.

30. Are the unions of F# discriminated?

31. What are the design issues for pointer types?

32. What are the two common problems with pointers?

33. Why are the pointers of most languages restricted to pointing at a single type variable?

34. What is a C++ reference type, and what is its common use?

35. Why are reference variables in C++ better than pointers for formal

parameters?

36. What advantages do Java and C# reference type variables have over the pointers in other languages?

37. Describe the lazy and eager approaches to reclaiming garbage.

38. Why wouldn't arithmetic on Java and C# references make sense?

39. What is a compatible type?

40. Define type error.

41. Define strongly typed.

42. Why is Java not strongly typed?

43. What is a nonconverting cast?

44. What languages have no type coercions?

45. Why are C and C++ not strongly typed?

46. What is name type equivalence?

47. What is structure type equivalence?

48. What is the primary advantage of name type equivalence?

49. What is the primary disadvantage to structure type equivalence?

50. For what types does C use structure type equivalence?

51. What set operation models C's `struct` data type?

PROBLEM SET

1. What are the arguments for and against representing Boolean values as single bits in memory?

2. How does a decimal value waste memory space?

3. VAX minicomputers use a format for floating-point numbers that is not the same as the IEEE standard. What is this format, and why was it chosen by the designers of the VAX computers? A reference for VAX floating-point representations is Sebesta (1991).

4. Compare the tombstone and lock-and-key methods of avoiding dangling pointers, from the points of view of safety and implementation cost.

5. What disadvantages are there in implicit dereferencing of pointers, but only in certain contexts?

6. Explain all of the differences between Ada's subtypes and derived types.

7. What significant justification is there for the `->` operator in C and C++?

8. What are all of the differences between the enumeration types of C++ and those of Java?

9. Multidimensional arrays can be stored in row major order, as in C++, or in column major order, as in Fortran. Develop the access functions for both of these arrangements for three-dimensional arrays.

10. In the Burroughs Extended ALGOL language, matrices are stored as a single-dimensioned array of pointers to the rows of the matrix, which are treated as single-dimensioned arrays of values. What are the advantages and disadvantages of such a scheme?

11. Analyze and write a comparison of C's `malloc` and `free` functions with C++'s **`new`** and **`delete`** operators. Use safety as the primary

consideration in the comparison.

12. Analyze and write a comparison of using C++ pointers and Java reference variables to refer to fixed heap-dynamic variables. Use safety and convenience as the primary considerations in the comparison.

13. Write a short discussion of what was lost and what was gained in Java's designers' decision to not include the pointers of C++.

14. What are the arguments for and against Java's implicit heap storage recovery, when compared with the explicit heap storage recovery required in C++? Consider real-time systems.

15. What are the arguments for the inclusion of enumeration types in C#, although they were not in the first few versions of Java?

16. What would you expect to be the level of use of pointers in C#? How often will they be used when it is not absolutely necessary?

17. Make two lists of applications of matrices, one for those that require jagged matrices and one for those that require rectangular matrices. Now, argue whether just jagged, just rectangular, or both should be included in a programming language.

18. Compare the string manipulation capabilities of the class libraries of C++, Java, and C#.

19. Look up the definition of *strongly typed* as given in Gehani (1983) and compare it with the definition given in this chapter. How do they differ?

20. In what way is static type checking better than dynamic type checking?

21. Explain how coercion rules can weaken the beneficial effect of strong typing.

PROGRAMMING EXERCISES

1. Design a set of simple test programs to determine the type compatibility rules of a C compiler to which you have access. Write a report of your findings.

2. Determine whether some C compiler to which you have access implements the `free` function.

3. Write a program that does matrix multiplication in some language that does subscript range checking and for which you can obtain an assembly language or machine language version from the compiler. Determine the number of instructions required for the subscript range checking and compare it with the total number of instructions for the matrix multiplication process.

4. If you have access to a compiler in which the user can specify whether subscript range checking is desired, write a program that does a large number of matrix accesses and time their execution. Run the program with subscript range checking and without it, and compare the times.

5. Write a simple program in C++ to investigate the safety of its enumeration types. Include at least 10 different operations on enumeration types to determine what incorrect or just silly things are legal. Now, write a C# program that does the same things and run it to determine how many of the incorrect or silly things are legal. Compare your results.

6. Write a program in C++ or C# that includes two different enumeration types and has a significant number of operations using the enumeration types. Also write the same program using only integer variables. Compare the readability and predict the reliability differences between the two programs.

7. Write a C program that does a large number of references to elements of

two-dimensioned arrays, using only subscripting. Write a second program that does the same operations but uses pointers and pointer arithmetic for the storage-mapping function to do the array references. Compare the time efficiency of the two programs. Which of the two programs is likely to be more reliable? Why?

8. Write a Perl program that uses a hash and a large number of operations on the hash. For example, the hash could store people's names and their ages. A random-number generator could be used to create three-character names and ages, which could be added to the hash. When a duplicate name was generated, it would cause an access to the hash but not add a new element. Rewrite the same program without using hashes. Compare the execution efficiency of the two. Compare the ease of programming and readability of the two.

9. Write a program in the language of your choice that behaves differently if the language used name equivalence than if it used structural equivalence.

10. For what types of A and B is the simple assignment statement A=B legal in C++ but not Java?

7 Expressions and Assignment Statements

As the title indicates, the topic of this chapter is expressions and assignment statements. The semantics rules that determine the order of evaluation of operators in expressions are discussed first. This is followed by an explanation of a potential problem with operand evaluation order when functions can have side effects. Overloaded operators, both predefined and user defined, are then discussed, along with their effects on the expressions in programs. Next, mixed-mode expressions are described and evaluated. This leads to the definition and evaluation of widening and narrowing type conversions, both implicit and explicit. Relational and Boolean expressions are then discussed, including the process of short-circuit evaluation. Finally, the assignment statement, from its simplest form to all of its variations, is - covered, including assignments as expressions and mixed-mode assignments.

Character string pattern-matching expressions were covered as a part of the - material on character strings in Chapter 6, so they are not mentioned in this chapter.

7.1 Introduction

Expressions are the fundamental means of specifying computations in a programming language. It is crucial for a programmer to understand both the syntax and semantics of expressions of the language he or she uses. A formal mechanism (BNF) for describing the syntax of expressions was introduced in Chapter 3. In this chapter, the semantics of expressions are discussed.

To understand expression evaluation, it is necessary to be familiar with the orders of operator and operand evaluation. The operator evaluation order of expressions is dictated by the associativity and precedence rules of the language. Although the value of an expression sometimes depends on it, the order of operand evaluation in expressions is often unstated by language designers. This allows implementors to choose the order, which leads to the possibility of programs producing different results in different implementations. Other issues in expression semantics are type mismatches, coercions, and short-circuit evaluation.

The essence of the imperative programming languages is the dominant role of assignment statements. The purpose of these statements is to cause the side effect of changing the values of variables, or the state, of the program. So an integral part of all imperative languages is the concept of variables whose values change during program execution.

Functional languages use variables of a different sort, such as the parameters of functions. These languages also have declaration statements that bind values to names. These declarations are similar to assignment statements, but do not have side effects.

7.2 Arithmetic Expressions

Automatic evaluation of arithmetic expressions similar to those found in mathematics, science, and engineering was one of the primary goals of the first high-level programming languages. Most of the characteristics of arithmetic expressions in programming languages were inherited from conventions that had evolved in mathematics. In programming languages, arithmetic expressions consist of operators, operands, parentheses, and function calls. An operator can be **unary**, meaning it has a single operand, **binary**, meaning it has two operands, or **ternary**, meaning it has three operands.

In most programming languages, binary operators are **infix**, which means they appear between their operands. One exception is Perl, which has some operators that are **prefix**, which means they precede their operands. In Scheme and Lisp, all operators are prefix. Most unary operators are prefix, but the ++ and -- operators of C-based languages can be either prefix or postfix.

The purpose of an arithmetic expression is to specify an arithmetic computation. An implementation of such a computation must cause two actions: fetching the operands, usually from memory, and executing arithmetic operations on those operands. In the following sections, we investigate the common design details of arithmetic expressions.

Following are the primary design issues for arithmetic expressions, all of which are discussed in this section:

- What are the operator precedence rules?
- What are the operator associativity rules?
- What is the order of operand evaluation?
- Are there restrictions on operand evaluation side effects?

- Does the language allow user-defined operator overloading?
- What type mixing is allowed in expressions?

7.2.1 Operator Evaluation Order

The operator precedence and associativity rules of a language dictate the order of evaluation of its operators.

7.2.1.1 Precedence

The value of an expression depends at least in part on the order of evaluation of the operators in the expression. Consider the following expression:

```
a + b * c
```

Suppose the variables `a`, `b`, and `c` have the values `3`, `4`, and `5`, respectively. If evaluated left to right (the addition first and then the multiplication), the result is `35`. If evaluated right to left, the result is `23`.

Instead of simply evaluating the operators in an expression from left to right or right to left, mathematicians long ago developed the concept of placing operators in a hierarchy of evaluation priorities and basing the evaluation order of expressions partly on this hierarchy. For example, in mathematics, multiplication is considered to be of higher priority than addition, perhaps due to its higher level of complexity. If that convention were applied in the previous example expression, as would be the case in most programming languages, the multiplication would be done first.

The **operator precedence rules** for expression evaluation partially define the order in which the operators of different precedence levels are evaluated. The operator precedence rules for expressions are based on the hierarchy of operator priorities, as seen by the language designer. The operator precedence rules of the common imperative languages are nearly all the same, because they are based on those of mathematics. In these languages, exponentiation

has the highest precedence (when it is provided by the language), followed by multiplication and division on the same level, followed by binary addition and subtraction on the same level.

Many languages also include unary versions of addition and subtraction. Unary addition is called the **identity operator** because it usually has no associated operation and thus has no effect on its operand. Ellis and Stroustrup (1990, p. 56), speaking about C++, call it a historical accident and correctly label it useless. Unary minus, of course, changes the sign of its operand. In Java and C#, unary minus also causes the implicit conversion of `short` and `byte` operands to `int` type.

In all of the common imperative languages, the unary minus operator can appear in an expression either at the beginning or anywhere inside the expression, as long as it is parenthesized to prevent it from being next to another operator. For example,

```
a + (- b) * c
```

is legal, but

```
a + - b * c
```

usually is not.

Next, consider the following expressions:

```
- a / b
- a * b
- a ** b
```

In the first two cases, the relative precedence of the unary minus operator and the binary operator is irrelevant—the order of evaluation of the two operators has no effect on the value of the expression. In the last case, however, it does matter.

Of the common programming languages, only Fortran, Ruby, Visual Basic, and Ada have the exponentiation operator. In all four, exponentiation has higher precedence than unary minus, so

```
- A ** B
```

is equivalent to

```
-(A ** B)
```

The precedences of the arithmetic operators of Ruby and the C-based languages are as follows:

	Ruby	*C-Based Languages*
Highest	**	postfix ++, --
	unary +, -	prefix ++, --, unary +, -
	*, /, %	*, /, %
Lowest	binary +, -	binary +, -

The ** operator is exponentiation. The % operator takes two integer operands and yields the remainder of the first after division by the second.[1] The ++ and -- operators of the C-based languages are described in Section 7.7.4.

1. In versions of C before C99, the % operator was implementation dependent in some situations, because division was also implementation dependent.

APL is odd among languages because it has a single level of precedence, as illustrated in the next section.

Precedence accounts for only some of the rules for the order of operator evaluation; associativity rules also affect it.

7.2.1.2 Associativity

Consider the following expression:

```
a - b + c - d
```

If the addition and subtraction operators have the same level of precedence,

as they do in programming languages, the precedence rules say nothing about the order of evaluation of the operators in this expression.

When an expression contains two adjacent[2] occurrences of operators with the same level of precedence, the question of which operator is evaluated first is answered by the **associativity** rules of the language. An operator can have either left or right associativity, meaning that when there are two adjacent operators with the same precedence, the left operator is evaluated first or the right operator is evaluated first, respectively.

2. We call operators "adjacent" if they are separated by a single operand.

Associativity in common languages is left to right, except that the exponentiation operator (when provided) sometimes associates right to left. In the Java expression

```
a - b + c
```

the left operator is evaluated first.

Exponentiation in Fortran and Ruby is right associative, so in the expression

```
A ** B ** C
```

the right operator is evaluated first.

In Visual Basic, the exponentiation operator, ^, is left associative.

The associativity rules for a few common languages are given here:

Language	*Associativity Rule*
Ruby	Left: *, /, +, -
	Right: **
C-based languages	Left: *, /, %, binary +, binary -
	Right: ++, --, unary -, unary +

As stated in Section 7.2.1.1, in APL, all operators have the same level of precedence. Thus, the order of evaluation of operators in APL expressions is determined entirely by the associativity rule, which is right to left for all operators. For example, in the expression

```
A × B + C
```

the addition operator is evaluated first, followed by the multiplication operator (× is the APL multiplication operator). If `A` were `3`, `B` were `4`, and `C` were `5`, then the value of this APL expression would be `27`.

Many compilers for the common languages make use of the fact that some arithmetic operators are mathematically associative, meaning that the associativity rules have no impact on the value of an expression containing only those operators. For example, addition is mathematically associative, so in mathematics the value of the expression

```
A + B + C
```

does not depend on the order of operator evaluation. If floating-point operations for mathematically associative operations were also associative, the compiler could use this fact to perform some simple optimizations. Specifically, if the compiler is allowed to reorder the evaluation of operators, it may be able to produce slightly faster code for expression evaluation. Compilers commonly do these kinds of optimizations.

Unfortunately, in a computer, both floating-point representations and floating-point arithmetic operations are only approximations of their mathematical counterparts (because of size limitations). The fact that a mathematical operator is associative does not necessarily imply that the corresponding computer floating-point operation is associative. In fact, only if all the operands and intermediate results can be exactly represented in floating-point notation will the process be precisely associative. For example, there are pathological situations in which integer addition on a computer is *not* associative. For example, suppose that a program must evaluate the expression

```
A + B + C + D
```

and that `A` and `C` are very large positive numbers, and `B` and `D` are negative numbers with very large absolute values. In this situation, adding `B` to `A` does not cause an overflow exception, but adding `C` to `A` does. Likewise, adding `C` to `B` does not cause overflow, but adding `D` to `B` does. Because of the limitations of computer arithmetic, addition is catastrophically nonassociative in this case. Therefore, if the compiler reorders these addition operations, it affects the value of the expression. This problem, of course, can be avoided by the programmer, assuming the approximate values of the variables are known. The programmer can specify the expression in two parts (in two assignment statements), ensuring that overflow is avoided. However, this situation can arise in far more subtle ways, in which the programmer is less likely to notice the order dependence.

7.2.1.3 Parentheses

Programmers can alter the precedence and associativity rules by placing parentheses in expressions. A parenthesized part of an expression has precedence over its adjacent unparenthesized parts. For example, although multiplication has precedence over addition, in the expression

```
(A + B) * C
```

the addition will be evaluated first. Mathematically, this is perfectly natural. In this expression, the first operand of the multiplication operator is not available until the addition in the parenthesized subexpression is evaluated. Also, the expression from Section 7.2.1.2 could be specified as

```
(A + B) + (C + D)
```

to avoid overflow.

Languages that allow parentheses in arithmetic expressions could dispense with all precedence rules and simply associate all operators left to right or right to left. The programmer would specify the desired order of evaluation with parentheses. This approach would be simple because neither the author nor the readers of programs would need to remember any precedence or associativity rules. The disadvantage of this scheme is that it makes writing

expressions more tedious, and it also seriously compromises the readability of the code. Yet this was the choice made by Ken Iverson, the designer of APL.

7.2.1.4 Ruby Expressions

Recall that Ruby is a pure object-oriented language, which means, among other things, that every data value, including literals, is an object. Ruby supports the collection of arithmetic and logic operations that are included in the C-based languages. What sets Ruby apart from the C-based languages in the area of expressions is that all of the arithmetic, relational, and assignment operators, as well as array indexing, shifts, and bitwise logic operators, are implemented as methods. For example, the expression `a + b` is a call to the `+` method of the object referenced by `a`, passing the object referenced by `b` as a parameter.

One interesting result of the implementation of operators as methods is that they can be overridden by application programs. Therefore, these operators can be redefined. While it is often not useful to redefine operators for predefined types, it is useful, as we will see in Section 7.3, to define predefined operators for user-defined types, which can be done with operator overloading in some languages.

In C++ and Ada, operators are actually implemented as function calls.

7.2.1.5 Expressions in Lisp

As is the case with Ruby, all arithmetic and logic operations in Lisp are performed by subprograms. But in Lisp, the subprograms must be explicitly called. For example, to specify the C expression `a + b * c` in Lisp, one must write the following expression:[3]

3. When a list is interpreted as code in Lisp, the first element is the function name and others are parameters to the function.

```
(+ a (* b c))
```

In this expression, `+` and `*` are the names of functions.

7.2.1.6 Conditional Expressions

`if-then-else` statements can be used to perform a conditional expression assignment. For example, consider

```
if  (count == 0)
  average = 0;
else
  average = sum / count;
```

In the C-based languages, this code can be specified more conveniently in an assignment statement using a conditional expression, which has the following form:

- expression_1 ? expression_2 : expression_3

where expression_1 is interpreted as a Boolean expression. If expression_1 evaluates to true, the value of the whole expression is the value of expression_2; otherwise, it is the value of expression_3. For example, the effect of the example **`if-then-else`** can be achieved with the following assignment statement, using a conditional expression:

```
average = (count == 0) ? 0 : sum / count;
```

In effect, the question mark denotes the beginning of the **`then`** clause, and the colon marks the beginning of the **`else`** clause. Both clauses are mandatory. Note that **`?`** is used in conditional expressions as a ternary operator.

Conditional expressions can be used anywhere in a program (in a C-based language) where any other expression can be used. In addition to the C-based languages, conditional expressions are provided in Perl, JavaScript, and Ruby.

7.2.2 Operand Evaluation Order

A less commonly discussed design characteristic of expressions is the order of evaluation of operands. Variables in expressions are evaluated by fetching their values from memory. Constants are sometimes evaluated the same way. In other cases, a constant may be part of the machine language instruction and not require a memory fetch. If an operand is a parenthesized expression, all of the operators it contains must be evaluated before its value can be used as an operand.

If neither of the operands of an operator has side effects, then operand evaluation order is irrelevant. Therefore, the only interesting case arises when the evaluation of an operand does have side effects.

7.2.2.1 Side Effects

A **side effect** of a function, naturally called a functional side effect, occurs when the function changes either one of its parameters or a global variable. (A global variable is declared outside the function but is accessible in the function.)

Consider the following expression:

```
a + fun(a)
```

If `fun` does not have the side effect of changing `a`, then the order of evaluation of the two operands, `a` and `fun(a)`, has no effect on the value of the expression. However, if `fun` changes `a`, there is an effect. Consider the following situation: `fun` returns `10` and changes the value of its parameter to `20`. Suppose we have the following:

```
a = 10;
b = a + fun(a);
```

Then, if the value of `a` is fetched first (in the expression evaluation process), its value is `10` and the value of the expression is `20`. But if the second operand

is evaluated first, then the value of the first operand is `20` and the value of the expression is `30`.

The following C program illustrates the same problem when a function changes a global variable that appears in an expression:

```
int a = 5;
int fun1() {
   a = 17;
   return  3;
}   /* end of fun1 */
 void  main() {
   a = a + fun1();
}   /* end of main */
```

The value computed for `a` in `main` depends on the order of evaluation of the operands in the expression `a + fun1()`. The value of `a` will be either `8` (if `a` is evaluated first) or `20` (if the function call is evaluated first).

Note that functions in mathematics do not have side effects, because there is no notion of variables in mathematics. The same is true for functional programming languages. In both mathematics and functional programming languages, functions are much easier to reason about and understand than those in imperative languages, because their context is irrelevant to their meaning.

There are two possible solutions to the problem of operand evaluation order and side effects. First, the language designer could disallow function evaluation from affecting the value of expressions by simply disallowing functional side effects. Second, the language definition could state that operands in expressions are to be evaluated in a particular order and demand that implementors guarantee that order.

Disallowing functional side effects in the imperative languages is difficult, and it eliminates some flexibility for the programmer. Consider the case of C and C++, which have only functions, meaning that all subprograms return one value. To eliminate the side effects of two-way parameters and still provide subprograms that return more than one value, the values would need to be placed in a struct and the struct returned. Access to globals in functions would also have to be disallowed. However, when efficiency is important,

using access to global variables to avoid parameter passing is an important method of increasing execution speed. In compilers, for example, global access to data such as the symbol table is commonplace.

The problem with having a strict evaluation order is that some code optimization techniques used by compilers involve reordering operand evaluations. A guaranteed order disallows those optimization methods when function calls are involved. There is, therefore, no perfect solution, as is borne out by actual language designs.

The Java language definition guarantees that operands appear to be evaluated in left-to-right order, eliminating the problem discussed in this section.

7.2.2.2 Referential Transparency and Side Effects

The concept of referential transparency is related to and affected by functional side effects. A program has the property of **referential transparency** if any two expressions in the program that have the same value can be substituted for one another anywhere in the program, without affecting the action of the program. The value of a referentially transparent function depends entirely on its parameters.4 The connection of referential transparency and functional side effects is illustrated by the following example:

4. Furthermore, the value of the function cannot depend on the order in which its parameters are evaluated.

```
result1 = (fun(a) + b) / (fun(a) - c);
temp = fun(a);
result2 = (temp + b) / (temp - c);
```

If the function `fun` has no side effects, `result1` and `result2` will be equal, because the expressions assigned to them are equivalent. However, suppose `fun` has the side effect of adding 1 to either `b` or `c`. Then `result1` would not be equal to `result2`. So, that side effect violates the referential transparency

of the program in which the code appears.

There are several advantages to referentially transparent programs. The most important of these is that the semantics of such programs is much easier to understand than the semantics of programs that are not referentially transparent. Being referentially transparent makes a function equivalent to a mathematical function, in terms of ease of understanding.

Because they do not have variables, programs written in pure functional languages are referentially transparent. Functions in a pure functional language cannot have state, which would be stored in local variables. If such a function uses a value from outside the function, that value must be a constant, since there are no variables. Therefore, the value of the function depends on the values of its parameters.

Referential transparency will be further discussed in Chapter 15.

7.3 Overloaded Operators

Arithmetic operators are often used for more than one purpose. For example, + usually is used to specify integer addition and floating-point addition. Some languages—Java, for example—also use it for string catenation. This multiple use of an operator is called **operator overloading** and is generally thought to be acceptable, as long as neither readability nor reliability suffers.

As an example of the possible dangers of overloading, consider the use of the ampersand (&) in C++. As a binary operator, it specifies a bitwise logical AND operation. As a unary operator, however, its meaning is totally different. As a unary operator with a variable as its operand, the expression value is the address of that variable. In this case, the ampersand is called the address-of operator. For example, the execution of

```
x = &y;
```

causes the address of y to be placed in x. There are two problems with this multiple use of the ampersand. First, using the same symbol for two completely unrelated operations is detrimental to readability. Second, the simple keying error of leaving out the first operand for a bitwise AND operation can go undetected by the compiler, because it is interpreted as an address-of operator. Such an error may be difficult to diagnose.

Virtually all programming languages have a less serious but similar problem, which is often due to the overloading of the minus operator. The problem is only that the compiler cannot tell if the operator is meant to be binary or unary.5 So once again, failure to include the first operand when the operator is meant to be binary cannot be detected as an error by the compiler. However, the meanings of the two operations, unary and binary, are at least closely related, so readability is not adversely affected.

5. ML alleviates this problem by using different symbols for unary and binary minus operators, tilde (') for unary and dash for binary.

Some languages that support abstract data types (see Chapter 11), for

example, C++, C#, and F#, allow the programmer to further overload operator symbols. For instance, suppose a user wants to define the * operator between a scalar integer and an integer array to mean that each element of the array is to be multiplied by the scalar. Such an operator could be defined by writing a function subprogram named * that performs this new operation. The compiler will choose the correct meaning when an overloaded operator is specified, based on the types of the operands, as with language-defined overloaded operators. For example, if this new definition for * is defined in a C# program, a C# compiler will use the new definition for * whenever the * operator appears with a simple integer as the left operand and an integer array as the right operand.

When sensibly used, user-defined operator overloading can aid readability. For example, if + and * are overloaded for a matrix abstract data type and A, B, C, and D are variables of that type, then

```
A * B + C * D
```

can be used instead of

```
MatrixAdd(MatrixMult(A, B), MatrixMult(C, D))
```

On the other hand, user-defined overloading can be harmful to readability. For one thing, nothing prevents a user from defining + to mean multiplication. Furthermore, seeing an * operator in a program, the reader must find both the types of the operands and the definition of the operator to determine its meaning. Any or all of these definitions could be in other files.

Another consideration is the process of building a software system from modules created by different groups. If the different groups overloaded the same operators in different ways, these differences would obviously need to be eliminated before putting the system together.

C++ has a few operators that cannot be overloaded. Among these are the class or structure member operator (.) and the scope resolution operator (::). Interestingly, operator overloading was one of the C++ features that was not copied into Java. However, it did reappear in C#.

The implementation of user-defined operator overloading is discussed in

Chapter 9.

7.4 Type Conversions

Type conversions are either narrowing or widening. A **narrowing conversion** converts a value to a type that cannot store even approximations of all of the values of the original type. For example, converting a `double` to a `float` in Java is a narrowing conversion, because the range of `double` is much larger than that of `float`. A **widening conversion** converts a value to a type that can include at least approximations of all of the values of the original type. For example, converting an `int` to a `float` in Java is a widening conversion. Widening conversions are nearly always safe, meaning that the approximate magnitude of the converted value is maintained. Narrowing conversions are not always safe—sometimes the magnitude of the converted value is changed in the process. For example, if the floating-point value 1.3E25 is converted to an integer in a Java program, the result will not be in any way related to the original value.

Although widening conversions are usually safe, they can result in reduced accuracy. In many language implementations, although integer-to-floating-point conversions are widening conversions, some precision may be lost. For example, in many cases, integers are stored in 32 bits, which allows at least 9 decimal digits of precision. But floating-point values are also stored in 32 bits, with only about seven decimal digits of precision (because of the space used for the exponent). So, integer-to-floating-point widening can result in the loss of two digits of precision.

Coercions of nonprimitive types are, of course, more complex. In Chapter 5, the complications of assignment compatibility of array and record types were discussed. There is also the question of what parameter types and return types of a method allow it to override a method in a superclass—only when the types are the same, or also some other situations. That issue, as well as the concept of subclasses as subtypes, are discussed in Chapter 12.

Type conversions can be either explicit or implicit. The following two subsections discuss these kinds of type conversions.

7.4.1 Coercion in Expressions

One of the design decisions concerning arithmetic expressions is whether an operator can have operands of different types. Languages that allow such expressions, which are called **mixed-mode expressions**, must define conventions for implicit operand type conversions because computers do not have binary operations that take operands of different types. Recall that in Chapter 5, coercion was defined as an implicit type conversion that is initiated by the compiler or runtime system. Type conversions explicitly requested by the programmer are referred to as explicit conversions, or casts, not coercions.

Although some operator symbols may be overloaded, we assume that a computer system, either in hardware or in some level of software simulation, has an operation for each operand type and operator defined in the language.6 For overloaded operators in a language that uses static type binding, the compiler chooses the correct type of operation on the basis of the types of the operands. When the two operands of an operator are not of the same type and that is legal in the language, the compiler must choose one of them to be coerced and generate the code for that coercion. In the following discussion, the coercion design choices of several common languages are examined.

6. This assumption is not true for many languages. An example is given later in this section.

Language designers are not in agreement on the issue of coercions in arithmetic expressions. Those against a broad range of coercions are concerned with the reliability problems that can result from such coercions, because they reduce the benefits of type checking. Those who would rather include a wide range of coercions are more concerned with the loss in flexibility that results from restrictions. The issue is whether programmers should need to be concerned with this category of errors or whether the compiler should detect them.

As a simple illustration of the problem, consider the following Java code:

```
int a;
```

```
float b, c, d;
. . .
d = b * a;
```

Assume that the second operand of the multiplication operator was supposed to be `c`, but because of a keying error it was typed as `a`. Because mixed-mode expressions are legal in Java, the compiler would not detect this as an error. It would simply insert code to coerce the value of the **`int`** operand, `a`, to **`float`**. If mixed-mode expressions were not legal in Java, this keying error would have been detected by the compiler as a type error.

Because error detection is reduced when mixed-mode expressions are allowed, F#, Ada, and ML do not allow them. For example, they do not allow mixing of integer and floating-point operands in expressions.

In most of the other common languages, there are no restrictions on mixed-mode arithmetic expressions.

The C-based languages have integer types that are smaller than the **`int`** type. In Java, these are **`byte`** and **`short`**. Operands of all of these types are coerced to **`int`** whenever virtually any operator is applied to them. So, while data can be stored in variables of these types, it cannot be manipulated before conversion to a larger type. For example, consider the following Java code:

history note

As a more extreme example of the dangers and costs of too much coercion, consider PL/I's efforts to achieve flexibility in expressions. In PL/I, a character string variable can be the operand of an arithmetic operator with an integer as the other operand. At run time, the string is scanned for a numeric value. If the value happens to contain a decimal point, the value is assumed to be of floating-point type, the other operand is coerced to floating point, and the resulting operation is floating-point. This coercion policy is very expensive, because both the type check and the conversion must be done at run time. It also eliminates the possibility of detecting programmer errors in expressions, because a binary operator can combine an operand of any type with an operand of virtually any other type.

```
byte  a, b, c;
. . .
a = b + c;
```

The values of `b` and `c` are coerced to **`int`** and an **`int`** addition is performed. Then, the sum is converted to **`byte`** and put in `a`. Given the large size of the memories of contemporary computers, there is little incentive to use **`byte`** and **`short`**, unless a large number of them must be stored.

7.4.2 Explicit Type Conversion

Most languages provide some capability for doing explicit conversions, both widening and narrowing. In some cases, warning messages are produced when an explicit narrowing conversion results in a significant change to the value of the object being converted.

In the C-based languages, explicit type conversions are called **casts**. To specify a cast, the desired type is placed in parentheses just before the expression to be converted, as in

```
(int)angle
```

One of the reasons for the parentheses around the type name in these conversions is that the first of these languages, C, has several two-word type names, such as **`long int`**.

In ML and F#, the casts have the syntax of function calls. For example, in F# we could have the following:

```
float(sum)
```

7.4.3 Errors in Expressions

A number of errors can occur during expression evaluation. If the language requires type checking, either static or dynamic, then operand type errors cannot occur. The errors that can occur because of coercions of operands in

expressions have already been discussed. The other kinds of errors are due to the limitations of computer arithmetic and the inherent limitations of arithmetic. The most common error occurs when the result of an operation cannot be represented in the memory cell where it must be stored. This is called **overflow** or **underflow**, depending on whether the result was too large or too small. One limitation of arithmetic is that division by zero is disallowed. Of course, the fact that it is not mathematically allowed does not prevent a program from attempting to do it.

Floating-point overflow, underflow, and division by zero are examples of run-time errors, which are sometimes called **exceptions**. Language facilities that allow programs to detect and deal with exceptions are discussed in Chapter 14.

7.5 Relational and Boolean Expressions

In addition to arithmetic expressions, programming languages support relational and Boolean expressions.

7.5.1 Relational Expressions

A **relational operator** is an operator that compares the values of its two operands. A relational expression has two operands and one relational operator. The value of a relational expression is Boolean, except when Boolean is not a type included in the language. The relational operators are often overloaded for a variety of types. The operation that determines the truth or falsehood of a relational expression depends on the operand types. It can be simple, as for integer operands, or complex, as for character string operands. Typically, the types of the operands that can be used for relational operators are numeric types, strings, and enumeration types.

history note

The Fortran I designers used English abbreviations for the relational operators because the symbols > and < were not on the card punches at the time of Fortran I's design (mid-1950s).

The syntax of the relational operators for equality and inequality differs among some programming languages. For example, for inequality, the C-based languages use `!=`, Fortran 95+ uses `.NE.` or <>, and ML and F# use <>.

JavaScript and PHP have two additional relational operators, `===` and `!==`. These are similar to their relatives, == and `!=`, but prevent their operands from being coerced. For example, the expression

```
"7" == 7
```

is true in JavaScript, because when a string and a number are the operands of a relational operator, the string is coerced to a number. However,

```
"7" === 7
```

is false, because no coercion is done on the operands of this operator.

Ruby uses == for the equality relational operator that uses coercions, and `eql?` for equality with no coercions. Ruby uses === only in the **`when`** clause of its **`case`** statement, as discussed in Chapter 8.

The relational operators always have lower precedence than the arithmetic operators, so that in expressions such as

```
a + 1 > 2 * b
```

the arithmetic expressions are evaluated first.

7.5.2 Boolean Expressions

Boolean expressions consist of Boolean variables, Boolean constants, relational expressions, and Boolean operators. The operators usually include those for the AND, OR, and NOT operations, and sometimes for exclusive OR and equivalence. Boolean operators usually take only Boolean operands (Boolean variables, Boolean literals, or relational expressions) and produce Boolean values.

In the mathematics of Boolean algebras, the OR and AND operators must have equal precedence. However, the C-based languages assign a higher precedence to AND than OR. Perhaps this resulted from the baseless correlation of multiplication with AND and of addition with OR, which would naturally assign higher precedence to AND.

Because arithmetic expressions can be the operands of relational expressions, and relational expressions can be the operands of Boolean expressions, the

three categories of operators must be placed in different precedence levels, relative to each other.

The precedence of the arithmetic, relational, and Boolean operators in the C-based languages is as follows:

Highest	postfix ++, --
	unary +, unary -, prefix ++, --, !
	*, /, %
	binary +, binary -
	<, >, <=, >=
	=, !=
	&&
Lowest	||

Versions of C prior to C99 are odd among the popular imperative languages in that they have no Boolean type and thus no Boolean values. Instead, numeric values are used to represent Boolean values. In place of Boolean operands, scalar variables (numeric or character) and constants are used, with zero considered false and all nonzero values considered true. The result of evaluating such an expression is an integer, with the value 0 if false and 1 if true. Arithmetic expressions can also be used for Boolean expressions in C99 and C++.

One odd result of C's design of relational expressions is that the following expression is legal:

```
a > b > c
```

The leftmost relational operator is evaluated first because the relational operators of C are left associative, producing either 0 or 1. Then, this result is

compared with the variable `c`. There is never a comparison between `b` and `c` in this expression.

Some languages, including Perl and Ruby, provide two sets of the binary logic operators, `&&` and **`and`** for AND and `||` and **`or`** for OR. One difference between `&&` and **`and`** (and `||` and **`or`**) is that the spelled versions have lower precedence. Also, **`and`** and **`or`** have equal precedence, but `&&` has higher precedence than `||`.

When the nonarithmetic operators of the C-based languages are included, there are more than 40 operators and at least 14 different levels of precedence. This is clear evidence of the richness of the collections of operators and the complexity of expressions possible in these languages.

Readability dictates that a language should include a Boolean type, as was stated in Chapter 6, rather than simply using numeric types in Boolean expressions. Some error detection is lost in the use of numeric types for Boolean operands, because any numeric expression, whether intended or not, is a legal operand to a Boolean operator. In the other imperative languages, any non-Boolean expression used as an operand of a Boolean operator is detected as an error.

7.6 Short-Circuit Evaluation

A **short-circuit evaluation** of an expression is one in which the result is determined without evaluating all of the operands and/or operators. For example, the value of the arithmetic expression

```
(13 * a) * (b / 13 - 1)
```

is independent of the value of `(b / 13 - 1)` if `a` is `0`, because `0 * x = 0` for any `x`. So, when `a` is `0`, there is no need to evaluate `(b / 13 - 1)` or perform the second multiplication. However, in arithmetic expressions, this shortcut is not easily detected during execution, so it is never taken.

The value of the Boolean expression

```
(a >= 0) && (b < 10)
```

is independent of the second relational expression if `a < 0`, because the expression (`FALSE && (b < 10)`) is `FALSE` for all values of `b`. So, when `a` is less than zero, there is no need to evaluate `b`, the constant `10`, the second relational expression, or the `&&` operation. Unlike the case of arithmetic expressions, this shortcut easily can be discovered during execution.

To illustrate a potential problem with non-short-circuit evaluation of Boolean expressions, suppose Java did not use short-circuit evaluation. A table lookup loop could be written using the **`while`** statement. One simple version of Java code for such a lookup, assuming that `list`, which has `listlen` elements, is the array to be searched and `key` is the searched-for value, is

```
index = 0;
while ((index < listlen) && (list[index] != key))
  index = index + 1;
```

If evaluation is not short-circuit, both relational expressions in the Boolean expression of the **`while`** statement are evaluated, regardless of the value of the first. Thus, if `key` is not in `list`, the program will terminate with a subscript out-of-range exception. The same iteration that has `index == listlen` will

reference `list[listlen]`, which causes the indexing error because list is declared to have `listlen-1` as an upper-bound subscript value.

If a language provides short-circuit evaluation of Boolean expressions and it is used, this is not a problem. In the preceding example, a short-circuit evaluation scheme would evaluate the first operand of the AND operator, but it would skip the second operand if the first operand is false.

A language that provides short-circuit evaluations of Boolean expressions and also has side effects in expressions allows subtle errors to occur. Suppose that short-circuit evaluation is used on an expression and part of the expression that contains a side effect is not evaluated; then the side effect will occur only in complete evaluations of the whole expression. If program correctness depends on the side effect, short-circuit evaluation can result in a serious error. For example, consider the Java expression

```
(a > b) || ((b++) / 3)
```

In this expression, `b` is changed (in the second arithmetic expression) only when `a <= b`. If the programmer assumed `b` would be changed every time this expression is evaluated during execution (and the program's correctness depends on it), the program will fail.

In the C-based languages, the usual AND and OR operators, `&&` and `||`, respectively, are short-circuit. However, these languages also have bitwise AND and OR operators, `&` and `|`, respectively, that can be used on Boolean-valued operands and are not short-circuit. Of course, the bitwise operators are only equivalent to the usual Boolean operators if all operands are restricted to being either `0` (for false) or `1` (for true).

All of the logical operators of Ruby, Perl, ML, F#, and Python are short--circuit evaluated.

7.7 Assignment Statements

As we have previously stated, the assignment statement is one of the central constructs in imperative languages. It provides the mechanism by which the user can dynamically change the bindings of values to variables. In the following section, the simplest form of assignment is discussed. Subsequent sections describe a variety of alternatives.

7.7.1 Simple Assignments

Nearly all programming languages currently being used use the equal sign for the assignment operator. All of these must use something different from an equal sign for the equality relational operator to avoid confusion with their assignment operator.

ALGOL 60 pioneered the use of `:=` as the assignment operator, which avoids the confusion of assignment with equality. Ada also uses this assignment operator.

The design choices of how assignments are used in a language have varied widely. In some languages, such as Fortran and Ada, an assignment can appear only as a stand-alone statement, and the destination is restricted to a single variable. There are, however, many alternatives.

7.7.2 Conditional Targets

Perl allows conditional targets on assignment statements. For example, consider

```
($flag ? $count1 : $count2) = 0;
```

which is equivalent to

```
if ($flag) {
  $count1 = 0;
} else  {
  $count2 = 0;
}
```

7.7.3 Compound Assignment Operators

A **compound assignment operator** is a shorthand method of specifying a commonly needed form of assignment. The form of assignment that can be abbreviated with this technique has the destination variable also appearing as the first operand in the expression on the right side, as in

```
a = a + b
```

Compound assignment operators were introduced by ALGOL 68, were later adopted in a slightly different form by C, and are part of the other C-based languages, as well as Perl, JavaScript, Python, and Ruby. The syntax of these assignment operators is the catenation of the desired binary operator to the = operator. For example,

```
sum += value;
```

is equivalent to

```
sum = sum + value;
```

The languages that support compound assignment operators have versions for most of their binary operators.

7.7.4 Unary Assignment Operators

The C-based languages, Perl, and JavaScript include two special unary

arithmetic operators that are actually abbreviated assignments. They combine increment and decrement operations with assignment. The operators ++ for increment and -- for decrement can be used either in expressions or to form stand-alone single-operator assignment statements. They can appear either as prefix operators, meaning that they precede the operands, or as postfix operators, meaning that they follow the operands. In the assignment statement

```
sum = ++ count;
```

the value of `count` is incremented by `1` and then assigned to `sum`. This operation could also be stated as

```
count = count + 1;
sum = count;
```

If the same operator is used as a postfix operator, as in

```
sum = count ++;
```

the assignment of the value of `count` to `sum` occurs first; then `count` is incremented. The effect is the same as that of the two statements

```
sum = count;
count = count + 1;
```

An example of the use of the unary increment operator to form a complete assignment statement is

```
count ++;
```

which simply increments `count`. It does not look like an assignment, but it certainly is one. It is equivalent to the statement

```
count = count + 1;
```

When two unary operators apply to the same operand, the association is right to left. For example, in

```
- count ++
```

`count` is first incremented and then negated. So, it is equivalent to

```
- (count ++)
```

rather than

```
(- count) ++
```

7.7.5 Assignment as an Expression

In the C-based languages, Perl, and JavaScript, the assignment statement produces a result, which is the same as the value assigned to the target. It can therefore be used as an expression and as an operand in other expressions. This design treats the assignment operator much like any other binary operator, except that it has the side effect of changing its left operand. For example, in C, it is common to write statements such as

```
while ((ch = getchar()) != EOF) { ... }
```

In this statement, the next character from the standard input file, usually the keyboard, is gotten with `getchar` and assigned to the variable `ch`. The result, or value assigned, is then compared with the constant `EOF`. If `ch` is not equal to `EOF`, the compound statement `{...}` is executed. Note that the assignment must be parenthesized—in the languages that support assignment as an expression, the precedence of the assignment operator is lower than that of the relational operators. Without the parentheses, the new character would be compared with `EOF` first. Then, the result of that comparison, either `0` or `1`, would be assigned to `ch`.

The disadvantage of allowing assignment statements to be operands in expressions is that it provides yet another kind of expression side effect. This type of side effect can lead to expressions that are difficult to read and understand. An expression with any kind of side effect has this disadvantage. Such an expression cannot be read as an expression, which in mathematics is a denotation of a value, but only as a list of instructions with an odd order of execution. For example, the expression

```
a = b + (c = d / b) - 1
```

denotes the instructions

```
Assign d / b to c
Assign b + c to temp
Assign temp - 1 to a
```

Note that the treatment of the assignment operator as any other binary operator allows the effect of multiple-target assignments, such as

```
sum = count = 0;
```

in which count is first assigned the zero, and then count's value is assigned to sum. This form of multiple-target assignments is also legal in Python.

There is a loss of error detection in the C design of the assignment operation that frequently leads to program errors. In particular, if we type

```
if  (x = y) ...
```

instead of

```
if  (x == y) ...
```

which is an easily made mistake, it is not detectable as an error by the compiler. Rather than testing a relational expression, the value that is assigned to `x` is tested (in this case, it is the value of `y` that reaches this statement). This is actually a result of two design decisions: allowing assignment to behave like an ordinary binary operator and using two very similar operators, = and ==, to have completely different meanings. This is another example of the safety deficiencies of C and C++ programs. Note that Java and C# allow only **`boolean`** expressions in their **`if`** statements, disallowing this problem.

7.7.6 Multiple Assignments

Several recent programming languages, including Perl and Ruby provide

multiple-target, multiple-source assignment statements. For example, in Perl one can write

```
($first, $second, $third) = (20, 40, 60);
```

The semantics is that `20` is assigned to `$first`, `40` is assigned to `$second`, and `60` is assigned to `$third`. If the values of two variables must be interchanged, this can be done with a single assignment, as with

```
($first, $second) = ($second, $first);
```

This correctly interchanges the values of `$first` and `$second`, without the use of a temporary variable (at least one created and managed by the programmer).

history note

The PDP-11 computer, on which C was first implemented, has autoincrement and autodecrement addressing modes, which are hardware versions of the increment and decrement operators of C when they are used as array indices. One might guess from this that the design of these C operators was based on the design of the PDP-11 architecture. That guess would be wrong, however, because the C operators were inherited from the B language, which was designed before the first PDP-11.

The syntax of the simplest form of Ruby’s multiple assignment is similar to that of Perl, except the left and right sides are not parenthesized. Also, Ruby includes a few more elaborate versions of multiple assignments, which are not discussed here.

7.7.7 Assignment in Functional Programming Languages

All of the identifiers used in pure functional languages and some of them

used in other functional languages are just names of values. As such, their values never change. For example, in ML, names are bound to values with the **`val`** declaration, whose form is exemplified in the following:

```
val cost = quantity * price;
```

If `cost` appears on the left side of a subsequent **`val`** declaration, that declaration creates a new version of the name `cost`, which has no relationship with the previous version, which is then hidden.

F# has a somewhat similar declaration that uses the **`let`** reserved word. The difference between F#'s **`let`** and ML's **`val`** is that **`let`** creates a new scope, whereas **`val`** does not. In fact, **`val`** declarations are often nested in **`let`** constructs in ML. **`let`** and **`val`** are further discussed in Chapter 15.

7.8 Mixed-Mode Assignment

Mixed-mode expressions were discussed in Section 7.4.1. Frequently, assignment statements also are mixed mode. The design question is: Does the type of the expression have to be the same as the type of the variable being assigned, or can coercion be used in some cases of type mismatch?

C, C++, and Perl use coercion rules for mixed-mode assignment that are similar to those they use for mixed-mode expressions; that is, many of the possible type mixes are legal, with coercion freely applied.7

7. Note that in Python and Ruby, types are associated with objects, not variables, so there is no such thing as mixed-mode assignment in those languages.

In a clear departure from C++, Java and C# allow mixed-mode assignment only if the required coercion is widening.8 So, an `int` value can be assigned to a `float` variable, but not vice versa. Disallowing half of the possible mixed-mode assignments is a simple but effective way to increase the reliability of Java and C#, relative to C and C++.

8. Not quite true: If an integer literal, which the compiler by default assigns the type `int`, is assigned to a **char**, `byte`, or `short` variable and the literal is in the range of the type of the variable, the `int` value is coerced to the type of the variable in a narrowing conversion. This narrowing conversion cannot result in an error.

Of course, in functional languages, where assignments are just used to name values, there is no such thing as a mixed-mode assignment.

SUMMARY

Expressions consist of constants, variables, parentheses, function calls, and operators. Assignment statements include target variables, assignment operators, and expressions.

The semantics of an expression is determined in large part by the order of evaluation of operators. The associativity and precedence rules for operators in the expressions of a language determine the order of operator evaluation in those expressions. Operand evaluation order is important if functional side effects are possible. Type conversions can be widening or narrowing. Some narrowing conversions produce erroneous values. Implicit type conversions, or coercions, in expressions are common, although they eliminate the error--detection benefit of type checking, thus lowering reliability.

Assignment statements have appeared in a wide variety of forms, including conditional targets, assigning operators, and list assignments.

REVIEW QUESTIONS

1. Define *operator precedence* and *operator associativity*.
2. What is a ternary operator?
3. What is a prefix operator?
4. What operator usually has right associativity?
5. What is a nonassociative operator?
6. What associativity rules are used by APL?
7. What is the difference between the way operators are implemented in C++ and Ruby?
8. Define *functional side effect*.
9. What is a coercion?
10. What is a conditional expression?
11. What is an overloaded operator?
12. Define *narrowing* and *widening conversions*.
13. In JavaScript, what is the difference between == and ===?
14. What is a mixed-mode expression?
15. What is referential transparency?
16. What are the advantages of referential transparency?
17. How does operand evaluation order interact with functional side effects?

18. What is short-circuit evaluation?

19. Name a language that always does short-circuit evaluation of Boolean expressions. Name one that never does it.

20. How does C support relational and Boolean expressions?

21. What is the purpose of a compound assignment operator?

22. What is the associativity of C's unary arithmetic operators?

23. What is one possible disadvantage of treating the assignment operator as if it were an arithmetic operator?

24. What two languages include multiple assignments?

25. What mixed-mode assignments are allowed in Java?

26. What mixed-mode assignments are allowed in ML?

27. What is a cast?

PROBLEM SET

1. When might you want the compiler to ignore type differences in an expression?

2. State your own arguments for and against allowing mixed-mode arithmetic expressions.

3. Do you think the elimination of overloaded operators in your favorite language would be beneficial? Why or why not?

4. Would it be a good idea to eliminate all operator precedence rules and require parentheses to show the desired precedence in expressions? Why or why not?

5. Should C's assigning operations (for example, `+=`) be included in other languages (that do not already have them)? Why or why not?

6. Should C's single-operand assignment forms (for example, `++count`) be included in other languages (that do not already have them)? Why or why not?

7. Describe a situation in which the add operator in a programming language would not be commutative.

8. Describe a situation in which the add operator in a programming language would not be associative.

9. Assume the following rules of associativity and precedence for expressions:

Precedence	*Highest*	*, /, **not**
		+, −, &, **mod**
		- (unary)
		=, / =, <, <=, >=, >
		and
	Lowest	**or**, **xor**
Associativity	*Left to right*	

Show the order of evaluation of the following expressions by parenthesizing all subexpressions and placing a superscript on the right parenthesis to indicate order. For example, for the expression

```
a + b * c + d
```

the order of evaluation would be represented as

```
((a + (b * c)1)2  + d)3
```

1. `a * b - 1 + c`
2. `a * (b - 1) / c` **mod** `d`
3. `(a - b) / c & (d * e / a - 3)`
4. `-a` **or** `c = d` **and** `e`
5. `a > b` **xor** `c` **or** `d <= 17`
6. `-a + b`

10. Show the order of evaluation of the expressions of Problem 9, assuming that there are no precedence rules and all operators associate right to left.

11. Write a BNF description of the precedence and associativity rules defined for the expressions in Problem 9. Assume the only operands are the names `a,b,c,d,` and `e`.

12. Using the grammar of Problem 11, draw parse trees for the expressions of Problem 9.

13. Let the function `fun` be defined as

```
int fun(int* k) {
 *k += 4;
 return 3 * (*k) - 1;
 }
```

Suppose `fun` is used in a program as follows:

```
void  main() {
  int i = 10, j = 10, sum1, sum2;
  sum1 = (i / 2) + fun(&i);
  sum2 = fun(&j) + (j / 2);
}
```

What are the values of `sum1` and `sum2`

1. operands in the expressions are evaluated left to right?

2. operands in the expressions are evaluated right to left?

14. What is your primary argument against (or for) the operator precedence rules of APL?

15. Explain why it is difficult to eliminate functional side effects in C.

16. For some language of your choice, make up a list of operator symbols that could be used to eliminate all operator overloading.

17. Determine whether the narrowing explicit type conversions in two languages you know provide error messages when a converted value loses its usefulness.

18. Should an optimizing compiler for C or C++ be allowed to change the order of subexpressions in a Boolean expression? Why or why not?

19. Consider the following C program:

```
int fun(int *i) {
  *i += 5;
  return  4;
}
void main() {
  int x = 3;
  x = x + fun(&x);
  }
```

What is the value of x after the assignment statement in main, assuming

1. operands are evaluated left to right.

2. operands are evaluated right to left.

20. Why does Java specify that operands in expressions are all evaluated in left-to-right order?

21. Explain how the coercion rules of a language affect its error detection.

PROGRAMMING EXERCISES

1. Run the code given in Problem 13 (in the Problem Set) on some system that supports C to determine the values of `sum1` and `sum2`. Explain the results.

2. Rewrite the program of Programming Exercise 1 in C++, Java, and C#, run them, and compare the results.

3. Write a test program in your favorite language that determines and outputs the precedence and associativity of its arithmetic and Boolean operators.

4. Write a Java program that exposes Java's rule for operand evaluation order when one of the operands is a method call.

5. Repeat Programming Exercise 4 with C++.

6. Repeat Programming Exercise 4 with C#.

7. Write a program in either C++, Java, or C# that illustrates the order of evaluation of expressions used as actual parameters to a method.

8. Write a C program that has the following statements:

```
int  a, b;
a = 10;
b = a + fun();
printf("With the function call on the right, ");
printf(" b is: %d\n", b);
a = 10;
b = fun() + a;
printf("With the function call on the left, ");
printf(" b is: %d\n", b);
```

   and define `fun` to add 10 to `a`. Explain the results.

9. Write a program in either Java, C++, or C# that performs a large number

of floating-point operations and an equal number of integer operations and compare the time required.

8 Statement-Level Control Structures

The flow of control, or execution sequence, in a program can be examined at several levels. In Chapter 7, the flow of control within expressions, which is - governed by operator associativity and precedence rules, was discussed. At the highest level is the flow of control among program units, which is discussed in Chapters 9 and 13. Between these two extremes is the important issue of the flow of control among statements, which is the subject of this chapter.

We begin by giving an overview of the evolution of control statements. This topic is followed by a thorough examination of selection statements, both those for two-way and those for multiple selection. Next, we discuss the variety of looping statements that have been developed and used in programming languages. Next, we take a brief look at the problems associated with unconditional branch statements. Finally, we describe the guarded command control statements.

8.1 Introduction

Computations in imperative-language programs are accomplished by evaluating expressions and assigning the resulting values to variables. However, there are few useful programs that consist entirely of assignment statements. At least two additional linguistic mechanisms are necessary to make the computations in programs flexible and powerful: some means of selecting among alternative control flow paths (of statement execution) and some means of causing the repeated execution of statements or sequences of statements. Statements that provide these kinds of capabilities are called **control statements**.

Computations in functional programming languages are accomplished by evaluating expressions and applying functions to given parameters. Furthermore, the flow of execution among the expressions and functions is controlled by other expressions and functions, although some of them are similar to the control statements in the imperative languages.

The control statements of the first successful programming language, Fortran, were, in effect, designed by the architects of the IBM 704. All were directly related to machine language instructions, so their capabilities were more the result of instruction design rather than language design. At the time, little was known about the difficulty of programming, and, as a result, the control statements of Fortran in the mid-1950s were thought to be entirely adequate. By today's standards, however, they are considered seriously lacking.

A great deal of research and discussion was devoted to control statements in the 10 years between the mid-1960s and the mid-1970s. One of the primary conclusions of these efforts was that, although a single control statement (a selectable goto) is minimally sufficient, a language that is designed *not* to include a goto needs only a small number of different control statements. In fact, it was proven that all algorithms that can be expressed by flowcharts can be coded in a programming language with only two control statements: one for choosing between two control flow paths and one for logically controlled iterations (Böhm and Jacopini, 1966). An important result of this is that the

unconditional branch statement is superfluous—potentially useful but nonessential. This fact, combined with the practical problems of using unconditional branches, or gotos, led to a great deal of debate about the goto, as will be discussed in Section 8.4.

Programmers care less about the results of theoretical research on control statements than they do about writability and readability. All languages that have been widely used include more control statements than the two that are minimally required, because writability is enhanced by a larger number and wider variety of control statements. For example, rather than requiring the use of a logically controlled loop statement for all loops, it is easier to write programs when a counter-controlled loop statement can be used to build loops that are naturally controlled by a counter. The primary factor that restricts the number of control statements in a language is readability, because the presence of a large number of statement forms demands that program readers learn a larger language. Recall that few people learn all of the statements of a relatively large language; instead, they learn the subset they choose to use, which is often a different subset from that used by the programmer who wrote the program they are trying to read. On the other hand, too few control statements can require the use of lower-level statements, such as the goto, which also makes programs less readable.

The question as to the best collection of control statements to provide the required capabilities and the desired writability has been widely debated. It is essentially a question of how much a language should be expanded to increase its writability at the expense of its simplicity, size, and readability.

A **control structure** is a control statement and the collection of statements whose execution it controls.

There is only one design issue that is relevant to all of the selection and iteration control statements: Should the control structure have multiple entries? All selection and iteration constructs control the execution of code segments, and the question is whether the execution of those code segments always begins with the first statement in the segment. It is now generally believed that multiple entries add little to the flexibility of a control statement, relative to the decrease in readability caused by the increased complexity. Note that multiple entries are possible only in languages that

include gotos and statement labels.

At this point, the reader might wonder why multiple exits from control structures are not considered a design issue. The reason is that all programming languages allow some form of multiple exits from control structures, the rationale being as follows: If all exits from a control structure are restricted to transferring control to the first statement following the structure, where control would flow if the control structure had no explicit exit, there is no harm to readability and also no danger. However, if an exit can have an unrestricted target and therefore can result in a transfer of control to anywhere in the program unit that contains the control structure, the harm to readability is the same as for a goto statement anywhere else in a program. Languages that have a goto statement allow it to appear anywhere, including in a control structure. Therefore, the issue is the inclusion of a goto, not whether multiple exits from control expressions are allowed.

8.2 Selection Statements

A **selection statement** provides the means of choosing between two or more execution paths in a program. Such statements are fundamental and essential parts of all programming languages, as was proven by Böhm and Jacopini.

Selection statements fall into two general categories: two-way and n-way, or multiple selection. Two-way selection statements are discussed in Section 8.2.1; multiple-selection statements are covered in Section 8.2.2.

8.2.1 Two-Way Selection Statements

Although the two-way selection statements of contemporary imperative languages are quite similar, there are some variations in their designs. The general form of a two-way selector is as follows:

`if` control_expression

then clause

else clause

8.2.1.1 Design Issues

The design issues for two-way selectors can be summarized as follows:

- What is the form and type of the expression that controls the selection?
- How are the then and else clauses specified?
- How should the meaning of nested selectors be specified?

8.2.1.2 The Control Expression

Control expressions are specified in parentheses if the **`then`** reserved word (or some other syntactic marker) is not used to introduce the then clause. In those cases where the **`then`** reserved word (or alternative marker) is used, there is less need for the parentheses, so they are often omitted, as in Ruby.

In C89, which did not have a Boolean data type, arithmetic expressions were used as control expressions. This can also be done in Python, C99, and C++. However, in those languages either arithmetic or Boolean expressions can be used. In other contemporary languages, only Boolean expressions can be used for control expressions.

8.2.1.3 Clause Form

In many languages, the then and else clauses appear as either single statements or compound statements. One variation of this is Perl, in which all then and else clauses must be compound statements, even if they have only one statement. Many languages use braces to form compound statements, which serve as the bodies of then and else clauses. In Python and Ruby, the then and else clauses are statement sequences, rather than compound statements. The complete selection statement is terminated in these languages with a reserved word.

Python uses indentation to specify compound statements. For example,

```
if x > y :
  x = y
  print "case 1"
```

All statements equally indented are included in the compound statement.1 Notice that rather than **`then`**, a colon is used to introduce the then clause in Python.

1. The statement following the compound statement must have the same indentation as the `if`.

The variations in clause form have implications for the specification of the meaning of nested selectors, as discussed in the next subsection.

8.2.1.4 Nesting Selectors

Recall that in Chapter 3, we discussed the problem of syntactic ambiguity of a straightforward grammar for a two-way selector statement. That ambiguous grammar was as follows:

<if_stmt> → **if** <logic_expr> **then** <stmt>

| **if** <logic_expr> **then** <stmt> **else** <stmt>

The issue is that when a selection statement is nested in the then clause of a selection statement, it is not clear with which if an else clause should be associated. This problem is reflected in the semantics of selection statements. Consider the following Java-like code:

```
if (sum == 0)
  if (count == 0)
    result = 0;
else
    result = 1;
```

This statement can be interpreted in two different ways, depending on whether the else clause is matched with the first then clause or the second. Notice that the indentation seems to indicate that the else clause belongs with the first then clause. However, with the exceptions of Python and F#, indentation has no effect on semantics in contemporary languages and is therefore ignored by their compilers.

The crux of the problem in this example is that the else clause follows two then clauses with no intervening else clause, and there is no syntactic indicator to specify a matching of the else clause to one of the then clauses. In Java, as in many other imperative languages, the static semantics of the language specify that the else clause is always paired with the nearest previous unpaired then clause. A static semantics rule, rather than a syntactic entity, is used to provide the disambiguation. So, in the example, the else

clause would be paired with the second then clause. The disadvantage of using a rule rather than some syntactic entity is that although the programmer may have meant the else clause to be the alternative to the first then clause and the compiler found the structure syntactically correct, its semantics is the opposite. To force the alternative semantics in Java, the inner **`if`** is put in a compound, as in

```
if (sum == 0) {
  if (count == 0)
    result = 0;
}
else
    result = 1;
```

C, C++, and C# have the same problem as Java with selection statement nesting. Because Perl requires that all then and else clauses be compound, it does not. In Perl, the previous code would be written as follows:

```
if (sum == 0) {
  if (count == 0) {
    result = 0;
  }
} else {
    result = 1;
}
```

If the alternative semantics were needed, it would be

```
if (sum == 0) {
  if (count == 0) {
    result = 0;
  }
  else {
    result = 1;
  }
}
```

Another way to avoid the issue of nested selection statements is to use an alternative means of forming compound statements. Consider the syntactic structure of the Java **`if`** statement. The then clause follows the control expression and the else clause is introduced by the reserved word **`else`**. When the then clause is a single statement and the else clause is present, although

there is no need to mark the end, the **else** reserved word in fact marks the end of the then clause. When the then clause is a compound, it is terminated by a right brace. However, if the last clause in an **if**, whether then or else, is not a compound, there is no syntactic entity to mark the end of the whole selection statement. The use of a special word for this purpose resolves the question of the semantics of nested selectors and also adds to the readability of the statement. This is the design of the selection statement in Ruby. For example, consider the following Ruby statement:

```
if a > b then sum = sum + a
  acount = acount + 1
else sum = sum + b
  bcount = bcount + 1
end
```

The design of this statement is more regular than that of the selection statements of the C-based languages, because the form is the same regardless of the number of statements in the then and else clauses. (This is also true for Perl.) Recall that in Ruby, the then and else clauses consist of statement sequences rather than compound statements. The first interpretation of the selector example at the beginning of this section, in which the else clause is matched to the nested **if**, can be written in Ruby as follows:

```
if sum == 0  then
  if count == 0 then
    result = 0
  else
    result = 1
  end
end
```

Because the **end** reserved word closes the nested **if**, it is clear that the else clause is matched to the inner then clause.

The second interpretation of the selection statement at the beginning of this section, in which the else clause is matched to the outer **if**, can be written in Ruby as follows:

```
if sum == 0 then
  if count == 0 then
    result = 0
```

```
  end
else
    result = 1
 end
```

The following statement, written in Python, is semantically equivalent to the last Ruby statement above:

```
if sum == 0 :
  if count == 0 :
    result = 0
else:
  result = 1
```

If the line **`else:`** were indented to begin in the same column as the nested **`if`**, the else clause would be matched with the inner **`if`**.

ML does not have a problem with nested selectors because it does not allow else-less **`if`** statements.

8.2.1.5 Selector Expressions

In the functional languages ML, F#, and LISP, the selector is not a statement; it is an expression that results in a value. Therefore, it can appear anywhere any other expression can appear. Consider the following example selector written in F#:

```
let y =
    if x > 0 then  x
    else 2 * x;;
```

This creates the name `y` and sets it to either `x` or `2 * x`, depending on whether `x` is greater than zero.

In F#, the type of the value returned by the then clause of an **`if`** construct must be the same as that of the value returned by its else clause. If there is no else clause, the then clause cannot return a value of a normal type. In this case, it can only return a **`unit`** type, which is a special type that means no value. A **`unit`** type is represented in code as `()`.

8.2.2 Multiple-Selection Statements

The **multiple-selection** statement allows the selection of one of any number of statements or statement groups. It is, therefore, a generalization of a selector. In fact, two-way selectors can be built with a multiple selector.

The need to choose from among more than two control paths in programs is common. Although a multiple selector can be built from two-way selectors and gotos, the resulting structures are cumbersome, unreliable, and difficult to write and read. Therefore, the need for a special structure is clear.

8.2.2.1 Design Issues

Some of the design issues for multiple selectors are similar to some of those for two-way selectors. For example, one issue is the question of the type of expression on which the selector is based. In this case, the range of possibilities is larger, in part because the number of possible selections is larger. A two-way selector needs an expression with only two possible values. Another issue is whether single statements, compound statements, or statement sequences may be selected. Next, there is the question of whether only a single selectable segment can be executed when the statement is executed. This is not an issue for two-way selectors, because they always allow only one of the clauses to be on a control path during one execution. As we shall see, the resolution of this issue for multiple selectors is a trade-off between reliability and flexibility. Another issue is the form of the case value specifications. Finally, there is the issue of what should result from the selector expression evaluating to a value that does not select one of the segments. (Such a value would be unrepresented among the selectable segments.) The choice here is between simply disallowing the situation from arising and having the statement do nothing at all when it does arise.

The following is a summary of these design issues:

- What is the form and type of the expression that controls the selection?
- How are the selectable segments specified?
- Is execution flow through the structure restricted to include just a single selectable segment?
- How are the case values specified?
- How should unrepresented selector expression values be handled, if at all?

8.2.2.2 Examples of Multiple Selectors

The C multiple-selector statement, **`switch`**, which is also part of C++, Java, and JavaScript, is a relatively primitive design. Its general form is

```
switch (expression) {
  case constant_ expression1: statement1;
  . . .
  case constantn: statement_n;
  [default: statementn+1]
}
```

where the control expression and the constant expressions are some discrete type. This includes integer types, as well as characters and enumeration types. The selectable statements can be statement sequences, compound statements, or blocks. The optional **`default`** segment is for unrepresented values of the control expression. If the value of the control expression is not represented and no default segment is present, then the statement does

nothing.

The **switch** statement does not provide implicit branches at the end of its code segments. This allows control to flow through more than one selectable code segment on a single execution. Consider the following example:

```
switch  (index) {
   case  1:
   case  3: odd += 1;
          sumodd += index;
   case  2:
   case  4: even += 1;
          sumeven += index;
   default: printf("Error in switch, index = %d\n", index);
}
```

This code prints the error message on every execution. Likewise, the code for the 2 and 4 constants is executed every time the code at the 1 or 3 constants is executed. To separate these segments logically, an explicit branch must be included. The **break** statement, which is actually a restricted goto, is normally used for exiting **switch** statements. **break** transfers control to the first statement after the compound statement in which it appears.

The following **switch** statement uses **break** to restrict each execution to a single selectable segment:

```
switch (index) {
  case 1:
  case 3: odd += 1;
          sumodd += index;
          break;
  case 2:
  case 4: even += 1;
          sumeven += index;
          break;
  default: printf("Error in switch, index = %d\n", index);
}
```

Occasionally, it is convenient to allow control to flow from one selectable code segment to another. For example, in the example above, the segments for the case values 1 and 2 are empty, allowing control to flow to the segments for 3 and 4, respectively. This is the reason why there are no

implicit branches in the **switch** statement. The reliability problem with this design arises when the mistaken absence of a **break** statement in a segment allows control to flow to the next segment incorrectly. The designers of C's **switch** traded a decrease in reliability for an increase in flexibility. Studies have shown, however, that the ability to have control flow from one selectable segment to another is rarely used. C's **switch** is modeled on the multiple-selection statement in ALGOL 68, which also does not have implicit branches from selectable segments.

The C switch statement has virtually no restrictions on the placement of the case expressions, which are treated as if they were normal statement labels. This laxness can result in highly complex structure within the switch body. The following example is taken from Harbison and Steele (2002).

```
switch (x)
  default:
  if (prime(x))
    case 2: case 3: case 5: case 7:
      process_prime(x);
  else
     case 4: case 6: case 8: case 9: case 10:
       process_composite(x);
```

This code may appear to have diabolically complex form, but it was designed for a real problem and works correctly and efficiently to solve that problem.2

2. The problem is to call `process_prime` when `x` is prime and `process_composite` when `x` is not prime. The design of the switch body resulted from an attempt to optimize based on the knowledge that `x` was most often in the range of `1` to `10`.

The Java switch prevents this sort of complexity by disallowing case expressions from appearing anywhere except the top level of the body of the switch.

The C# switch statement differs from that of its C-based predecessors in two ways. First, C# has a static semantics rule that disallows the implicit execution of more than one segment. The rule is that every selectable segment must end with an explicit unconditional branch statement: either a **break**, which transfers control out of the **switch** statement, or a goto, which

can transfer control to one of the selectable segments (or virtually anywhere else). For example,

```
switch (value) {
   case -1:
      Negatives++;
      break;
   case 0:
      Zeros++;
      goto case 1;
   case 1:
      Positives++;
   default:
      Console.WriteLine("Error in switch \n");
}
```

Note that `Console.WriteLine` is the method for displaying strings in C#.

The other way C#'s **`switch`** differs from that of its predecessors is that the control expression and the case statements can be strings in C#.

PHP's **`switch`** uses the syntax of C's **`switch`** but allows more type flexibility. The case values can be any of the PHP scalar types—string, integer, or double precision. As with C, if there is no **`break`** at the end of the selected segment, execution continues into the next segment.

Ruby has two forms of multiple-selection constructs, both of which are called *case expressions* and both of which yield the value of the last expression evaluated. The only version of Ruby's case expressions that is described here is semantically similar to a list of nested if statements:

```
case
when Boolean_expression then expression
. . .
when Boolean_expression then expression
[else expression]
end
```

The semantics of this case expression is that the Boolean expressions are evaluated one at a time, top to bottom. The value of the case expression is the value of the first then expression whose Boolean expression is true. The else represents true in this statement, and the else clause is optional. For

example,[3]

3. This example is from Thomas et al. (2013).

```
leap = case
       when year % 400 == 0 then true
       when year % 100 == 0 then false
       else year % 4 == 0
       end
```

This case expression evaluates to true if `year` is a leap year.

The other Ruby case expression form is similar to the switch of Java. Perl and Python do not have multiple-selection statements.

8.2.2.3 Implementing Multiple Selection Structures

A multiple selection statement is essentially an n-way branch to segments of code, where n is the number of selectable segments. Implementing such a statement must be done with multiple conditional branch instructions. Consider again the general form of the C switch statement, with breaks:

- **switch** (expression) {
- **case** constant_expression1 : statement1;
- **break**;
- . . .
- **case** constantn : statementn;
- **break**;
- [**default** : statementn+1]

- }

One simple translation of this statement follows:

- Code to evaluate expression into t
- **goto** branches
- label1 : code for statement1
- **goto** out
- . . .
- labeln : code for statementn
- **goto** out
- **default**: code for statementn+1
- **goto** out
- branches: **if** t = constant_expression1 **goto** label1
- . . .
- **if** t = constant_expressionn **goto** labeln
- **goto** default
- out:

The code for the selectable segments precedes the branches so that the targets of the branches are all known when the branches are generated. An alternative to these coded conditional branches is to put the case values and labels in a table and use a linear search with a loop to find the correct label. This requires less space than the coded conditionals.

The use of conditional branches or a linear search on a table of cases and

labels is a simple but inefficient approach that is acceptable when the number of cases is small, say less than 10. It takes an average of about half as many tests as there are cases to find the right one. For the default case to be chosen, all other cases must be tested. In statements with 10 or more cases, the low efficiency of this form is not justified by its simplicity.

When the number of cases is 10 or greater, the compiler can build a hash table of the segment labels, which would result in approximately equal (and short) times to choose any of the selectable segments. If the language allows ranges of values for case expressions, as in Ruby, the hash method is not suitable. For these situations, a binary search table of case values and segment addresses is better.

If the range of the case values is relatively small and more than half of the whole range of values is represented, an array whose indices are the case values and whose values are the segment labels can be built. Array elements whose indices are not among the represented case values are filled with the default segment's label. Then finding the correct segment label is found by array indexing, which is very fast.

Of course, choosing among these approaches is an additional burden on the compiler. In many compilers, only two different methods are used. As in other situations, determining and using the most efficient method costs more compiler time.

8.2.2.4 Multiple Selection Using `if`

In many situations, a **`switch`** or **`case`** statement is inadequate for multiple selection (Ruby's **`case`** is an exception). For example, when selections must be made on the basis of a Boolean expression rather than some ordinal type, nested two-way selectors can be used to simulate a multiple selector. To alleviate the poor readability of deeply nested two-way selectors, some languages, such as Perl and Python, have been extended specifically for this use. The extension allows some of the special words to be left out. In particular, else-if sequences are replaced with a single special word, and the closing special word on the nested **`if`** is dropped. The nested selector is then

called an **else-if clause**. Consider the following Python selector statement (note that else-if is spelled **`elif`** in Python):

```
if  count < 10 :
  bag1 = True
elif count < 100 :
  bag2 = True
elif count < 1000 :
  bag3 = True
```

which is equivalent to the following:

```
if count < 10 :
  bag1 = True
else :
  if count < 100 :
    bag2 = True
  else :
    if count < 1000 :
      bag3 = True
    else :
      bag4 = True
```

The else-if version (the first) is the more readable of the two. Notice that this example is not easily simulated with a **`switch`** statement, because each selectable statement is chosen on the basis of a Boolean expression. Therefore, the else-if statement is not a redundant form of **`switch`**. In fact, none of the multiple selectors in contemporary languages are as general as the if-then-else-if statement. An operational semantics description of a general selector statement with else-if clauses, in which the E's are logic expressions and the S's are statements, is given here:

- **if** E1 **goto** 1
- **if** E2 **goto** 2
- . . .
- 1: S1
- **goto** out

- 2: S2
- **goto** out
- . . .
- out: . . .

From this description, we can see the difference between multiple selection structures and else-if statements: In a multiple selection statement, all the E's would be restricted to comparisons between the value of a single expression and some other values.

Languages that do not include the else-if statement can use the same control structure, with only slightly more typing.

The Python example if-then-else-if statement above can be written as the Ruby `case` statement:

```
case
when count < 10 then bag1 = true
when count < 100 then bag2 = true
when count < 1000 then bag3 = true
end
```

Else-if statements are based on the common mathematics statement, the conditional expression.

The Scheme multiple selector, which is based on mathematical conditional expressions, is a special form function named `COND`. `COND` is a slightly generalized version of the mathematical conditional expression; it allows more than one predicate to be true at the same time. Because different mathematical conditional expressions have different numbers of parameters, `COND` does not require a fixed number of actual parameters. Each parameter to `COND` is a pair of expressions in which the first is a predicate (it evaluates to either `#T` or `#F`).

The general form of `COND` is

- `(COND`

- `(` predicate1 expression1`)`
- `(` predicate2 expression2`)`
- . . .
- `(` predicaten expressionn`)`
- `[(ELSE` expressionn+1`)]`
- `)`

where the `ELSE` clause is optional.

The semantics of `COND` is as follows: The predicates of the parameters are evaluated one at a time, in order from the first, until one evaluates to `#T`. The expression that follows the first predicate that is found to be `#T` is then evaluated and its value is returned as the value of `COND`. If none of the predicates is true and there is an `ELSE`, its expression is evaluated and the value is returned. If none of the predicates is true and there is no `ELSE`, the value of `COND` is unspecified. Therefore, all `COND`s should include an `ELSE`.

Consider the following example call to `COND`:

```
(COND
  ((> x y) "x is greater than y")
  ((< x y) "y is greater than x")
  (ELSE "x and y are equal")
)
```

Note that string literals evaluate to themselves, so that when this call to `COND` is evaluated, it produces a string result.

8.3 Iterative Statements

An **iterative statement** is one that causes a statement or collection of statements to be executed zero, one, or more times. An iterative statement is often called a **loop**. Every programming language from Plankalkül on has included some method of repeating the execution of segments of code. Iteration is the very essence of the power of the computer. If some means of repetitive execution of a statement or collection of statements were not possible, programmers would be required to state every action in sequence; useful programs would be huge and inflexible and take unacceptably large amounts of time to write and mammoth amounts of memory to store.

The first iterative statements in programming languages were directly related to arrays. This resulted from the fact that in the earliest years of computers, computing was largely numerical in nature, frequently using loops to process data in arrays.

Several categories of iteration control statements have been developed. The primary categories are defined by how designers answered two basic design questions:

- How is the iteration controlled?
- Where should the control mechanism appear in the loop statement?

The primary possibilities for iteration control are logical, counting, or a combination of the two. The main choices for the location of the control mechanism are the top of the loop or the bottom of the loop. Top and bottom here are logical, rather than physical, denotations. The issue is not the physical placement of the control mechanism; rather, it is whether the mechanism is executed and affects control before or after execution of the statement's body. A third option, which allows the user to decide where to put the control, at the top, at the bottom, or even within the controlled segment, is discussed in Section 8.3.3.

The **body** of an iterative statement is the collection of statements whose execution is controlled by the iteration statement. We use the term **pretest** to mean that the test for loop completion occurs before the loop body is executed and **posttest** to mean that it occurs after the loop body is executed. The iteration statement and the associated loop body together form an **iteration statement**.

8.3.1 Counter-Controlled Loops

A counting iterative control statement has a variable, called the **loop variable**, in which the count value is maintained. It also includes some means of specifying the **initial** and **terminal** values of the loop variable, and the difference between sequential loop variable values, often called the **stepsize**. The initial, terminal, and stepsize specifications of a loop are called the **loop parameters**.

Although logically controlled loops are more general than counter-controlled loops, they are not necessarily more commonly used. Because counter-controlled loops are more complex, their design is more demanding.

Counter-controlled loops are sometimes supported by machine instructions designed for that purpose. Unfortunately, machine architecture can outlive the prevailing approaches to programming at the time of the architecture design. For example, VAX computers have a very convenient instruction for the implementation of posttest counter-controlled loops, which Fortran had at the time of the design of the VAX (mid-1970s). But Fortran no longer had such a loop by the time VAX computers became widely used (it had been replaced by a pretest loop). Furthermore, no other widely used language of the time had a posttest counting loop.

8.3.1.1 Design Issues

There are many design issues for iterative counter-controlled statements. The nature of the loop variable and the loop parameters provide a number of

design issues. The type of the loop variable and that of the loop parameters obviously should be the same or at least compatible, but what types should be allowed? One apparent choice is integer, but what about enumeration, character, and floating-point types? Another question is whether the loop variable is a normal variable, in terms of scope, or whether it should have some special scope. Allowing the user to change the loop variable or the loop parameters within the loop can lead to code that is very difficult to understand, so another question is whether the additional flexibility that might be gained by allowing such changes is worth that additional complexity. A similar question arises about the number of times and the specific time when the loop parameters are evaluated: If they are evaluated just once, loops are simple but less flexible. Finally, what is the value of the loop variable after loop termination, if its scope extends beyond the loop?

The following is a summary of these design issues:

- What are the type and scope of the loop variable?
- Should it be legal for the loop variable or loop parameters to be changed in the loop, and if so, does the change affect loop control?
- Should the loop parameters be evaluated only once, or once for every iteration?
- What is the value of the loop variable after loop termination?

The issue of the value of the loop variable after loop termination is solved in some languages, such as Fortran 90, by making the loop variable undefined after loop termination. Other languages, such as Ada, make the scope of the loop variable the loop itself.

8.3.1.2 The `for` Statement of the C-Based Languages

The general form of C's **`for`** statement is

- `for` (expression_1; expression_2; expression_3)
- loop body

The loop body can be a single statement, a compound statement, or a null statement.

Because assignment statements in C produce results and thus can be considered expressions, the expressions in a `for` statement are often assignment statements. The first expression is for initialization and is evaluated only once, when the `for` statement execution begins. The second expression is the loop control and is evaluated before each execution of the loop body. As is usual in C, a zero value means false and all nonzero values mean true. Therefore, if the value of the second expression is zero, the `for` is terminated; otherwise, the loop body statements are executed. In C99, the expression also could be a Boolean type. A C99 Boolean type stores only the values 0 or 1. The last expression in the `for` is executed after each execution of the loop body. It is often used to increment the loop counter. An operational semantics description of the C `for` statement is shown next. Because C expressions can be used as statements, expression evaluations are shown as statements.

- expression_1
- loop:
- **if** expression_2 = 0 **goto** out
- [loop body]
- expression_3
- **goto** loop
- out: . . .

Following is an example of a skeletal C `for` statement:

```
for (count = 1; count <= 10; count++)
```

```
  . . .
}
```

All of the expressions of C's **`for`** are optional. An absent second expression is considered true, so a **`for`** without one is potentially an infinite loop. If the first and/or third expressions are absent, no assumptions are made. For example, if the first expression is absent, it simply means that no initialization takes place.

Note that C's **`for`** need not count. It can easily model counting *and* logical loop structures, as demonstrated in the next section.

The C **`for`** design choices are the following: There is no explicit loop variable and no loop parameters. All involved variables can be changed in the loop body. The expressions are evaluated in the order stated previously. Although it can create havoc, it is legal to branch into a C **`for`** loop body.

C's **`for`** is one of the most flexible, because each of the expressions can comprise multiple expressions, which in turn allow multiple loop variables that can be of any type. When multiple expressions are used in a single expression of a **`for`** statement, they are separated by commas. All C statements have values, and this form of multiple expression is no exception. The value of such a multiple expression is the value of the last component.

Consider the following **`for`** statement:

```
for (count1 = 0, count2 = 1.0;
    count1 <= 10 && count2 <= 100.0;
    sum = ++count1 + count2, count2 *= 2.5);
```

The operational semantics description of this is

- count1 = 0
- count2 = 1.0
- loop:
- **if** count1 > 10 **goto** out

- **if** count2 > 100.0 **goto** out
- count1 = count1 + 1
- sum = count1 + count2
- count2 = count2 * 2.5
- **goto** loop
- out: . . .

The example C **`for`** statement does not need and thus does not have a loop body. All the desired actions are part of the **`for`** statement itself, rather than in its body. The first and third expressions are multiple statements. In both of these cases, the whole expression is evaluated, but the resulting value is not used in the loop control.

The **`for`** statement of C99 and C++ differs from that of earlier versions of C in two ways. First, in addition to an arithmetic expression, it can use a Boolean expression for loop control. Second, the first expression can include variable definitions. For example,

```
for (int count = 0; count < len; count++) { . . . }
```

The scope of a variable defined in the **`for`** statement is from its definition to the end of the loop body.

The **`for`** statement of Java and C# is like that of C++, except that the loop control expression is restricted to **`boolean`**.

In all of the C-based languages, the last two loop parameters are evaluated with every iteration. Furthermore, variables that appear in the loop parameter expression can be changed in the loop body. Therefore, these loops can be complex and potentially unreliable.

8.3.1.3 The `for` Statement of Python

The general form of Python's **for** is

- **for** loop_variable **in** object:
- - loop body
- [**else**:
- - else clause]

The loop variable is assigned the value in the object, which is often a range, one for each execution of the loop body. After loop termination, the loop variable has the value last assigned to it. The loop variable can be changed in the loop body, but such a change has no effect on loop operation. The else clause, when present, is executed if the loop terminates normally.

Consider the following example:

```
for count in [2, 4, 6]:
   print count
```

produces

```
2
4
6
```

For most simple counting loops in Python, the **range** function is used. **range** takes one, two, or three parameters. The following examples demonstrate the actions of **range**:

```
range(5) returns [0, 1, 2, 3, 4]
range(2, 7) returns [2, 3, 4, 5, 6]
range(0, 8, 2) returns [0, 2, 4, 6]
```

Note that **range** never returns the highest value in a given parameter range.

8.3.1.4 Counter-Controlled Loops in

Functional Languages

Counter-controlled loops in imperative languages use a counter variable, but such variables do not exist in pure functional languages. Rather than iteration to control repetition, functional languages use recursion. Rather than a statement, functional languages use a recursive function. Counting loops can be simulated in functional languages as follows: The counter can be a parameter for a function that repeatedly executes the loop body, which can be specified in a second function sent to the loop function as a parameter. So, such a loop function takes the body function and the number of repetitions as parameters.

The general form of an F# function for simulating counting loops, named `forLoop` in this case, is as follows:

```
let rec forLoop loopBody reps =
    if reps <= 0  then
        ()
    else
         loopBody()
         forLoop loopBody, (reps - 1);;
```

In this function, the parameter `loopBody` is the function with the body of the loop and the parameter `reps` is the number of repetitions. The reserved word **`rec`** appears before the name of the function to indicate that it is recursive. The empty parentheses do nothing; they are there because in F# an empty statement is illegal and every **`if`** must have an else clause.

8.3.2 Logically Controlled Loops

In many cases, collections of statements must be repeatedly executed, but the repetition control is based on a Boolean expression rather than a counter. For these situations, a logically controlled loop is convenient. Actually, logically controlled loops are more general than counter-controlled loops. Every counting loop can be built with a logical loop, but the reverse is not true. Also, recall that only selection and logical loops are essential to express the

control structure of any flowchart.

8.3.2.1 Design Issues

Because they are much simpler than counter-controlled loops, logically controlled loops have fewer design issues.

- Should the control be pretest or posttest?
- Should the logically controlled loop be a special form of a counting loop or a separate statement?

8.3.2.2 Examples

The C-based programming languages include both pretest and posttest logically controlled loops that are not special forms of their counter-controlled iterative statements. The pretest and posttest logical loops have the following forms:

- **`while`** (control_expression)
- loop body

and

- **`do`**
- loop body
- **`while`** (control_expression);

These two statement forms are exemplified by the following C# code segments:

```
sum = 0;
indat = Int32.Parse(Console.ReadLine());
```

```
while (indat >= 0) {
  sum += indat;
  indat = Int32.Parse(Console.ReadLine());
}
value = Int32.Parse(Console.ReadLine());
do {
   value /= 10;
   digits ++;
}  while  (value > 0);
```

Note that all variables in these examples are integer type. The `ReadLine` method of the `Console` object gets a line of text from the keyboard. `Int32.Parse` finds the number in its string parameter, converts it to **`int`** type, and returns it.

In the pretest version of a logical loop (**`while`**), the statement or statement segment is executed as long as the expression evaluates to true. In the posttest version (**`do`**), the loop body is executed until the expression evaluates to false. In both cases, the statement can be compound. The operational semantics descriptions of those two statements follow:

- **`while`**
- loop:
- **`if`** control_expression is false **`goto`** out
- [loop body]
- **`goto`** loop
- out: . . .
- **do-while**
- loop:
- [loop body]
- **`if`** control_expression is true **`goto`** loop

It is legal in both C and C++ to branch into both **`while`** and **`do`** loop bodies. The C89 version uses an arithmetic expression for control; in C99 and C++, it may be either arithmetic or Boolean.

Java's **`while`** and **`do`** statements are similar to those of C and C++, except the control expression must be **`boolean`** type, and because Java does not have a goto, the loop bodies cannot be entered anywhere except at their beginnings.

Posttest loops are infrequently useful and also can be somewhat dangerous, in the sense that programmers sometimes forget that the loop body will always be executed at least once. The syntactic design of placing a posttest control physically after the loop body, where it has its semantic effect, helps avoid such problems by making the logic clear.

A pretest logical loop can be simulated in a purely functional form with a recursive function that is similar to the one used to simulate a counting loop statement in Section 8.3.1.5. In both cases, the loop body is written as a function. Following is the general form of a simulated logical pretest loop, written in F#:

```
let rec whileLoop test body =
    if test() then
        body()
        whileLoop test body
     else
         ();;
```

8.3.3 User-Located Loop Control Mechanisms

In some situations, it is convenient for a programmer to choose a location for loop control other than the top or bottom of the loop body. As a result, some languages provide this capability. A syntactic mechanism for user-located loop control can be relatively simple, so its design is not difficult. Such loops have the structure of infinite loops but include one or more user-located loop exits. Perhaps the most interesting question is whether a single loop or

several nested loops can be exited. The design issues for such a mechanism are the following:

- Should the conditional mechanism be an integral part of the exit?
- Should only one loop body be exited, or can enclosing loops also be exited?

C, C++, Python, Ruby, and C# have unconditional unlabeled exits (**`break`**). Java and Perl have unconditional labeled exits (**`break`** in Java, **`last`** in Perl).

Following is an example of nested loops in Java, in which there is a break out of the outer loop from the nested loop:

```
outerLoop:
  for (row = 0; row < numRows; row++)
     for  (col = 0; col < numCols; col++) {
       sum += mat[row][col];
       if (sum > 1000.0)
         break outerLoop;
     }
```

C, C++, and Python include an unlabeled control statement, **`continue`**, that transfers control to the control mechanism of the smallest enclosing loop. This is not an exit but rather a way to skip the rest of the loop statements on the current iteration without terminating the loop construct. For example, consider the following:

```
while (sum < 1000) {
  getnext(value);
  if (value < 0) continue;
  sum += value;
}
```

A negative value causes the assignment statement to be skipped, and control is transferred instead to the conditional at the top of the loop. On the other hand, in

```
while (sum < 1000) {
  getnext(value);
  if (value < 0) break;
  sum += value;
```

```
}
```

a negative value terminates the loop.

Both **`last`** and **`break`** provide for multiple exits from loops, which may seem to be somewhat of a hindrance to readability. However, unusual conditions that require loop termination are so common that such a statement is justified. Furthermore, readability is not seriously harmed, because the target of all such loop exits is the first statement after the loop (or an enclosing loop) rather than just anywhere in the program. Finally, the alternative of using multiple breaks to leave more than one level of loops is even more detrimental to readability.

The motivation for user-located loop exits is simple: They fulfill a common need for goto statements using a highly restricted branch statement. The target of a goto can be many places in the program, both above and below the goto itself. However, the targets of user-located loop exits must be below the exit and can only follow immediately at the end of a compound statement.

8.3.4 Iteration Based on Data Structures

A general data-based iteration statement uses a user-defined data structure and a user-defined function (the iterator) to go through the structure's elements. The iterator is called at the beginning of each iteration, and each time it is called, the iterator returns an element from a particular data structure in some specific order. For example, suppose a program has a user-defined binary tree of data nodes, and the data in each node must be processed in some particular order. A user-defined iteration statement for the tree would successively set the loop variable to point to the nodes in the tree, one for each iteration. The initial execution of the user-defined iteration statement needs to issue a special call to the iterator to get the first tree element. The iterator must always remember which node it presented last so that it visits all nodes without visiting any node more than once. So an iterator must be history sensitive. A user-defined iteration statement

terminates when the iterator fails to find more elements.

The **`for`** statement of the C-based languages, because of its great flexibility, can be used to simulate a user-defined iteration statement. Once again, suppose the nodes of a binary tree are to be processed. If the tree root is pointed to by a variable named `root`, and if `traverse` is a function that sets its parameter to point to the next element of a tree in the desired order, the following could be used:

```
for (ptr = root; ptr == null; ptr = traverse(ptr)) {
  . . .
}
```

In this statement, `traverse` is the iterator.

User-defined iteration statements are more important in object-oriented programming than they were in earlier software development paradigms, because users of object-oriented programming routinely use abstract data types for data structures, especially collections. In such cases, a user-defined iteration statement and its iterator must be provided by the author of the data abstraction because the representation of the objects of the type is not known to the user.

An enhanced version of the **`for`** statement was added to Java in Java 5.0. This statement simplifies iterating through the values in an array or objects in a collection that implements the `Iterable` interface. (All of the predefined generic collections in Java implement `Iterable`.) For example, if we had an `ArrayList`[4] collection named `myList` of strings, the following statement would iterate through all of its elements, setting each to `myElement`:

4. An `ArrayList` is a predefined generic collection that is actually a dynamic-length array of whatever type it is declared to store.

```
for  (String myElement : myList) { . . . }
```

This new statement is referred to as “foreach,” although its reserved word is **`for`**.

C# and F# (and the other .NET languages) also have generic library classes

for collections. For example, there are generic collection classes for lists, which are dynamic length arrays, stacks, queues, and dictionaries (hash table). All of these predefined generic collections have built-in iterators that are used implicitly with the **`foreach`** statement. Furthermore, users can define their own collections and write their own iterators, which can implement the `IEnumerator` interface, which enables the use of **`foreach`** on these collections.

For example, consider the following C# code:

```
List<String> names =  new  List<String>();
names.Add("Bob");
names.Add("Carol");
names.Add("Alice");
. . .
foreach  (String name  in  names)
  Console.WriteLine(name);
```

In Ruby, a **block** is a sequence of code, delimited by either braces or the **`do`** and **`end`** reserved words. Blocks can be used with specially written methods to create many useful constructs, including iterators for data structures. This construct consists of a method call followed by a block. A block is actually an anonymous method that is sent to the method (whose call precedes it) as a parameter. The called method can then call the block, which can produce output or objects.

Ruby predefines several iterator methods, such as `times` and `upto` for counter-controlled loops, and `each` for simple iterations of arrays and hashes. For example, consider the following example of using `times`:

```
>> 4.times {puts "Hey!"}
Hey!
Hey!
Hey!
Hey!
=> 4
```

Note that `>>` is the prompt of the interactive Ruby interpreter and `=>` is used to indicate the return value of the expression. The Ruby `puts` statement displays its parameter. In this example, the `times` method is sent to the object `4`, with the block sent along as a parameter. The `times` method calls the block

four times, producing the four lines of output. The destination object, `4`, is the return value from `times`.

The most common Ruby iterator is `each`, which is often used to go through arrays and apply a block to each element.5 For this purpose, it is convenient to allow blocks to have parameters, which, if present, appear at the beginning of the block, delimited by vertical bars (|). The following example, which uses a block parameter, illustrates the use of `each`:

5. This is similar to the map functions discussed in Chapter 15.

```
>> list = [2, 4, 6, 8]
=> [2, 4, 6, 8]
>> list.each {|value| puts value}
2
4
6
8
=> [2, 4, 6, 8]
```

In this example, the block is called for each element of the array to which the `each` method is sent. The block produces the output, which is a list of the array's elements. The return value of `each` is the array to which it is sent.

Instead of a counting loop, Ruby has the `upto` method. For example, we could have the following:

```
1.upto(5) {|x| print x, " "}
```

This produces the following output:

```
1 2 3 4 5
```

Syntax that resembles a **`for`** loop in other languages could also be used, as in the following:

```
for x in 1..5
  print x, " "
end
```

Ruby actually has no **`for`** statement—constructs like the above are converted

by Ruby into `upto` method calls.

Now we consider how blocks work. The **`yield`** statement is similar to a method call, except that there is no receiver object and the call is a request to execute the block attached to the method call, rather than a call to a method. **`yield`** is only called in a method that has been called with a block. If the block has parameters, they are specified in parentheses in the **`yield`** statement. The value returned by a block is that of the last expression evaluated in the block. It is this process that is used to implement the built-in iterators, such as `times`.

Python provides strong support for iteration. Suppose one needs to process the nodes in some user-defined data structure. Further suppose that the structure has a traversal method that goes through the nodes of the structure in the desired order. The following skeletal class definition includes such a traversal method that produces the nodes of an instance of this class, one at a time.

```
class MyStructure:
    # Other method definitions, including a constructor
   def traverse(self):
       # if there is another node:
       #    set nod to next node
       # else:
       #    return
       yield nod
```

The `traverse` method appears to be a regular Python method, but it contains a **`yield`** statement, which dramatically changes the semantics of the method. In effect, the method is run in a separate thread of control. The **`yield`** statement acts like a return. On the first call to `traverse`, **`yield`** returns the initial node of the structure. However, on the second call, it returns the second node. On all but the first call to `traverse`, it begins its execution where it left off on the previous execution. Instead of restarting at its beginning, it is resumed. Any local storage in such a method is maintained across its calls. In the case of `traverse`, subsequent calls begin their execution at the beginning of its code, but in the state that it was in its previous execution. In Python, any method that contains a **`yield`** statement is called a *generator*, because it generates data one element at a time.

Of course, one could also produce all of the nodes of the structure, store them in an array, and process them from the array. However, the number of nodes could be large, requiring a large array to store them. The approach using the iterator is more elegant and is not affected by the size of the data structure.

8.4 Unconditional Branching

An **unconditional branch statement** transfers execution control to a specified location in the program. The most heated debate in language design in the late 1960s was over the issue of whether unconditional branching should be part of any high-level language, and if so, whether its use should be restricted. The unconditional branch, or goto, is the most powerful statement for controlling the flow of execution of a program's statements. However, careless use of the goto can lead to serious problems. The goto has stunning power and great flexibility (all other control structures can be built with goto and a selector), but it is this power that makes its use dangerous. Without usage restrictions, imposed by either language design or programming standards, goto statements can make programs very difficult to read, and as a result, highly unreliable and costly to maintain.

history note

Although several thoughtful people had pointed out the potential problems of gotos earlier, it was Edsger Dijkstra who gave the computing world the first widely read exposé on the dangers of the goto. In his letter he noted, "The goto statement as it stands is just too primitive; it is too much an invitation to make a mess of one's program" (Dijkstra, 1968a). During the first few years after publication of Dijkstra's views on the goto, a large number of people argued publicly for either outright banishment or at least restrictions on the use of the goto. Among those who did not favor complete elimination was Donald Knuth (1974), who argued that there were occasions when the efficiency of the goto outweighed its harm to readability.

These problems follow directly from a goto's ability to force any program statement to follow any other in execution sequence, regardless of whether that statement precedes or follows the previously executed statement in textual order. Readability is best when the execution order of statements in a program is nearly the same as the order in which they appear—in our case,

this would mean top to bottom, which is the order to which we are accustomed. Thus, restricting gotos so they can transfer control only downward in a program partially alleviates the problem. It allows gotos to transfer control around code sections in response to errors or unusual conditions but disallows their use to build any sort of loop.

A few languages have been designed without a goto—for example, Java, Python, and Ruby. However, most currently popular languages include a goto statement. Kernighan and Ritchie (1978) call the goto infinitely abusable, but it is nevertheless included in Ritchie's language, C. The languages that have eliminated the goto have provided additional control statements, usually in the form of loop exits, to code one of the justifiable applications of the goto.

The relatively new language, C#, includes a goto, even though one of the languages on which it is based, Java, does not. One legitimate use of C#'s goto is in the `switch` statement, as discussed in Section 8.2.2.2.

All of the loop exit statements discussed in Section 8.3.3 are actually camouflaged goto statements. They are, however, severely restricted gotos and are not harmful to readability. In fact, it can be argued that they improve readability, because to avoid their use results in convoluted and unnatural code that would be much more difficult to understand.

8.5 Guarded Commands

Quite different forms of selection and loop structures were suggested by Dijkstra (1975). His primary motivation was to provide control statements that would support a program design methodology that ensured correctness during development rather than when verifying or testing completed programs. This methodology is described in Dijkstra (1976). Another motivation is the increased clarity in reasoning that is possible with guarded commands. Simply put, a selectable segment of a selection statement in a guarded-command statement can be considered independently of any other part of the statement, which is not true for the selection statements of the common programming languages.

Guarded commands are covered in this chapter because they are the basis for the linguistic mechanism developed later for concurrent programming in CSP (Hoare, 1978). Guarded commands are also used to define functions in Haskell, as discussed in Chapter 15.

Dijkstra's selection statement has the form

- **`if`** <Boolean expression> `->` <statement>
- `[]` <Boolean expression> `->` <statement>
- `[] . . .`
- `[]` <Boolean expression> `->` <statement>
- **`fi`**

The closing reserved word, **`fi`**, is the opening reserved word spelled backward. This form of closing reserved word is taken from ALGOL 68. The small blocks, called *fatbars*, are used to separate the guarded clauses and allow the clauses to be statement sequences. Each line in the selection statement, consisting of a Boolean expression (a guard) and a statement or statement sequence, is called a **guarded command**.

This selection statement has the appearance of a multiple selection, but its semantics is different. All of the Boolean expressions are evaluated each time the statement is reached during execution. If more than one expression is true, one of the corresponding statements can be nondeterministically chosen for execution. An implementation might always choose the statement associated with the first Boolean expression that evaluates to be true. But it may choose any statement associated with a true Boolean expression. So, the correctness of the program cannot depend on which statement is chosen (among those associated with true Boolean expressions). If none of the Boolean expressions are true, a run-time error occurs that causes program termination. This forces the programmer to consider and list all possibilities. Consider the following example:

```
if i = 0 -> sum := sum + i
[] i > j -> sum := sum + j
[] j > i -> sum := sum + k
fi
```

If `i = 0` and `j > i`, this statement chooses nondeterministically between the first and third assignment statements. If `i` is equal to `j` and is not zero, a run-time error occurs because none of the conditions are true.

This statement can be an elegant way of allowing the programmer to state that the order of execution, in some cases, is irrelevant. For example, to find the largest of two numbers, we can use

```
if  x >= y -> max := x
[] y >= x -> max := y
fi
```

This computes the desired result without overspecifying the solution. In particular, if `x` and `y` are equal, it does not matter which we assign to `max`. This is a form of abstraction provided by the nondeterministic semantics of the statement.

Now, consider this same process coded in a traditional programming language selector:

```
if (x >= y)
  max = x;
```

```
else
  max = y;
```

This could also be coded as follows:

```
if (x > y)
  max = x;
else
  max = y;
```

There is no practical difference between these two statements. The first assigns `x` to `max` when `x` and `y` are equal; the second assigns `y` to `max` in the same circumstance. This choice between the two statements complicates the formal analysis of the code and the correctness proof of it. This is one of the reasons why guarded commands were developed by Dijkstra.

The loop structure proposed by Dijkstra has the form

- **`do`** <Boolean expression> `->` <statement>
- `[]` <Boolean expression> `->` <statement>
- `[] . . .`
- `[]` <Boolean expression> `->` <statement>
- **`od`**

The semantics of this statement is that all Boolean expressions are evaluated on each iteration. If more than one is true, one of the associated statements is nondeterministically (perhaps randomly) chosen for execution, after which the expressions are again evaluated. When all expressions are simultaneously false, the loop terminates.

Consider the following problem: Given four integer variables, `q1`, `q2`, `q3`, and `q4`, rearrange the values of the four so that `q1 <= q2 <= q3 <= q4`. Without guarded commands, one straightforward solution is to put the four values into an array, sort the array, and then assign the values from the array back into the scalar variables `q1`, `q2`, `q3`, and `q4`. While this solution is not difficult, it requires a good deal of code, especially if the sort process must be included.

Now, consider the following code, which uses guarded commands to solve the same problem but in a more concise and elegant way.6

6. This code appears in a slightly different form in Dijkstra (1975).

```
do q1 > q2 -> temp := q1; q1 := q2; q2 := temp;
[] q2 > q3 -> temp := q2; q2 := q3; q3 := temp;
[] q3 > q4 -> temp := q3; q3 := q4; q4 := temp;
od
```

Dijkstra's guarded command control statements are interesting, in part because they illustrate how the syntax and semantics of statements can have an impact on program verification and vice versa. Program verification is virtually impossible when goto statements are used. Verification is greatly simplified if (1) only logical loops and selections are used or (2) only guarded commands are used. The axiomatic semantics of guarded commands are conveniently specified (Gries, 1981). It should be obvious, however, that there is considerably increased complexity in the implementation of the guarded commands over their conventional deterministic counterparts.

8.6 Conclusions

We have described and discussed a variety of statement-level control structures. A brief evaluation now seems to be in order.

First, we have the theoretical result that only sequence, selection, and pretest logical loops are absolutely required to express computations (Böhm and Jacopini, 1966). This result has been used by those who wish to ban unconditional branching altogether. Of course, there are already sufficient practical problems with the goto to condemn it without also using a theoretical reason. One of the main legitimate needs for gotos—premature exits from loops—can be met with restricted branch statements, such as `break`.

One obvious misuse of the Böhm and Jacopini result is to argue against the inclusion of *any* control structures beyond selection and pretest logical loops. No widely used language has yet taken that step; furthermore, we doubt that any ever will, because of the negative effect on writability and readability. Programs written with only selection and pretest logical loops are generally less natural in structure, more complex, and therefore harder to write and more difficult to read. For example, the C# multiple selection structure is a great boost to C# writability, with no obvious negatives. Another example is the counting loop structure of many languages, especially when the statement is simple.

It is not so clear that the utility of many of the other control structures that have been proposed is worth their inclusion in languages (Ledgard and Marcotty, 1975). This question rests to a large degree on the fundamental question of whether the size of languages must be minimized. Both Wirth (1975) and Hoare (1973) strongly endorse simplicity in language design. In the case of control structures, simplicity means that only a few control statements should be in a language, and they should be simple.

The rich variety of statement-level control structures that have been invented shows the diversity of opinion among language designers. After all the

invention, discussion, and evaluation, there is still no unanimity of opinion on the precise set of control statements that should be in a language. Most contemporary languages do, of course, have similar control statements, but there is still some variation in the details of their syntax and semantics. Furthermore, there is still disagreement on whether a language should include a goto; C++ and C# do, but Java and Ruby do not.

SUMMARY

Control statements occur in several categories: selection, multiple selection, iterative, and unconditional branching.

The `switch` statement of the C-based languages is representative of multiple-selection statements. The C# version eliminates the reliability problem of its predecessors by disallowing the implicit continuation from a selected segment to the following selectable segment.

A large number of different loop statements have been invented for high-level languages. C's `for` statement is the most flexible iteration statement, although its flexibility leads to some reliability problems.

Most languages have exit statements for their loops; these statements take the place of one of the most common uses of goto statements.

Data-based iterators are loop statements for processing data structures, such as linked lists, hashes, and trees. The `for` statement of the C-based languages allows the user to create iterators for user-defined data. The `foreach` statement of Perl and C# is a predefined iterator for standard data structures. In the contemporary object-oriented languages, iterators for collections are specified with standard interfaces, which are implemented by the designers of the collections.

Ruby includes iterators that are a special form of methods that are sent to various objects. The language predefines iterators for common uses, but also allows user-defined iterators.

The unconditional branch, or goto, has been part of most imperative languages. Its problems have been widely discussed and debated. The current consensus is that it should remain in most languages but that its dangers should be minimized through programming discipline.

Dijkstra's guarded commands are alternative control statements with positive theoretical characteristics. Although they have not been adopted as the

control statements of a language, part of the semantics appear in the concurrency mechanisms of CSP and the function definitions of Haskell.

REVIEW QUESTIONS

1. What is the definition of *control structure?*
2. What did Böhm and Jocopini prove about flowcharts?
3. What is the definition of *block?*
4. What is/are the design issue(s) for all selection and iteration control statements?
5. What are the design issues for selection structures?
6. What is unusual about Python's design of compound statements?
7. Under what circumstances must an F# selector have an else clause?
8. What are the common solutions to the nesting problem for two-way selectors?
9. What are the design issues for multiple-selection statements?
10. Between what two language characteristics is a trade-off made when deciding whether more than one selectable segment is executed in one execution of a multiple selection statement?
11. What is unusual about C's multiple-selection statement?
12. On what previous language was C's **`switch`** statement based?
13. Explain how C#'s switch statement is safer than that of C.
14. What are the design issues for all iterative control statements?
15. What are the design issues for counter-controlled loop statements?
16. What is a pretest loop statement? What is a posttest loop statement?

17. What is the difference between the **`for`** statement of C++ and that of Java?

18. In what way is C's **`for`** statement more flexible than that of many other languages?

19. What does the **`range`** function in Python do?

20. What contemporary languages do not include a goto?

21. What are the design issues for logically controlled loop statements?

22. What is the main reason user-located loop control statements were invented?

23. What are the design issues for user-located loop control mechanisms?

24. What advantage does Java's **`break`** statement have over C's **`break`** statement?

25. What are the differences between the **`break`** statement of C++ and that of Java?

26. What is a user-defined iteration control?

27. What Scheme function implements a multiple selection statement?

28. How does a functional language implement repetition?

29. How are iterators implemented in Ruby?

30. What language predefines iterators that can be explicitly called to iterate over its predefined data structures?

31. What common programming language borrows part of its design from Dijkstra's guarded commands?

PROBLEM SET

1. Describe three situations where a combined counting and logical looping statement is needed.

2. Study the iterator feature of CLU in Liskov et al. (1981) and determine its advantages and disadvantages.

3. Compare the set of Ada control statements with those of C# and decide which are better and why.

4. What are the pros and cons of using unique closing reserved words on compound statements?

5. What are the arguments, pros and cons, for Python's use of indentation to specify compound statements in control statements?

6. Analyze the potential readability problems with using closure reserved words for control statements that are the reverse of the corresponding initial reserved words, such as the **`case-esac`** reserved words of ALGOL 68. For example, consider common typing errors such as transposing characters.

7. Use the *Science Citation Index* to find an article that refers to Knuth (1974). Read the article and Knuth's paper and write a paper that summarizes both sides of the goto issue.

8. In his paper on the goto issue, Knuth (1974) suggests a loop control statement that allows multiple exits. Read the paper and write an operational semantics description of the statement.

9. What are the arguments both for and against the exclusive use of Boolean expressions in the control statements in Java (as opposed to also allowing arithmetic expressions, as in C++)?

10. Describe a programming situation in which the else clause in Python's

`for` statement would be convenient.

11. Describe three specific programming situations that require a posttest loop.

12. Speculate as to the reason control can be transferred into a C loop statement.

PROGRAMMING EXERCISES

1. Rewrite the following pseudocode segment using a loop structure in the specified languages:

```
  k = (j + 13) / 27
loop:
  if k > 10 then goto  out
  k = k + 1
  i = 3 * k - 1
  goto  loop
out: . . .
```

   1. C, C++, Java, or C#

   2. Python

   3. Ruby

   Assume all variables are integer type. Discuss which language, for this code, has the best writability, the best readability, and the best combination of the two.

2. Redo Programming Exercise 1, except this time make all the variables and constants floating-point type, and change the statement

   ```
   k = k + 1
   ```

   to

   ```
   k = k + 1.2
   ```

3. Rewrite the following code segment using a multiple-selection statement in the following languages:

```
if ((k == 1) || (k == 2)) j = 2 * k - 1
if ((k == 3) || (k == 5)) j = 3 * k + 1
if (k == 4) j = 4 * k - 1
if ((k == 6) || (k == 7) || (k == 8)) j = k - 2
```

1. C, C++, Java, or C#

2. Python

3. Ruby

Assume all variables are integer type. Discuss the relative merits of the use of these languages for this particular code.

4. Consider the following C program segment. Rewrite it using no gotos or **`break`**s.

```
j = -3;
for (i = 0; i < 3; i++) {
  switch  (j + 2) {
    case  3:
    case  2: j--; break;
    case  0: j += 2; break;
    default: j = 0;
  }
  if  (j > 0)  break;
 j = 3 - i
}
```

5. In a letter to the editor of *CACM*, Rubin (1987) uses the following code segment as evidence that the readability of some code with gotos is better than the equivalent code without gotos. This code finds the first row of an *n* by *n* integer matrix named `x` that has nothing but zero values.

```
for (i = 1; i <= n; i++) {
  for (j = 1; j <= n; j++)
    if (x[i][j] != 0)
     goto reject;
  println ('First all-zero row is:', i);
  break;
reject:
 }
```

Rewrite this code without gotos in one of the following languages: C, C++, Java, or C#. Compare the readability of your code to that of the example code.

6. Consider the following programming problem: The values of three integer variables—`first`, `second`, and `third`–must be placed in the three variables `max`, `mid`, and `min`, with the obvious meanings, without using arrays or user-defined or predefined subprograms. Write two solutions to this problem, one that uses nested selections and one that does not. Compare the complexity and expected reliability of the two.

7. Rewrite the C program segment of Programming Exercise 4 using **`if`** and goto statements in C.

8. Rewrite the C program segment of Programming Exercise 4 in Java without using a **`switch`** statement.

9. Translate the following call to Scheme's `COND` to C and set the resulting value to `y`.

```
(COND
 ((> x 10) x)
 ((< x 5) (* 2 x))
 ((= x 7) (+ x 10))
)
```

9 Subprograms

Subprograms are the fundamental building blocks of programs and are therefore among the most important concepts in programming language design. We now explore the design of subprograms, including parameter-passing methods, local referencing environments, overloaded subprograms, generic subprograms, and the aliasing and problematic side effects that are associated with subprograms. We also include discussions of indirectly called subprograms, closures, and coroutines.

Implementation methods for subprograms are discussed in Chapter 10.

9.1 Introduction

Two fundamental abstraction facilities can be included in a programming language: process abstraction and data abstraction. In the early history of high-level programming languages, only process abstraction was included. Process abstraction, in the form of subprograms, has been a central concept in all programming languages. In the 1980s, however, many people began to believe that data abstraction was equally important. Data abstraction is discussed in detail in Chapter 11.

The first programmable computer, Babbage's Analytical Engine, built in the 1840s, had the capability of reusing collections of instruction cards at several different places in a program. In a modern programming language, such a collection of statements is written as a subprogram. This reuse results in savings in memory space and coding time. Such reuse is also an abstraction, for the details of the subprogram's computation are replaced in a program by a statement that calls the subprogram. Instead of describing how some computation is to be done in a program, that description (the collection of statements in the subprogram) is enacted by a call statement, effectively abstracting away the details. This increases the readability of a program by emphasizing its logical structure while hiding its low-level details.

The methods of object-oriented languages are closely related to the subprograms discussed in this chapter. The primary way methods differ from subprograms is the way they are called and their associations with classes and objects. Although these special characteristics of methods are discussed in Chapter 12, the features they share with subprograms, such as parameters and local variables, are discussed in this chapter.

9.2 Fundamentals of Subprograms

9.2.1 General Subprogram Characteristics

All subprograms discussed in this chapter, except the coroutines described in Section 9.13, have the following characteristics:

- Each subprogram has a single entry point.
- The calling program unit is suspended during the execution of the called subprogram, which implies that there is only one subprogram in execution at any given time.
- Control always returns to the caller when the subprogram execution terminates.

Alternatives to these result in coroutines and concurrent units (Chapter 13).

Most subprograms have names, although some are anonymous. Section 9.12 has examples of anonymous subprograms in C#.

9.2.2 Basic Definitions

A **subprogram definition** describes the interface to and the actions of the subprogram abstraction. A **subprogram call** is the explicit request that a specific subprogram be executed. A subprogram is said to be **active** if, after having been called, it has begun execution but has not yet completed that execution. The two fundamental kinds of subprograms, procedures and functions, are defined and discussed in Section 9.2.4.

A **subprogram header**, which is the first part of the definition, serves

several purposes. First, it specifies that the following syntactic unit is a subprogram definition of some particular kind.1 In languages that have more than one kind of subprogram, the kind of the subprogram is usually specified with a special word. Second, if the subprogram is not anonymous, the header provides a name for the subprogram. Third, it may specify a list of parameters.

1. Some programming languages include both kinds of subprograms, procedures and functions.

Consider the following header examples:

```
def adder parameters):
```

This is the header of a Python subprogram named `adder`. Ruby subprogram headers also begin with **`def`**. The header of a JavaScript subprogram begins with **`function`**.

In C, the header of a function named **`adder`** might be as follows:

- `void adder (parameters)`

The reserved word **`void`** in this header indicates that the subprogram does not return a value.

The body of subprograms defines its actions. In the C-based languages (and some others—for example, JavaScript) the body of a subprogram is delimited by braces. In Ruby, an **`end`** statement terminates the body of a subprogram. As with compound statements, the statements in the body of a Python function must be indented and the end of the body is indicated by the first statement that is not indented.

One characteristic of Python functions that sets them apart from the functions of other common programming languages is that function **`def`** statements are executable. When a **`def`** statement is executed, it assigns the given name to the given function body. Until a function's **`def`** has been executed, the function cannot be called. Consider the following skeletal example:

```
if . . .
```

```
  def fun( . . . ):
    . . .
else
  def fun( . . . ):
    . . .
```

If the then clause of this selection construct is executed, that version of the function `fun` can be called, but not the version in the else clause. Likewise, if the else clause is chosen, its version of the function can be called but the one in the then clause cannot.

Ruby methods differ from the subprograms of other programming languages in several interesting ways. Ruby methods are often defined in class definitions but can also be defined outside class definitions, in which case they are considered methods of the root object, `Object`. Such methods can be called without an object receiver, as if they were functions in C or C++. If a Ruby method is called without a receiver, **`self`** is assumed. If there is no method by that name in the class, enclosing classes are searched, up to `Object`, if necessary.

The **parameter profile** of a subprogram contains the number, order, and types of its formal parameters. The **protocol** of a subprogram is its parameter profile plus, if it is a function, its return type. In languages in which subprograms have types, those types are defined by the subprogram's protocol.

Subprograms can have declarations as well as definitions. This form parallels the variable declarations and definitions in C, in which declarations are used to provide type information but not to define variables. Subprogram declarations provide the subprogram's protocol but do not include their bodies. They are necessary in languages that do not allow forward references to subprograms. In both the cases of variables and subprograms, declarations are needed for static type checking. In the case of subprograms, it is the type of the parameters that must be checked. Function declarations are common in C and C++ programs, where they are called **prototypes**. Such declarations are often placed in header files.

In most other languages (other than C and C++), subprograms do not need declarations, because there is no requirement that subprograms be defined

before they are called.

9.2.3 Parameters

Subprograms typically describe computations. There are two ways that a nonmethod subprogram can gain access to the data that it is to process: through direct access to nonlocal variables (declared elsewhere but visible in the subprogram) or through parameter passing. Data passed through parameters are accessed using names that are local to the subprogram. Parameter passing is more flexible than direct access to nonlocal variables. In essence, a subprogram with parameter access to the data that it is to process is a parameterized computation. It can perform its computation on whatever data it receives through its parameters (presuming the types of the parameters are as expected by the subprogram). If data access is through nonlocal variables, the only way the computation can proceed on different data is to assign new values to those nonlocal variables between calls to the subprogram. Extensive access to nonlocals can reduce reliability. Variables that are visible to the subprogram where access is desired often end up also being visible where access to them is not needed. This problem was discussed in Chapter 5.

Although methods also access external data through nonlocal references and parameters, the primary data to be processed by a method is the object through which the method is called. However, when a method does access nonlocal data, the reliability problems are the same as with nonmethod subprograms. Also, in an object-oriented language, method access to class variables (those associated with the class, rather than an object) is related to the concept of nonlocal data and should be avoided whenever possible. In this case, as well as the case of a C function accessing nonlocal data, the method can have the side effect of changing something other than its parameters or local data. Such changes complicate the semantics of the method and make it less reliable.

Pure functional programming languages, such as Haskell, do not have mutable data, so functions written in them are unable to change memory in any way—they simply perform calculations and return a resulting value (or

function, since functions are values in a pure functional language).

In some situations, it is convenient to be able to transmit computations, rather than data, as parameters to subprograms. In these cases, the name of the subprogram that implements that computation may be used as a parameter. This form of parameter is discussed in Section 9.6. Data parameters are discussed in Section 9.5.

The parameters in the subprogram header are called **formal parameters**. They are sometimes thought of as dummy variables because they are not variables in the usual sense: In most cases, they are bound to storage only when the subprogram is called, and that binding is often through some other program variables.

Subprogram call statements must include the name of the subprogram and a list of parameters to be bound to the formal parameters of the subprogram. These parameters are called **actual parameters**.2 They must be distinguished from formal parameters, because the two usually have different restrictions on their forms, and of course, their uses are quite different.

2. Some authors call actual parameters *arguments* and formal parameters just *parameters*.

In most programming languages, the correspondence between actual and formal parameters—or the binding of actual parameters to formal parameters—is done by position: The first actual parameter is bound to the first formal parameter and so forth. Such parameters are called **positional parameters**. This is an effective and safe method of relating actual parameters to their corresponding formal parameters, as long as the parameter lists are relatively short.

When parameter lists are long, however, it is easy for a programmer to make mistakes in the order of actual parameters in the list. One solution to this problem is to provide **keyword parameters**, in which the name of the formal parameter to which an actual parameter is to be bound is specified with the actual parameter in a call. The advantage of keyword parameters is that they can appear in any order in the actual parameter list. Python functions can be called using this technique, as in

```
sumer(length = my_length,
      list = my_array,
      sum = my_sum)
```

where the definition of `sumer` has the formal parameters `length`, `list`, and `sum`.

The disadvantage to keyword parameters is that the user of the subprogram must know the names of formal parameters.

In addition to keyword parameters, some languages, for example Python, allow positional parameters. Keyword and positional parameters can be mixed in a call, as in

```
sumer(my_length,
      sum = my_sum,
      list = my_array)
```

The only restriction with this approach is that after a keyword parameter appears in the list, all remaining parameters must be keyworded. This restriction is necessary because a position may no longer be well defined after a keyword parameter has appeared.

In Python, Ruby, C++, and PHP, formal parameters can have default values. A default value is used if no actual parameter is passed to the formal parameter in the subprogram header. Consider the following Python function header:

```
def compute_pay(income, exemptions = 1, tax_rate)
```

The `exemptions` formal parameter can be absent in a call to `compute_pay`; when it is, the value `1` is used. No comma is included for an absent actual parameter in a Python call, because the only value of such a comma would be to indicate the position of the next parameter, which in this case is not necessary because all actual parameters after an absent actual parameter must be keyworded. For example, consider the following call:

```
pay = compute_pay(20000.0, tax_rate = 0.15)
```

In C++, which does not support keyword parameters, the rules for default

parameters are necessarily different. The default parameters must appear last, because parameters are positionally associated. Once a default parameter is omitted in a call, all remaining formal parameters must have default values. A C++ function header for the `compute_pay` function can be written as follows:

```
float compute_pay(float income, float tax_rate,
                  int exemptions = 1)
```

Notice that the parameters are rearranged so that the one with the default value is last. An example call to the C++ `compute_pay` function is

```
pay = compute_pay(20000.0, 0.15);
```

In most languages that do not have default values for formal parameters, the number of actual parameters in a call must match the number of formal parameters in the subprogram definition header. However, in C, C++, Perl, and JavaScript, this is not required. When there are fewer actual parameters in a call than formal parameters in a function definition, it is the programmer's responsibility to ensure that the parameter correspondence, which is always positional, and the subprogram execution are sensible.

Although this design, which allows a variable number of parameters, is clearly prone to error, it is also sometimes convenient. For example, the `printf` function of C can print any number of items (data values and/or literals).

C# allows methods to accept a variable number of parameters, as long as they are of the same type. The method specifies its formal parameter with the **`params`** modifier. The call can send either an array or a list of expressions, whose values are placed in an array by the compiler and provided to the called method. For example, consider the following method:

```
public void DisplayList(params int[] list) {
   foreach (int next in list) {
      Console.WriteLine("Next value {0}", next);
   }
}
```

If `DisplayList` is defined for the class `MyClass` and we have the following declarations,

```
Myclass myObject = new Myclass;
int[] myList = new int[6] {2, 4, 6, 8, 10, 12};
```

`DisplayList` could be called with either of the following:

```
myObject.DisplayList(myList);
myObject.DisplayList(2, 4, 3 * x - 1, 17);
```

Ruby supports a complicated but highly flexible actual parameter configuration. The initial parameters are expressions, whose value objects are passed to the corresponding formal parameters. The initial parameters can be followed by a list of key => value pairs, which are placed in an anonymous hash and a reference to that hash is passed to the next formal parameter. These are used as a substitute for keyword parameters, which Ruby does not support. The hash item can be followed by a single parameter preceded by an asterisk. This parameter is called the *array formal parameter*. When the method is called, the array formal parameter is set to reference a new `Array` object. All remaining actual parameters are assigned to the elements of the new `Array` object. If the actual parameter that corresponds to the array formal parameter is an array, it must also be preceded by an asterisk, and it must be the last actual parameter.3 So, Ruby allows a variable number of parameters in a way similar to that of C#. Because Ruby arrays can store different types, there is no requirement that the actual parameters passed to the array have the same type.

3. Not quite true, because the array formal parameter can be followed by a method or function reference, which is preceded by an ampersand (&).

The following example skeletal function definition and call illustrate the parameter structure of Ruby:

```
list = [2, 4, 6, 8]
def tester(p1, p2, p3, *p4)
  . . .
end  . . .
tester('first', mon => 72, tue => 68, wed => 59, *list)
```

Inside `tester`, the values of its formal parameters are as follows:

```
p1 is 'first'
p2 is {mon => 72, tue => 68, wed => 59}
```

```
p3 is 2
p4 is [4, 6, 8]
```

Python supports parameters that are similar to those of Ruby.

9.2.4 Procedures and Functions

There are two distinct categories of subprograms—procedures and functions—both of which can be viewed as approaches to extending the language. Subprograms are collections of statements that define parameterized computations. Functions return values and procedures do not. In most languages that do not include procedures as a separate form of subprogram, functions can be defined not to return values and they can be used as procedures. The computations of a procedure are enacted by single call statements. In effect, procedures define new statements. For example, if a particular language does not have a sort statement, a user can build a procedure to sort arrays of data and use a call to that procedure in place of the unavailable sort statement. Only some older languages, such as Fortran and Ada, support procedures.

Procedures can produce results in the calling program unit by two methods: (1) If there are variables that are not formal parameters but are still visible in both the procedure and the calling program unit, the procedure can change them; and (2) if the procedure has formal parameters that allow the transfer of data to the caller, those parameters can be changed.

Functions structurally resemble procedures but are semantically modeled on mathematical functions. If a function is a faithful model, it produces no side effects; that is, it modifies neither its parameters nor any variables defined outside the function. Such a function returns a value—that is its only desired effect. The functions in most programming languages have side effects.

Functions are called by appearances of their names in expressions, along with the required actual parameters. The value produced by a function's execution is returned to the calling code, effectively replacing the call itself. For example, the value of the expression `f(x)` is whatever value `f` produces when called with the parameter `x`. For a function that does not produce side effects,

the returned value is its only effect.

Functions define new user-defined operators. For example, if a language does not have an exponentiation operator, a function can be written that returns the value of one of its parameters raised to the power of another parameter. Its header in C++ could be

```
float power(float base, float exp)
```

which could be called with

```
result = 3.4 * power(10.0, x)
```

The standard C++ library includes a similar function named `pow`. Compare this with the same operation in Perl, in which exponentiation is a built-in operation:

```
result = 3.4 * 10.0 ** x
```

In some programming languages, users are permitted to overload operators by defining new functions for operators. User-defined overloaded operators are discussed in Section 9.11.

9.3 Design Issues for Subprograms

Subprograms are complex structures, and it follows from this that a lengthy list of issues is involved in their design. One obvious issue is the choice of one or more parameter-passing methods that will be used. The wide variety of approaches that have been used in various languages is a reflection of the diversity of opinion on the subject. A closely related issue is whether the types of actual parameters will be type checked against the types of the corresponding formal parameters.

The nature of the local environment of a subprogram dictates to some degree the nature of the subprogram. The most important question here is whether local variables are statically or dynamically allocated.

Next, there is the question of whether subprogram definitions can be nested. Another issue is whether subprogram names can be passed as parameters. If subprogram names can be passed as parameters and the language allows subprograms to be nested, there is the question of the correct referencing environment of a subprogram that has been passed as a parameter.

As seen in Chapter 5, side effects of functions can cause problems. So, restrictions on side effects are a design issue for functions. The types and number of values that can be returned from functions are other design issues.

Finally, there are the questions of whether subprograms can be overloaded or generic. An **overloaded subprogram** is one that has the same name as another subprogram in the same referencing environment. A **generic subprogram** is one whose computation can be done on data of different types in different calls. A **closure** is a nested subprogram and its referencing environment, which together allow the subprogram to be called from anywhere in a program.

The following is a summary of these design issues for subprograms in general. Additional issues that are specifically associated with functions are discussed in Section 9.10.

- Are local variables statically or dynamically allocated?
- Can subprogram definitions appear in other subprogram definitions?
- What parameter-passing method or methods are used?
- Are the types of the actual parameters checked against the types of the formal parameters?
- If subprograms can be passed as parameters and subprograms can be nested, what is the referencing environment of a passed subprogram?
- Are functional side effects allowed?
- What types of values can be returned from functions?
- How many values can be returned from functions?
- Can subprograms be overloaded?
- Can subprograms be generic?
- If the language allows nested subprograms, are closures supported?

These issues and example designs are discussed in the following sections.

9.4 Local Referencing Environments

This section discusses the issues related to variables that are defined within subprograms. The issue of nested subprogram definitions is also briefly covered.

9.4.1 Local Variables

Subprograms can define their own variables, thereby defining local referencing environments. Variables that are defined inside subprograms are called **local variables**, because their scope is usually the body of the subprogram in which they are defined.

In the terminology of Chapter 5, local variables can be either static or stack dynamic. If local variables are stack dynamic, they are bound to storage when the subprogram begins execution and are unbound from storage when that execution terminates. There are several advantages of stack-dynamic local variables, the primary one being flexibility. It is essential that recursive subprograms have stack-dynamic local variables. Another advantage of stack-dynamic locals is that the storage for local variables in an active subprogram can be shared with the local variables in all inactive subprograms. This is not as important an advantage as it was when computers had smaller memories.

The main disadvantages of stack-dynamic local variables are the following: First, there is the cost of the time required to allocate, initialize (when necessary), and deallocate such variables for each call to the subprogram. Second, accesses to stack-dynamic local variables must be indirect, whereas accesses to static variables can be direct.[4] This indirectness is required because the place in the stack where a particular local variable will reside can be determined only during execution (see Chapter 10). Finally, when all local

variables are stack dynamic, subprograms cannot be history sensitive; that is, they cannot retain data values of local variables between calls. It is sometimes convenient to be able to write history-sensitive subprograms. A common example of a need for a history-sensitive subprogram is one whose task is to generate pseudorandom numbers. Each call to such a subprogram computes one pseudorandom number, using the last one it computed. It must, therefore, store the last one in a static local variable. Coroutines and the subprograms used in iterator loop constructs (discussed in Chapter 8) are other examples of subprograms that need to be history sensitive.

4. In some implementations, static variables are also accessed indirectly, thereby eliminating this disadvantage.

The primary advantage of static local variables over stack-dynamic local variables is that they are slightly more efficient—they require no run-time overhead for allocation and deallocation. Also, if accessed directly, these accesses are obviously more efficient. And, of course, they allow subprograms to be history sensitive. The greatest disadvantage of static local variables is their inability to support recursion. Also, their storage cannot be shared with the local variables of other inactive subprograms.

In most contemporary languages, local variables in a subprogram are by default stack dynamic. In C and C++ functions, locals are stack dynamic unless specifically declared to be **`static`**. For example, in the following C (or C++) function, the variable `sum` is static and `count` is stack dynamic.

```
int adder(int list[], int listlen) {
  static int sum = 0;
  int count;
  for (count = 0; count < listlen; count ++)
    sum += list [count];
  return  sum;
}
```

The methods of C++, Java, and C# have only stack-dynamic local variables.

In Python, the only declarations used in method definitions are for globals. Any variable declared to be global in a method must be a variable defined outside the method. A variable defined outside the method can be referenced

in the method without declaring it to be global, but such a variable cannot be assigned in the method. If the name of a global variable is assigned in a method, it is implicitly declared to be a local and the assignment does not disturb the global. All local variables in Python methods are stack dynamic.

9.4.2 Nested Subprograms

The idea of nesting subprograms originated with ALGOL 60. The motivation was to be able to create a hierarchy of both logic and scopes. If a subprogram is needed only within another subprogram, why not place it there and hide it from the rest of the program? Because static scoping is usually used in languages that allow subprograms to be nested, this also provides a highly structured way to grant access to nonlocal variables in enclosing subprograms. Recall that in Chapter 5, the problems introduced by this were discussed. For a long time, the only languages that allowed nested subprograms were those directly descending from ALGOL 60, which were ALGOL 68, Pascal, and Ada. Many other languages, including all of the direct descendants of C, do not allow subprogram nesting. Recently, some new languages again allow it. Among these are JavaScript, Python, and Ruby. Also, most functional programming languages allow subprograms to be nested.

9.5 Parameter-Passing Methods

Parameter-passing methods are the ways in which parameters are transmitted to and/or from called subprograms. First, we focus on the different semantics models of parameter-passing methods. Then, we discuss the various implementation models invented by language designers for these semantics models. Next, we survey the design choices of several languages and discuss the actual methods used to implement the implementation models. Finally, we consider the design considerations that face a language designer in choosing among the methods.

9.5.1 Semantics Models of Parameter Passing

Formal parameters are characterized by one of three distinct semantics models: (1) They can receive data from the corresponding actual parameter; (2) they can transmit data to the actual parameter; or (3) they can do both. These models are called **in mode**, **out mode**, and **inout mode**, respectively. For example, consider a subprogram that takes two arrays of `int` values as parameters—`list1` and `list2`. The subprogram must add `list1` to `list2` and return the result as a revised version of `list2`. Furthermore, the subprogram must create a new array from the two given arrays and return it. For this subprogram, `list1` should be in mode, because it is not to be changed by the subprogram. `list2` must be inout mode, because the subprogram needs the given value of the array and must return its new value. The third array should be out mode, because there is no initial value for this array and its computed value must be returned to the caller.

There are two conceptual models of how data transfers take place in parameter transmission: Either an actual value is copied (to the caller, to the called, or both ways) or an access path is transmitted. Most commonly, the access path is a simple pointer or reference. Figure 9.1 illustrates the three

semantics models of parameter passing when values are copied.

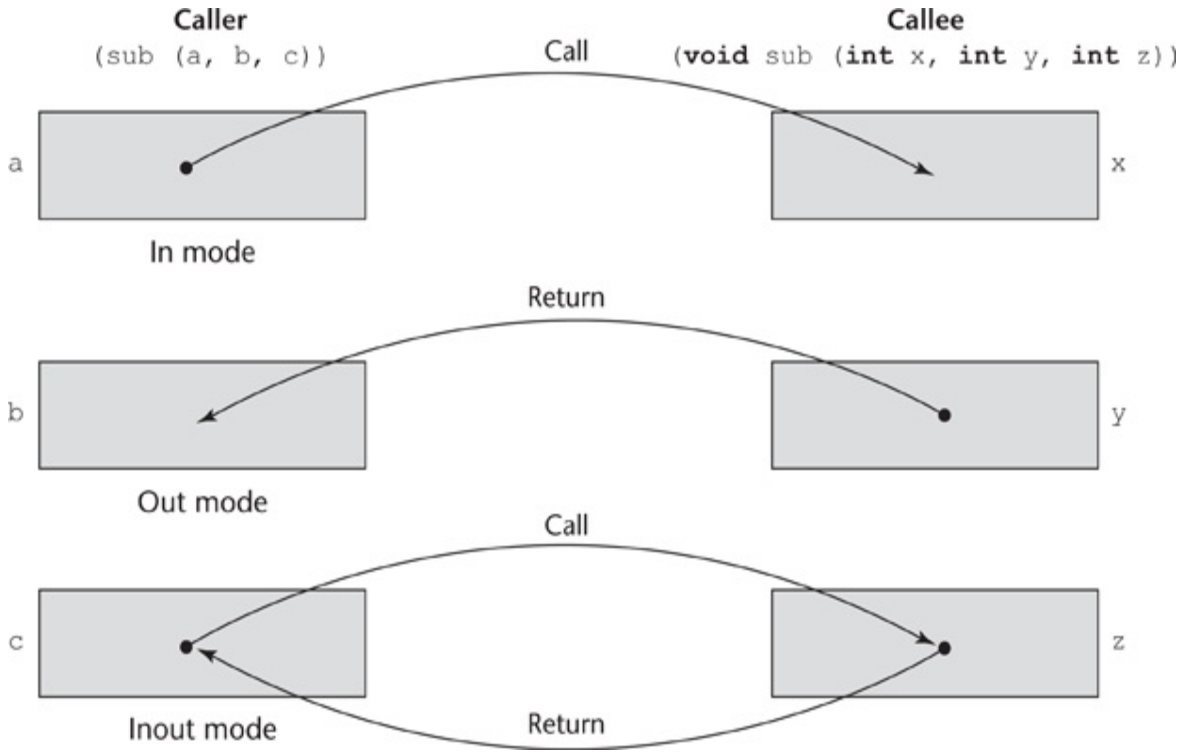


Figure 9.1 The three semantics models of parameter passing when physical moves are used

Figure 9.1 Full Alternative Text

9.5.2 Implementation Models of Parameter Passing

A variety of models have been developed by language designers to guide the implementation of the three basic parameter transmission modes. In the following sections, we discuss several of these, along with their relative

strengths and weaknesses.

9.5.2.1 Pass-by-Value

When a parameter is **passed by value**, the value of the actual parameter is used to initialize the corresponding formal parameter, which then acts as a local variable in the subprogram, thus implementing in-mode semantics.

Pass-by-value is normally implemented by copy, because accesses often are more efficient with this approach. It could be implemented by transmitting an access path to the value of the actual parameter in the caller, but that would require that the value be in a write-protected cell (one that can only be read). Enforcing the write protection is not always a simple matter. For example, suppose the subprogram to which the parameter was passed passes it in turn to another subprogram. This is another reason to use copy transfer. As we will see in Section 9.5.4, C++ provides a convenient and effective method for specifying write protection on pass-by-value parameters that are transmitted by access path.

The advantage of pass-by-value is that for scalars it is fast, in both linkage cost and access time.

The main disadvantage of the pass-by-value method if copies are used is that additional storage is required for the formal parameter, either in the called subprogram or in some area outside both the caller and the called subprogram. In addition, the actual parameter must be copied to the storage area for the corresponding formal parameter. The storage and the copy operations can be costly if the parameter is large, such as an array with many elements.

9.5.2.2 Pass-by-Result

Pass-by-result is an implementation model for out-mode parameters. When a parameter is passed by result, no value is transmitted to the subprogram. The corresponding formal parameter acts as a local variable, but just before

control is transferred back to the caller, its value is transmitted back to the caller's actual parameter, which obviously must be a variable. (How would the caller reference the computed result if it were a literal or an expression?)

The pass-by-result method has the advantages and disadvantages of pass-by-value, plus some additional disadvantages. If values are returned by copy (as opposed to access paths), as they typically are, pass-by-result also requires the extra storage and the copy operations that are required by pass-by-value. As with pass-by-value, the difficulty of implementing pass-by-result by transmitting an access path usually results in it being implemented by copy. In this case, the problem is in ensuring that the initial value of the actual parameter is not used in the called subprogram.

One additional problem with the pass-by-result model is that there can be an actual parameter collision, such as the one created with the call

```
sub(p1, p1)
```

In `sub`, assuming the two formal parameters have different names, the two can obviously be assigned different values. Then, whichever of the two is copied to their corresponding actual parameter last becomes the value of `p1` in the caller. Thus, the order in which the actual parameters are copied determines their value. For example, consider the following C# method, which specifies the pass-by-result method with the `out` specifier on its formal parameter.[5]

5. The **`out`** specifier must also be specified on the corresponding actual parameter.

```
void Fixer(out int x, out int y) {
  x = 17;
  y = 35;
}
. . .
f.Fixer(out a, out a);
```

If, at the end of the execution of `Fixer`, the formal parameter `x` is assigned to its corresponding actual parameter first, then the value of the actual parameter `a` in the caller will be `35`. If `y` is assigned first, then the value of the actual parameter `a` in the caller will be `17`.

Because the order can be implementation dependent for some languages, different implementations can produce different results.

Calling a subprogram with two identical actual parameters can also lead to different kinds of problems when other parameter-passing methods are used, as discussed in Section 9.5.2.4.

Another problem that can occur with pass-by-result is that the implementor may be able to choose between two different times to evaluate the addresses of the actual parameters: at the time of the call or at the time of the return. For example, consider the following C# method and following code:

```
void DoIt(out int x, int index){
  x = 17;
  index = 42;
}
. . .
sub = 21;
f.DoIt(list[sub], sub);
```

The address of `list[sub]` changes between the beginning and end of the method. The implementor must choose the time to bind this parameter to an address—at the time of the call or at the time of the return. If the address is computed on entry to the method, the value `17` will be returned to `list[21]`; if computed just before return, `17` will be returned to `list[42]`. This makes programs unportable between an implementation that chooses to evaluate the addresses for out-mode parameters at the beginning of a subprogram and one that chooses to do that evaluation at the end. An obvious way to avoid this problem is for the language designer to specify when the address to be used to return the parameter value must be computed.

9.5.2.3 Pass-by-Value-Result

Pass-by-value-result is an implementation model for inout-mode parameters in which actual values are copied. It is in effect a combination of pass-by-value and pass-by-result. The value of the actual parameter is used to initialize the corresponding formal parameter, which then acts as a local

variable. In fact, pass-by-value-result formal parameters must have local storage associated with the called subprogram. At subprogram termination, the value of the formal parameter is transmitted back to the actual parameter.

Pass-by-value-result is sometimes called **pass-by-copy**, because the actual parameter is copied to the formal parameter at subprogram entry and then copied back at subprogram termination.

Pass-by-value-result shares with pass-by-value and pass-by-result the - disadvantages of requiring multiple storage for parameters and time for copying values. It shares with pass-by-result the problems associated with the order in which actual parameters are assigned.

The advantages of pass-by-value-result are relative to pass-by-reference, so they are discussed in Section 9.5.2.4.

9.5.2.4 Pass-by-Reference

Pass-by-reference is a second implementation model for inout-mode parameters. Rather than copying data values back and forth, however, as in pass-by-value-result, the pass-by-reference method transmits an access path, usually just an address, to the called subprogram. This provides the access path to the cell storing the actual parameter. Thus, the called subprogram is allowed to access the actual parameter in the calling program unit. In effect, the actual parameter is shared with the called subprogram.

The advantage of pass-by-reference is that the passing process itself is efficient, in terms of both time and space. Duplicate space is not required and no copying is required.

There are, however, several disadvantages to the pass-by-reference method. First, access to the formal parameters will be slower than pass-by-value parameters, because of the additional level of indirect addressing that is required.6 Second, if only one-way communication to the called subprogram is required, inadvertent and erroneous changes may be made to the actual parameter. This issue is addressed below.

6. This is further explained in Section 9.5.3.

Another problem of pass-by-reference is that aliases can be created. This problem should be expected, because pass-by-reference makes access paths available to the called subprograms, thereby providing access to nonlocal variables. The problem with these kinds of aliasing is the same as in other circumstances: It is harmful to readability and thus to reliability. It also makes program verification more difficult. Another issue with pass by reference is whether the called subprogram is allowed to change a passed pointer. In C, this is possible, but in some other languages, such as Pascal and C++, formal parameters that are addresses are implicitly dereferenced in the called subprogram, which prevents such changes.

There are several ways pass-by-reference parameters can create aliases. First, collisions can occur between actual parameters. Consider a C++ function that has two parameters that are to be passed by reference, as in

```
void fun(int &first, int &second)
```

If the call to `fun` happens to pass the same variable twice, as in

```
fun(total, total)
```

then `first` and `second` in `fun` will be aliases.

Second, collisions between array elements can also cause aliases. For example, suppose the function `fun` is called with two array elements that are specified with variable subscripts, as in

```
fun(list[i], list[j])
```

If these two parameters are passed by reference and `i` happens to be equal to `j`, then `first` and `second` are again aliases.

Third, if two of the formal parameters of a subprogram are an element of an array and the whole array, and both are passed by reference, then a call such as

```
fun1(list[i], list)
```

could result in aliasing in `fun1`, because `fun1` can access all elements of `list` through the second parameter and access a single element through its first parameter.

Still another way to get aliasing with pass-by-reference parameters is through collisions between formal parameters and nonlocal variables that are visible. For example, consider the following C code:

```
int * global;
void main() {
   . . .
   sub(global);
   . . .
}
 void  sub(int  * param) {
   . . .
}
```

Inside `sub`, `param` and `global` are aliases.

All these possible aliasing situations are eliminated if pass-by-value-result is used instead of pass-by-reference. However, in place of aliasing, other problems sometimes arise, as discussed in Section 9.5.2.3.

9.5.2.5 Pass-by-Name

Pass-by-name is an inout-mode parameter transmission method that does not correspond to a single implementation model. When parameters are passed by name, the actual parameter is, in effect, textually substituted for the corresponding formal parameter in all its occurrences in the subprogram. This method is quite different from those discussed thus far; in which case, formal parameters are bound to actual values or addresses at the time of the subprogram call. A pass-by-name formal parameter is bound to an access method at the time of the subprogram call, but the actual binding to a value or an address is delayed until the formal parameter is assigned or referenced. Implementing a pass-by-name parameter requires a subprogram to be passed to the called subprogram to evaluate the address or value of the formal parameter. The referencing environment of the passed subprogram must also

be passed. This subprogram/referencing environment is a closure (see Section 9.12).7 Pass-by-name parameters are both complex to implement and inefficient. They also add significant complexity to the program, thereby lowering its readability and reliability.

7. These closures were originally (in ALGOL 60) called *thunks*.

Because pass-by-name is not part of any widely used language, it is not discussed further here. However, it is used at compile time by the macros in assembly languages and for the generic parameters of the generic subprograms in C++, Java 5.0, and C# 2005, as discussed in Section 9.9.

9.5.3 Implementing Parameter-Passing Methods

We now address the question of how the various implementation models of parameter passing are actually implemented.

In most contemporary languages, parameter communication takes place through the run-time stack. The run-time stack is initialized and maintained by the run-time system, which manages the execution of programs. The run-time stack is used extensively for subprogram control linkage and parameter passing, as discussed in Chapter 10. In the following discussion, we assume that the stack is used for all parameter transmission.

Pass-by-value parameters have their values copied into stack locations. The stack locations then serve as storage for the corresponding formal parameters. Pass-by-result parameters are implemented as the opposite of pass-by-value. The values assigned to the pass-by-result actual parameters are placed in the stack, where they can be retrieved by the calling program unit upon termination of the called subprogram. Pass-by-value-result parameters can be implemented directly from their semantics as a combination of pass-by-value and pass-by-result. The stack location for such a parameter is initialized by the call and is then used like a local variable in the called subprogram.

Pass-by-reference parameters are perhaps the simplest to implement. Most languages only allow variables to be passed by reference. However, Fortran passes all forms of parameters by reference. In Fortran, regardless of the type of the actual parameter, only its address must be placed in the stack. In the case of literals, the address of the literal is put in the stack. In the case of an expression, the compiler must build code to evaluate the expression, which must be executed just before the transfer of control to the called subprogram. The address of the memory cell in which the code places the result of its evaluation is then put in the stack. The Fortran compiler must prevent the called subprogram from changing parameters that are literals or expressions.

Access to the formal parameters in the called subprogram is by indirect addressing from the stack location of the address. The implementation of pass-by-value, -result, -value-result, and -reference, where the run-time stack is used, is shown in Figure 9.2. Subprogram `sub` is called from `main` with the call `sub(w, x, y, z)`, where `w` is passed by value, `x` is passed by result, `y` is passed by value-result, and `z` is passed by reference.

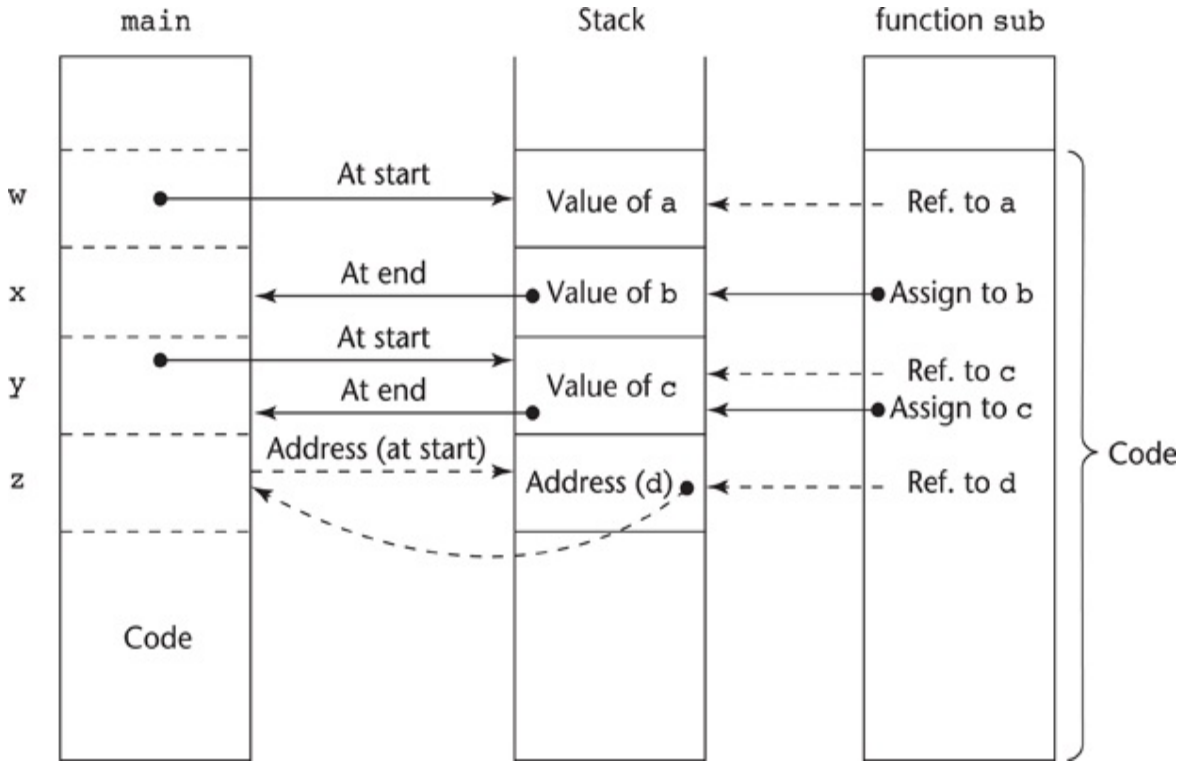


Figure 9.2 One possible stack

implementation of the common parameter-passing methods

Figure 9.2 Full Alternative Text

Function header: `void sub (int a, int b, int c, int d)`

Function call in `main`: `sub (w,x,y,z)`

(pass `w` by value, `x` by result, `y` by value-result, `z` by reference)

9.5.4 Parameter-Passing Methods of Some Common Languages

C uses pass-by-value. Pass-by-reference (inout mode) semantics is achieved by using pointers as parameters. The value of the pointer is made available to the called function and nothing is copied back. However, because what was passed is an access path to the data of the caller, the called function can change the caller's data. But all references to pointer formal parameter must be explicitly dereferenced in the function. C copied this use of the pass-by-value method from ALGOL 68. In both C and C++, formal parameters can be typed as pointers to constants. The corresponding actual parameters need not be constants, for in such cases they are coerced to constants. This allows pointer parameters to provide the efficiency of pass-by-reference with the one-way semantics of pass-by-value. Write protection of those parameters in the called function is implicitly specified.

C++ includes a special pointer type, called a *reference type*, as discussed in Chapter 6, which is often used for parameters. Reference parameters are implicitly dereferenced in the function or method, and their semantics is pass-by-reference. C++ also allows reference parameters to be defined to be constants. For example, we could have

```
void fun(const int &p1, int p2, int &p3) { . . . }
```

where `p1` is pass-by-reference but cannot be changed in the function `fun`, `p2` is pass-by-value, and `p3` is pass-by-reference. Neither `p1` nor `p3` need be explicitly dereferenced in `fun`.

history note

ALGOL 60 introduced the pass-by-name method. It also allowed pass-by-value as an option. Primarily because of the difficulty in implementing them, pass-by-name parameters were not carried from ALGOL 60 to any subsequent languages that became popular (other than SIMULA 67).

Constant parameters and in-mode parameters are not exactly alike. Constant parameters clearly implement in mode. However, in all of the common imperative languages except Ada, in-mode parameters can be assigned in the subprogram even though those changes are never reflected in the values of the corresponding actual parameters. In Ada, such an assignment is illegal. Constant parameters can never be assigned.

As with C and C++, all Java parameters are passed by value. However, because objects can be accessed only through reference variables, object parameters are in effect passed by reference. Although an object reference passed as a parameter cannot itself be changed in the called subprogram, the referenced object can be changed if a method is available to cause the change. Because reference variables cannot point to scalar variables directly and Java does not have pointers, scalars cannot be passed by reference in Java (although a reference to an object that contains a scalar can). Therefore, if a scalar is passed to a Java method, it cannot be changed by that method.

history note

ALGOL W (Wirth and Hoare, 1966) introduced the pass-by-value-result method of parameter passing as an alternative to the inefficiency of pass-by-name and the problems of pass-by-reference.

The default parameter-passing method of C# is pass-by-value. Pass-by-reference can be specified by preceding both a formal parameter and its corresponding actual parameter with **`ref`**. For example, consider the following C# skeletal method and call:

```
void sumer(ref int oldSum, int newOne) { . . . }
. . .
sumer(ref sum, newValue);
```

The first parameter to `sumer` is passed by reference; the second is passed by value. All **`ref`** parameters must be assigned a value before they are passed to an actual parameter.

C# also supports out-mode parameters, which are pass-by-reference parameters that do not need initial values. Such parameters are specified in the formal parameter list with the **`out`** modifier.

PHP's parameter passing is similar to that of C#, except that either the actual parameter or the formal parameter can specify pass-by-reference. Pass-by-reference is specified by preceding one or both of the parameters with an ampersand.

In Swift, the default parameter passing method is pass by value, and formal parameters passed this way cannot be changed in the called subprogram. Pass-by-reference semantics can be specified by preceding the formal parameter with the reserved word **`inout`**.

Perl employs a primitive means of passing parameters. All actual parameters are implicitly placed in a predefined array named `@_` (of all things!). The subprogram retrieves the actual parameter values (or addresses) from this array. The most peculiar thing about this array is its magical nature, exposed by the fact that its elements are in effect aliases for the actual parameters. Therefore, if an element of `@_` is changed in the called subprogram, that change is reflected in the corresponding actual parameter in the call, assuming there is a corresponding actual parameter (the number of actual parameters need not be the same as the number of formal parameters) and it is a variable.

The parameter-passing method of Python and Ruby is called **pass-by--**

assignment or **pass-by-sharing**. Because all data values are objects, every variable is a reference to an object. In pass-by-assignment, the actual parameter value is assigned to the formal parameter. Therefore, pass-by-assignment is in effect pass-by-reference, because the value of all actual parameters are references. However, only in certain cases does this result in pass-by-reference semantics. For example, many objects are essentially immutable. In a pure object-oriented language, the process of changing the value of a variable with an assignment statement, as in

```
x = x + 1
```

does not change the object referenced by `x`. Rather, it takes the object referenced by `x`, increments it by `1`, thereby creating a new object (with the value `x + 1`), and then changes `x` to reference the new object. So, when a reference to a scalar object is passed to a subprogram, the object being referenced cannot be changed in place. Because the reference is passed by value, even though the formal parameter is changed in the subprogram, that change has no effect on the actual parameter in the caller.

Now, suppose a reference to an array is passed as a parameter. If the corresponding formal parameter is assigned a new array object, there is no effect on the caller. However, if the formal parameter is used to assign a value to an element of the array, as in

```
list[3] = 47
```

the actual parameter is affected. So, changing the reference of the formal parameter has no effect on the caller, but changing an element of the array that is passed as a parameter does.

9.5.5 Type Checking Parameters

It is now widely accepted that software reliability demands that the types of actual parameters be checked for consistency with the types of the corresponding formal parameters. Without such type checking, small typographical errors can lead to program errors that may be difficult to diagnose because they are not detected by the compiler or the run-time

system. For example, in the function call

```
result = sub1(1)
```

the actual parameter is an integer constant. If the formal parameter of `sub1` is a floating-point type, no error will be detected without parameter type checking. Although an integer 1 and a floating-point 1 have the same value, the representations of these two are very different. `sub1` cannot produce a correct result given an integer actual parameter value if it expects a floating-point value.

Early programming languages, such as Fortran 77 and the original version of C, did not require parameter type checking; most later languages require it. However, the relatively recent languages Perl, JavaScript, and PHP do not.

C and C++ require some special discussion in the matter of parameter type checking. In the original C, neither the number of parameters nor their types were checked. In C89, the formal parameters of functions can be defined in two ways. They can be defined as in the original C; that is, the names of the parameters are listed in parentheses and the type declarations for them follow, as in the following function:

```
double sin(x)
  double x;
  { . . . }
```

Using this form avoids type checking, thereby allowing calls such as

```
double value;
int count;
. . .
value = sin(count);
```

to be legal, although they are never correct.

The alternative to the original C definition approach is called the **prototype** method, in which the formal parameter types are included in the list, as in

```
double sin(double x)
 { . . . }
```

If this version of `sin` is called with the same call, that is, with the following, it is also legal:

```
value = sin(count);
```

The type of the actual parameter (**`int`**) is checked against that of the formal parameter (**`double`**). Although they do not match, **`int`** is coercible to **`double`** (it is a widening coercion), so the conversion is done. If the conversion is not possible (for example, if the actual parameter had been an array) or if the number of parameters is wrong, then a semantics error is detected. So in C89, the user chooses whether parameters are to be type checked.

In C99 and C++, all functions must have their formal parameters in prototype form. However, type checking can be avoided for some of the parameters by replacing the last part of the parameter list with an ellipsis, as in

```
int  printf(const char* format_string, . . .);
```

A call to `printf` must include at least one parameter, a pointer to a literal character string. Beyond that, anything (including nothing) is legal. The way `printf` determines whether there are additional parameters is by the presence of format codes in the string parameter. For example, the format code for integer output is `%d`. This appears as part of the string, as in the following:

```
printf("The sum is %d\n", sum);
```

The `%` tells the `printf` function that there is one more parameter.

There is one more interesting issue with actual to formal parameter coercions when primitives can be passed by reference, as in C#. Suppose a call to a method passes a **`float`** value to a **`double`** formal parameter. If this parameter is passed by value, the **`float`** value is coerced to **`double`** and there is no problem. This particular coercion is very useful, for it allows a library to provide double versions of subprograms that can be used for both **`float`** and **`double`** values. However, suppose the parameter is passed by reference. When the value of the **`double`** formal parameter is returned to the **`float`** actual parameter in the caller, the value will overflow its location. To avoid this problem, C# requires the type of a **`ref`** actual parameter to match exactly the type of its corresponding formal parameter (no coercion is allowed).

In Python and Ruby, there is no type checking of parameters, because typing in these languages is a different concept. Objects have types, but variables do not, so formal parameters are typeless. This disallows the very idea of type checking parameters.

9.5.6 Multidimensional Arrays as Parameters

The storage-mapping functions that are used to map the index values of references to elements of multidimensional arrays to addresses in memory were discussed at length in Chapter 6. In some languages, such as C and C++, when a multidimensional array is passed as a parameter to a subprogram, the compiler must be able to build the mapping function for that array while seeing only the text of the subprogram (not the calling subprogram). This is true because the subprograms can be compiled separately from the programs that call them. Consider the problem of passing a matrix to a function in C. Multidimensional arrays in C are really arrays of arrays, and they are stored in row major order. Following is a storage-mapping function for row major order for matrices when the lower bound of all indices is 0 and the element size is 1:

- address `(mat[i, j]) =` address`(mat[0,0]) + i *`

  number_of_columns `+ j`

Notice that this mapping function needs the number of columns but not the number of rows. Therefore, in C and C++, when a matrix is passed as a parameter, the formal parameter must include the number of columns in the second pair of brackets. This is illustrated in the following skeletal C program:

```
void fun(int matrix[][10]) {
 . . . }
void main() {
  int mat[5][10];
  . . .
```

```
  fun(mat);
  . . .
}
```

The problem with this method of passing matrices as parameters is that it does not allow a programmer to write a function that can accept matrices with different numbers of columns; a new function must be written for every matrix with a different number of columns. This, in effect, disallows writing flexible functions that may be effectively reusable if the functions deal with multidimensional arrays. In C and C++, there is a way around the problem because of their inclusion of pointer arithmetic. The matrix can be passed as a pointer, and the actual dimensions of the matrix also can be passed as parameters. Then, the function can evaluate the user-written storage-mapping function using pointer arithmetic each time an element of the matrix must be referenced. For example, consider the following function prototype:

```
void fun(float *mat_ptr,
         int num_rows,
         int num_cols);
```

The following statement can be used to move the value of the variable `x` to the `[row][col]` element of the parameter matrix in `fun`:

```
*(mat_ptr + (row * num_cols) + col) = x;
```

Although this works, it is obviously difficult to read, and because of its complexity, it is error prone. The difficulty with reading this can be alleviated by using a macro to define the storage-mapping function, such as

```
#define mat_ptr(r,c) (*mat_ptr + ((r) *
                     (num_cols) + (c)))
```

With this, the assignment can be written as

```
mat_ptr(row,col) = x;
```

Other languages use different approaches to dealing with the problem of passing multidimensional arrays.

In Java and C#, arrays are objects. They are all single dimensioned, but the elements can be arrays. Each array inherits a named constant (`length` in Java

and `Length` in C#) that is set to the length of the array when the array object is created. The formal parameter for a matrix appears with two sets of empty brackets, as in the following Java method:

```
float sumer(float mat[][]) {
   float sum = 0.0f;
   for (int row = 0; row < mat.length; row++) {
     for (int col = 0; col < mat[row].length; col++) {
       sum += mat[row][col];
     }  //** for (int row . . .
   } //** for (int col . . .
   return sum;
}
```

Because each array has its own length value, in a matrix the rows can have different lengths.

9.5.7 Design Considerations

Two important considerations are involved in choosing parameter-passing methods: efficiency and whether one-way or two-way data transfer is needed.

Contemporary software-engineering principles dictate that access by subprogram code to data outside the subprogram should be minimized. With this goal in mind, in-mode parameters should be used whenever no data are to be returned through parameters to the caller. Out-mode parameters should be used when no data are transferred to the called subprogram but the subprogram must transmit data back to the caller. Finally, inout-mode parameters should be used only when data must move in both directions between the caller and the called subprogram.

There is a practical consideration that is in conflict with this principle. Sometimes it is justifiable to pass access paths for one-way parameter transmission. For example, when a large array is to be passed to a subprogram that does not modify it, a one-way method may be preferred. However, pass-by-value would require that the entire array be moved to a local storage area of the subprogram. This would be costly in both time and space. Because of this, large arrays are often passed by reference. This is

precisely the reason why the Ada 83 definition allowed implementors to choose between the two methods for structured parameters. C++ constant reference parameters offer another solution. Another alternative approach would be to allow the user to choose between the methods.

The choice of a parameter-passing method for functions is related to another design issue: functional side effects. This issue is discussed in Section 9.10.

9.5.8 Examples of Parameter Passing

Consider the following C function:

```
void swap1(int a, int b) {
  int temp = a;
  a = b;
  b = temp;
}
```

Suppose this function is called with

```
swap1(c, d);
```

Recall that C uses pass-by-value. The actions of `swap1` can be described by the following pseudocode:

```
a = c        - Move first parameter value in
b = d        - Move second parameter value in
temp = a
a = b
b = temp
```

Although `a` ends up with `d`'s value and `b` ends up with `c`'s value, the values of `c` and `d` are unchanged because nothing is transmitted back to the caller.

We can modify the C swap function to deal with pointer parameters to achieve the effect of pass-by-reference:

```
void swap2(int *a, int *b) {
  int temp = *a;
  *a = *b;
  *b = temp;
}
```

swap2 can be called with

```
swap2(&c, &d);
```

The actions of swap2 can be described with the following:

```
a = &c        - Move first parameter address in
b = &d        - Move second parameter address in
temp = *a
*a = *b
*b = temp
```

In this case, the swap operation is successful: The values of c and d are in fact interchanged. swap2 can be written in C++ using reference parameters as follows:

```
void swap2(int &a, int &b) {
   int  temp = a;
  a = b;
  b = temp;
}
```

This simple swap operation is not possible in Java, because it has neither pointers nor C++'s kind of references. In Java, a reference variable can point to only an object, not a scalar value.

The semantics of pass-by-value-result is identical to those of pass-by--reference, except when aliasing is involved. Ada uses pass-by-value-result for inout-mode scalar parameters. To explore pass-by-value-result, consider the following function, swap3, which we assume uses pass-by-value-result parameters. It is written in a syntax similar to that of Ada.

```
procedure swap3(a : in out Integer, b : in out Integer) is
  temp : Integer;
   begin
   temp := a;
   a := b;
```

```
   b := temp;
   end swap3
```

Suppose `swap3` is called with

```
swap3(c, d);
```

The actions of `swap3` with this call are

```
addr_c = &c         - Move first parameter address in
addr_d = &d         - Move second parameter address in
a = *addr_c         - Move first parameter value in
b = *addr_d         - Move second parameter value in
temp = a
a = b
b = temp
*addr_c = a         - Move first parameter value out
*addr_d = b         - Move second parameter value out
```

So once again, this swap subprogram operates correctly. Next, consider the call

```
swap3(i, list[i]);
```

In this case, the actions are

```
addr_i = &i           - Move first parameter address in
addr_listi= &list[i] - Move second parameter address in
a = *addr_i           - Move first parameter value in
b = *addr_listi       - Move second parameter value in
temp = a
a = b
b = temp
*addr_i = a           - Move first parameter value out
*addr_listi = b       - Move second parameter value out
```

Again, the subprogram operates correctly, in this case because the addresses to which to return the values of the parameters are computed at the time of the call rather than at the time of the return. If the addresses of the actual parameters were computed at the time of the return, the results would be wrong.

Finally, we must explore what happens when aliasing is involved with pass-

by-value-result and pass-by-reference. Consider the following skeletal program written in C-like syntax:

```
int i = 3; /* i is a global variable */
void fun(int a, int b) {
  i = b;
}
void main() {
  int list[10];
  list[i] = 5;
  fun(i, list[i]);
}
```

In `fun`, if pass-by-reference is used, `i` and `a` are aliases. If pass-by-value-result is used, `i` and `a` are not aliases. The actions of `fun`, assuming pass-by-value-result, are as follows:

```
addr_i = &i           - Move first parameter address in
addr_listi = &list[i] - Move second parameter address in
a = *addr_i           - Move first parameter value in
b = *addr_listi       - Move second parameter value in
i = b                 - Sets i to 5
*addr_i = a           - Move first parameter value out
*addr_listi = b       - Move second parameter value out
```

In this case, the assignment to the global `i` in `fun` changes its value from `3` to `5`, but the copy back of the first formal parameter (the second to last line in the example) sets it back to `3`. The important observation here is that if pass-by-reference is used, the result is that the copy back is not part of the semantics, and `i` remains `5`. Also note that because the address of the second parameter is computed at the beginning of `fun`, any change to the global `i` has no effect on the address used at the end to return the value of `list[i]`.

9.6 Parameters That Are Subprograms

In programming, a number of situations occur that are most conveniently handled if subprogram names can be sent as parameters to other subprograms. One common example of these occurs when a subprogram must sample some mathematical function. For example, a subprogram that does numerical integration estimates the area under the graph of a function by sampling the function at a number of different points. When such a subprogram is written, it should be usable for any given function; it should not need to be rewritten for every function that must be integrated. It is therefore natural that the name of a program function that evaluates the mathematical function to be integrated be sent to the integrating subprogram as a parameter.

Although the idea is natural and seemingly simple, the details of how it works can be confusing. If only the transmission of the subprogram code was necessary, it could be done by passing a single pointer. However, two complications arise.

First, there is the matter of type checking the parameters of the activations of the subprogram that was passed as a parameter. In C and C++, functions cannot be passed as parameters, but pointers to functions can. The type of a pointer to a function includes the function's protocol. Because the protocol includes all parameter types, such parameters can be completely type checked.

The second complication with parameters that are subprograms appears only with languages that allow nested subprograms. The issue is what referencing environment for executing the passed subprogram should be used. There are three choices:

- The environment of the call statement that enacts the passed subprogram (**shallow binding**)

- The environment of the definition of the passed subprogram (**deep binding**)

- The environment of the call statement that passed the subprogram as an actual parameter (**ad hoc binding**)

The following example program, written with the syntax of JavaScript, illustrates these choices:

```
function sub1() {
  var x;
  function sub2() {
    alert(x); // Creates a dialog box with the value of x
    };
  function  sub3() {
    var  x;
    x = 3;
    sub4(sub2);
    };
  function  sub4(subx) {
    var  x;
    x = 4;
    subx();
    };
  x = 1;
  sub3();
  };
```

Consider the execution of `sub2` when it is called in `sub4`. For shallow binding, the referencing environment of that execution is that of `sub4`, so the reference to `x` in `sub2` is bound to the local `x` in `sub4`, and the output of the program is `4`. For deep binding, the referencing environment of `sub2`'s execution is that of `sub1`, so the reference to `x` in `sub2` is bound to the local `x` in `sub1`, and the output is `1`. For ad hoc binding, the binding is to the local `x` in `sub3`, and the output is `3`.

In some cases, the subprogram that declares a subprogram also passes that subprogram as a parameter. In those cases, deep binding and ad hoc binding are the same. Ad hoc binding has never been used because, one might surmise, the environment in which the procedure appears as a parameter has no natural connection to the passed subprogram.

history note

The original definition of Pascal (Jensen and Wirth, 1974) allowed subprograms to be passed as parameters without including their parameter type information. If independent compilation is possible (it was not possible in the original Pascal), the compiler is not even allowed to check for the correct number of parameters. In the absence of independent compilation, checking for parameter consistency is possible but is a very complex task, and it usually is not done.

Shallow binding is not appropriate for static-scoped languages with nested subprograms. For example, suppose the procedure `Sender` passes the procedure `Sent` as a parameter to the procedure `Receiver`. The problem is that `Receiver` may not be in the static environment of `Sent`, thereby making it highly unnatural for `Sent` to have access to `Receiver`'s variables. On the other hand, it is perfectly normal in such a language for any subprogram, including one sent as a parameter, to have its referencing environment determined by the lexical position of its definition. It is therefore more logical for these languages to use deep binding. Some dynamic-scoped languages use shallow binding.

9.7 Calling Subprograms Indirectly

There are situations in which subprograms must be called indirectly. These most often occur when the specific subprogram to be called is not known until run time. The call to the subprogram is made through a pointer or reference to the subprogram, which has been set during execution before the call is made. The two most common applications of indirect subprogram calls are for event handling in graphical user interfaces, which are now part of nearly all Web applications, as well as many non-Web applications, and for callbacks, in which a subprogram is called and instructed to notify the caller when the called subprogram has completed its work. As always, our interest is not in these specific kinds of programming, but rather in programming language support for them.

The concept of calling subprograms indirectly is not a recently developed concept. C and C++ allow a program to define a pointer to a function, through which the function can be called. In C++, pointers to functions are typed according to the return type and parameter types of the function, so that such a pointer can point only at functions with one particular protocol. For example, the following declaration defines a pointer (`pfun`) that can point to any function that takes a **`float`** and an **`int`** as parameters and returns a **`float`**:

```
float (*pfun)(float,  int);
```

Any function with the same protocol as this pointer can be used as the initial value of this pointer or be assigned to the pointer in a program. In C and C++, a function name without following parentheses, like an array name without following brackets, is the address of the function (or array). So, both of the following are legal ways of giving an initial value or assigning a value to a pointer to a function:

```
int myfun2 (int, int);  // A function declaration
int (*pfun2)(int, int) = myfun2;  // Create a pointer and
                                  // initialize
                                  // it to point to myfun2
pfun2 = myfun2;  // Assigning a function's address to a
                 // pointer
```

The function `myfun2` can now be called with either of the following statements:

```
(*pfun2)(first, second); pfun2(first, second);
```

The first of these explicitly dereferences the pointer `pfun2`, which is legal, but unnecessary.

The function pointers of C and C++ can be sent as parameters and returned from functions, although functions cannot be used directly in either of those roles.

In C#, the power and flexibility of method pointers is increased by making them objects. These are called **delegates**, because instead of calling a method, a program delegates that action to a delegate.

To use a delegate, first the delegate class must be defined with a specific method protocol. An instantiation of a delegate holds the name of a method with the delegate's protocol that it is able to call. The syntax of a declaration of a delegate is the same as that of a method declaration, except that the reserved word **`delegate`** is inserted just before the return type. For example, we could have the following:

```
public delegate int Change(int x);
```

This delegate can be instantiated with any method that takes an **`int`** as a parameter and returns an **`int`**. For example, consider the following method declaration:

```
static int fun1(int x);
```

The delegate `Change` can be instantiated by sending the name of this method to the delegate's constructor, as in the following:

```
Change chgfun1 =  new  Change(fun1);
```

This can be shortened to the following:

```
Change chgfun1 = fun1;
```

Following is an example call to `fun1` through the delegate `chgfun1`:

```
chgfun1(12);
```

Objects of a delegate class can store more than one method. A second method can be added using the operator `+=`, as in the following:

```
Change chgfun1 += fun2;
```

This places `fun2` in the `chgfun1` delegate, even if `chgfun1` previously had the value **`null`**. All of the methods stored in a delegate instance are called in the order in which they were placed in the instance. This is called a **multicast delegate**. Regardless of what is returned by the methods, only the value or object returned by the last one called is returned. Of course, this means that in most cases, **`void`** is returned by the methods called through a multicast delegate.

In our example, a static method is placed in the delegate `Change`. Instance methods can also be called through a delegate, in which case the delegate must store a reference to the method. Delegates can also be generic.

Delegates are used for event handling by .NET applications. They are also used to implement closures (see Section 9.12).

As is the case with C and C++, the name of a function in Python without the following parentheses is a pointer to that function. Ada 95 has pointers to subprograms, but Java does not. In Python and Ruby, as well as most functional languages, subprograms are treated like data, so they can be assigned to variables. Therefore, in these languages, there is little need for pointers to subprograms.

9.8 Design Issues for Functions

The following design issues are specific to functions:

- Are side effects allowed?
- What types of values can be returned?
- How many values can be returned?

9.8.1 Functional Side Effects

Because of the problems of side effects of functions that are called in expressions, as described in Chapter 5, parameters to functions should always be in-mode. In fact, some languages require this; for example, Ada functions can have only in-mode formal parameters. This requirement effectively prevents a function from causing side effects through its parameters or through aliasing of parameters and globals. In most other imperative languages, however, functions can have either pass-by-value or pass-by-reference parameters, thus allowing functions that cause side effects and aliasing.

Pure functional languages, such as Haskell and Ruby, do not have variables, so their functions cannot have side effects.

9.8.2 Types of Returned Values

Most imperative programming languages restrict the types that can be returned by their functions. C allows any type to be returned by its functions except arrays and functions. Both of these can be handled by pointer type return values. C++ is like C but also allows user-defined types, or classes, to be returned from its functions. Ada, Python, and Ruby are the only languages

among current imperative languages whose functions (and/or methods) can return values of any type. In the case of Ada, however, because functions are not types in Ada, they cannot be returned from functions. Of course, pointers to functions can be returned by functions.

In some programming languages, subprograms are first-class objects, which means that they can be passed as parameters, returned from functions, and assigned to variables. Methods are first-class objects in some imperative languages, for example, Python and Ruby. The same is true for the functions in most functional languages.

Neither Java nor C# can have functions, although their methods are similar to functions. In both, any type or class can be returned by methods. Because methods are not types, they cannot be returned.

9.8.3 Number of Returned Values

In most languages, only a single value can be returned from a function. However, that is not always the case. Ruby allows the return of more than one value from a method. If a `return` statement in a Ruby method is not followed by an expression, `nil` is returned. If followed by one expression, the value of the expression is returned. If followed by more than one expression, an array of the values of all of the expressions is returned.

In ML, F#, and Python, and some other languages that include tuples, multiple values can be returned by placing them in a tuple.

9.9 Overloaded Subprograms

An overloaded operator is one that has multiple meanings. The meaning of a particular instance of an overloaded operator is determined by the types of its operands. For example, if the * operator has two floating-point operands in a Java program, it specifies floating-point multiplication. But if the same operator has two integer operands, it specifies integer multiplication.

An **overloaded subprogram** is a subprogram that has the same name as another subprogram in the same referencing environment. Every version of an overloaded subprogram must have a unique protocol; that is, it must be different from the others in the number, order, or types of its parameters, and possibly in its return type. The meaning of a call to an overloaded subprogram is determined by the actual parameter list and/or possibly the type of the returned value. Although it is not necessary, overloaded subprograms usually implement the same process.

C++, Java, and C# include predefined overloaded subprograms. For example, many classes in C++, Java, and C# have overloaded constructors. Because each version of an overloaded subprogram has a unique parameter profile, the compiler can disambiguate occurrences of calls to them by the different type parameters. Unfortunately, it is not that simple. Parameter coercions, when allowed, complicate the disambiguation process enormously. Simply stated, the issue is that if no method's parameter profile matches the number and types of the actual parameters in a method call, but two or more methods have parameter profiles that can be matched through coercions, which method should be called? For a language designer to answer this question, he or she must decide how to rank all of the different coercions, so that the compiler can choose the method that "best" matches the call. This can be a complicated task. To understand the level of complexity of this process, we suggest the reader refer to the rules for disambiguation of method calls used in C++ (Stroustrup, 1997).

Because C++, Java, and C# allow mixed-mode expressions, the return type is irrelevant to disambiguation of overloaded functions (or methods). The

context of the call does not allow the determination of the return type. For example, if a C++ program has two functions named `fun` and both take an `int` parameter but one returns an `int` and one returns a `float`, the program would not compile, because the compiler could not determine which version of `fun` should be used.

Users are also allowed to write multiple versions of subprograms with the same name in Java, C++, C#, and F#. Once again, in C++, Java, and C# the most common user-defined overloaded methods are constructors.

Overloaded subprograms that have default parameters can lead to ambiguous subprogram calls. For example, consider the following C++ code:

```
void fun(float b = 0.0);
void fun();
  . . .
  fun();
```

The call is ambiguous and will cause a compilation error.

9.10 Generic Subprograms

Software reuse can be an important contributor to software productivity. One way to increase the reusability of software is to lessen the need to create different subprograms that implement the same algorithm on different types of data. For example, a programmer should not need to write four different sort subprograms to sort four arrays that differ only in element type.

A **polymorphic** subprogram takes parameters of different types on different activations. Overloaded subprograms provide a particular kind of polymorphism called **ad hoc polymorphism**. Overloaded subprograms need not behave similarly.

Languages that support object-oriented programming usually support subtype polymorphism. **Subtype polymorphism** means that a variable of type T can access any object of type T or any type derived from T.

A more general kind of polymorphism is provided by the methods of Python and Ruby. Recall that variables in these languages do not have types, so formal parameters do not have types. Therefore, a method will work for any type of actual parameter, as long as the operators used on the formal parameters in the method are defined.

Parametric polymorphism is provided by a subprogram that takes generic parameters that are used in type expressions that describe the types of the parameters of the subprogram. Different instantiations of such subprograms can be given different generic parameters, producing subprograms that take different types of parameters. Parametric definitions of subprograms all behave the same. Parametrically polymorphic subprograms are often called **generic** subprograms. C++, Java 5.0+, C# 2005+, and F# provide a kind of compile-time parametric polymorphism.

9.10.1 Generic Functions in C++

Generic functions in C++ have the descriptive name of *template functions*. The definition of a template function has the general form

- **`template`** <template parameters>

—a function definition that may include the template parameters

A template parameter (there must be at least one) has one of the forms

- **`class`** identifier
- **`typename`** identifier

The class form is used for type names. The `typename` form is used for passing a value to the template function. For example, it is sometimes convenient to pass an integer value for the size of an array in the template function.

A template can take another template, in practice often a template class that defines a user-defined generic type, as a parameter, but we do not consider that option here.8

8. Template classes are discussed in Chapter 11.

As an example of a template function, consider the following:

```
template <class Type>
Type max(Type first, Type second) {
  return  first > second ? first : second;
}
```

where `Type` is the parameter that specifies the type of data on which the function will operate. This template function can be instantiated for any type for which the operator `>` is defined. For example, if it were instantiated with **`int`** as the parameter, it would be

```
int max(int  first, int second) {
  return first > second ? first : second;
}
```

Although this process could be defined as a macro, a macro would have the

disadvantage of not operating correctly if the parameters were expressions with side effects. For example, suppose the macro were defined as

```
#define max(a, b) ((a) > (b)) ? (a) : (b)
```

This definition is generic in the sense that it works for any numeric type. However, it does not always work correctly if called with a parameter that has a side effect, such as

```
max(x++, y)
```

which produces

```
((x++) > (y) ? (x++) : (y))
```

Whenever the value of `x` is greater than that of `y`, `x` will be incremented twice.

C++ template functions are instantiated implicitly either when the function is named in a call or when its address is taken with the `&` operator. For example, the example template function `max` would be instantiated twice by the following code segment—once for **`int`** type parameters and once for **`char`** type parameters:

```
int a, b, c;
char d, e, f;
. . .
c = max(a, b);
f = max(d, e);
```

The following is a C++ generic sort subprogram:

```
template <class Type>
void generic_sort(Type list[], int len) {
  int top, bottom;
  Type temp;
  for (top = 0; top < len - 2; top++)
    for (bottom = top + 1; bottom < len - 1; bottom++)
      if (list[top] > list[bottom]) {
        temp = list[top];
        list[top] = list[bottom];
        list[bottom] = temp;
      }  //** end of if (list[top] . . .
}  //** end of generic_sort
```

The following is an example instantiation of this template function:

```
float flt_list[100];
. . .
generic_sort(flt_list, 100);
```

The templated functions of C++ are a kind of poor cousin to a subprogram in which the types of the formal parameters are dynamically bound to the types of the actual parameters in a call. In this case, only a single copy of the code is needed, whereas with the C++ approach, a copy must be created at compile time for each different type that is required and the binding of subprogram calls to subprograms is static.

9.10.2 Generic Methods in Java 5.0

Support for generic types and methods was added to Java in Java 5.0. The name of a generic class in Java 5.0 is specified by a name followed by one or more type variables delimited by pointed brackets. For example,

```
generic_class<T>
```

where `T` is the type variable. Generic types are discussed in more detail in Chapter 11.

Java's generic methods differ from the generic subprograms of C++ in several important ways. First, generic parameters must be classes—they cannot be primitive types. This requirement disallows a generic method that mimics our example in C++, in which the component types of arrays are generic and can be primitives. In Java, the components of arrays (as opposed to containers) cannot be generic. Second, although Java generic methods can be instantiated any number of times, only one copy of the code is built. The internal version of a generic method, which is called a *raw* method, operates on `Object` class objects. At the point where the generic value of a generic method is returned, the compiler inserts a cast to the proper type. Third, in Java, restrictions can be specified on the range of classes that can be passed

to the generic method as generic parameters. Such restrictions are called **bounds**.

As an example of a generic Java 5.0 method, consider the following skeletal method definition:

```
public static  <T> T doIt(T[] list) {
  . . .
}
```

This defines a method named `doIt` that takes an array of elements of a generic type. The name of the generic type is `T` and it must be an array. Following is an example call to `doIt`:

```
doIt<string>(myList);
```

Now, consider the following version of `doIt`, which has a bound on its generic parameter:

```
public static <T extends Comparable> T doIt(T[] list) {
  . . .
}
```

This defines a method that takes a generic array parameter whose elements are of a class that implements the `Comparable` interface. That is the restriction, or bound, on the generic parameter. The reserved word **`extends`** seems to imply that the generic class subclasses the following class. In this context, however, **`extends`** has a different meaning. The expression `<T extends BoundingType>` specifies that `T` should be a "subtype" of the bounding type. So, in this context, **`extends`** means the generic class (or interface) either extends the bounding class (the bound if it is a class) or implements the bounding interface (if the bound is an interface). The bound ensures that the elements of any instantiation of the generic can be compared with the `Comparable` method, `compareTo`.

If a generic method has two or more restrictions on its generic type, they are added to the **`extends`** clause, separated by ampersands (`&`). Also, generic methods can have more than one generic parameter.

Java 5.0 supports *wildcard types*. For example, `Collection<?>` is a wildcard

type for collection classes. This type can be used for any collection type of any class components. For example, consider the following generic method:

```
void  printCollection(Collection<?> c) {
   for  (Object e: c) {
     System.out.println(e);
  }
}
```

This method prints the elements of any `Collection` class, regardless of the class of its components. Some care must be taken with objects of the wildcard type. For example, because the components of a particular object of this type have a type, other type objects cannot be added to the collection. For example, consider

```
Collection<?> c = new ArrayList<String>();
```

It would be illegal to use the `add` method to put something into this collection unless its type were `String`.

Wildcard types can be restricted, as is the case with nonwildcard types. Such types are called *bounded wildcard types*. For example, consider the following method header:

```
public void drawAll(ArrayList<? extends Shape> things)
```

The generic type here is a wildcard type that is a subclass of the `Shape` class. This method could be written to draw any object whose type is a subclass of `Shape`.

9.10.3 Generic Methods in C# 2005

The generic methods of C# 2005 are similar in capability to those of Java 5.0, except there is no support for wildcard types. One unique feature of C# 2005 generic methods is that the actual type parameters in a call can be omitted if the compiler can infer the unspecified type. For example, consider the

following skeletal class definition:

```
class MyClass {
  public static  T DoIt<T>(T p1) {
    . . .
  }
}
```

The method `DoIt` can be called without specifying the generic parameter if the compiler can infer the generic type from the actual parameter in the call. For example, both of the following calls are legal:

```
int myInt = MyClass.DoIt(17);  // Calls DoIt<int>
string myStr = MyClass.DoIt('apples');
    // Calls DoIt<string>
```

9.10.4 Generic Functions in F#

The type inferencing system of F# is not always able to determine the type of parameters or the return type of a function. When this is the case, for some functions F# infers a generic type for the parameters and the return value. This is called **automatic generalization**. For example, consider the following function definition:

```
let getLast (a, b, c) = c;;
```

Because no type information was included, the types of the parameters and the return value are all inferred to be generic. Because this function does not include any computations, this is a simple generic function.

Functions can be defined to have generic parameters, as in the following example:

```
let printPair (x: 'a) (y: 'a) =
   printfn "%A %A" x y;;
```

The `%A` format specification is for any type. The apostrophe in front of the type named a specifies it to be a generic type.[9] This function definition works (with generic parameters) because no type-constrained operation is included.

Arithmetic operators are examples of type-constrained operations. For example, consider the following function definition:

9. There is nothing special about a—it could be any legal identifier. By convention, lowercase letters at the beginning of the alphabet are used.

```
let adder x y = x + y;;
```

Type inferencing sets the type of `x` and `y` and the return value to **`int`**. Because there is no type coercion in F#, the following call is illegal:

```
adder 2.5 3.6;;
```

Even if the type of the parameters were set to be generic, the `+` operator would cause the types of `x` and `y` to be **`int`**.

The generic type could also be specified explicitly in angle brackets, as in the following:

```
let printPair2<'T> x y =
    printfn "%A %A" x y;;
```

This function must be called with a type,10 as in the following:

10. Convention explicitly states that generic types are named with uppercase letters starting at T.

```
printPair2<float> 3.5 2.4;;
```

Because of type inferencing and the lack of type coercions, F# generic functions are far less useful, especially for numeric computations, than those of C+ +, Java 5.0+, and C# 2005+.

9.11 User-Defined Overloaded Operators

Operators can be overloaded by the user in Ada, C++, Python, and Ruby. Suppose that a Python class is developed to support complex numbers and arithmetic operations on them. A complex number can be represented with two floating-point values. The `Complex` class would have members for these two named `real` and `imag`. In Python, binary arithmetic operations are implemented as method calls sent to the first operand, sending the second operand as a parameter. For addition, the method is named `__add__`. For example, the expression `x + y` is implemented as `x.__add__(y)`. To overload `+` for the addition of objects of the new `Complex` class, we only need to provide `Complex` with a method named `__add__` that performs the operation. Following is such a method:

```
def __add__ (self, second):
  return Complex(self.real + second.real, self.imag +
   second.imag)
```

In most languages that support object-oriented programming, a reference to the current object is implicitly sent with each method call. In Python, this reference must be sent explicitly; that is the reason why `self` is the first parameter to our method, `__add__`.

The example add method could be written for a complex class in C++ as follows11 :

11. Both C++ and Python have predefined classes for complex numbers, so our example methods are unnecessary, except as illustrations.

```
Complex operator +(Complex &second) {
  return  Complex(real + second.real, imag + second.imag);
}
```

9.12 Closures

Defining a closure is a simple matter; a **closure** is a subprogram and the referencing environment where it was defined. The referencing environment is needed if the subprogram can be called from any arbitrary place in the program. Explaining a closure is not so simple.

If a static-scoped programming language does not allow nested subprograms, closures are not useful, so such languages do not support them. All of the variables in the referencing environment of a subprogram in such a language (its local variables and the global variables) are accessible, regardless of the place in the program where the subprogram is called.

When subprograms can be nested, in addition to locals and globals, the referencing environment of a subprogram can include variables defined in all enclosing subprograms. However, this is not an issue if the subprogram can be called only in places where all of the enclosing scopes are active and visible. It becomes an issue if a subprogram can be called elsewhere. This can happen if the subprogram can be passed as a parameter or assigned to a variable, thereby allowing it to be called from virtually anywhere in the program. There is an associated problem: The subprogram could be called after one or more of its nesting subprograms has terminated, which normally means that the variables defined in such nesting subprograms have been deallocated—they no longer exist. For the subprogram to be callable from anywhere in the program, its referencing environment must be available wherever it might be called. Therefore, the variables defined in nesting subprograms may need lifetimes that are of the entire program, rather than just the time during which the subprogram in which they were defined is active. A variable whose lifetime is that of the whole program is said to have **unlimited extent**. This usually means they must be heap dynamic, rather than stack dynamic.

Nearly all functional programming languages, most scripting languages, and at least one primarily imperative language, C#, support closures. These languages are static-scoped, allow nested subprograms,[12] and allow

subprograms to be passed as parameters. Following is an example of a closure written in JavaScript:

12. In C#, the only methods that can be nested are anonymous delegates and lambda expressions.

```
function makeAdder(x) {
    return function(y) {return x + y;}
}
. . .
var add10 = makeAdder(10);
var add5 = makeAdder(5);
document.write("Add 10 to 20: " + add10(20) +
"<br />");
document.write("Add 5 to 20: " + add5(20) +
"<br />");
```

The output of this code, assuming it was embedded in an HTML document and displayed with a browser, is as follows:

```
Add 10 to 20: 30
Add 5 to 20: 25
```

In this example, the closure is the anonymous function defined inside the `makeAdder` function, which `makeAdder` returns. The variable `x` referenced in the closure function is bound to the parameter that was sent to `makeAdder`. The `makeAdder` function is called twice, once with a parameter of `10` and once with `5`. Each of these calls returns a different version of the closure because they are bound to different values of `x`. The first call to `makeAdder` creates a function that adds `10` to its parameter; the second creates a function that adds `5` to its parameter. The two versions of the function are bound to different activations of `makeAdder`. Obviously, the lifetime of the version of `x` created when `makeAdder` is called must extend over the lifetime of the program.

This same closure function can be written in C# using a nested anonymous delegate. The type of the nesting method is specified to be a function that takes an **`int`** as a parameter and returns an anonymous delegate. The return type is specified with the special notation for such delegates, `Func<int, int>`. The first type in the angle brackets is the parameter type. Such a delegate can encapsulate methods that have only one parameter. The second

type is the return type of the method encapsulated by the delegate.

```
static Func<int, int> makeAdder(int x) {
  return delegate(int y) { return x + y;};
}
. . .
Func<int, int> Add10 = makeAdder(10);
Func<int, int> Add5 = makeAdder(5);
Console.WriteLine("Add 10 to 20: {0}", Add10(20));
Console.WriteLine("Add 5 to 20: {0}", Add5(20));
```

The output of this code is exactly the same as for the previous JavaScript closure example.

The anonymous delegate could have been written as a lambda expression. The following is a replacement for the body of the `makeAdder` method, using a lambda expression instead of the delegate:

```
return y => x + y
```

Ruby’s blocks are implemented so that they can reference variables visible in the position in which they were defined, even if they are called at a place in which those variables would have disappeared. This makes such blocks closures.

9.13 Coroutines

A **coroutine** is a special kind of subprogram. Rather than the master-slave relationship between a caller and a called subprogram that exists with conventional subprograms, caller and called coroutines are more equitable. In fact, the coroutine control mechanism is often called the **symmetric unit control model**.

Coroutines can have multiple entry points, which are controlled by the coroutines themselves. They also have the means to maintain their status between activations. This means that coroutines must be history sensitive and thus have static local variables. Secondary executions of a coroutine often begin at points other than its beginning. Because of this, the invocation of a coroutine is called a **resume** rather than a call.

For example, consider the following skeletal coroutine:

```
sub co1(){
  . . .
  resume  co2();
  . . .
  resume  co3();
  . . .
}
```

The first time `co1` is resumed, its execution begins at the first statement and executes down to and including the resume of `co2`, which transfers control to `co2`. The next time `co1` is resumed, its execution begins at the first statement after its call to `co2`. When `co1` is resumed the third time, its execution begins at the first statement after the resume of `co3`.

One of the usual characteristics of subprograms is maintained in coroutines: Only one coroutine is actually in execution at a given time.

As seen in the example above, rather than executing to its end, a coroutine often partially executes and then transfers control to some other coroutine, and when restarted, a coroutine resumes execution just after the statement it

used to transfer control elsewhere. This sort of interleaved execution sequence is related to the way multiprogramming operating systems work. Although there may be only one processor, all of the executing programs in such a system appear to run concurrently while sharing the processor. In the case of coroutines, this is sometimes called **quasi-concurrency**.

Typically, coroutines are created in an application by a program unit called the master unit, which is not a coroutine. When created, coroutines execute their initialization code and then return control to that master unit. When the entire family of coroutines is constructed, the master program resumes one of the coroutines, and the members of the family of coroutines then resume each other in some order until their work is completed, if in fact it can be completed. If the execution of a coroutine reaches the end of its code section, control is transferred to the master unit that created it. This is the mechanism for ending execution of the collection of coroutines, when that is desirable. In some programs, the coroutines run whenever the computer is running.

One example of a problem that can be solved with this sort of collection of coroutines is a card game simulation. Suppose the game has four players who all use the same strategy. Such a game can be simulated by having a master program unit create a family of four coroutines, each with a collection, or hand, of cards. The master program could then start the simulation by resuming one of the player coroutines, which, after it had played its turn, could resume the next player coroutine, and so forth until the game ended.

Suppose program units `A` and `B` are coroutines. Figure 9.3 shows two ways an execution sequence involving `A` and `B` might proceed.

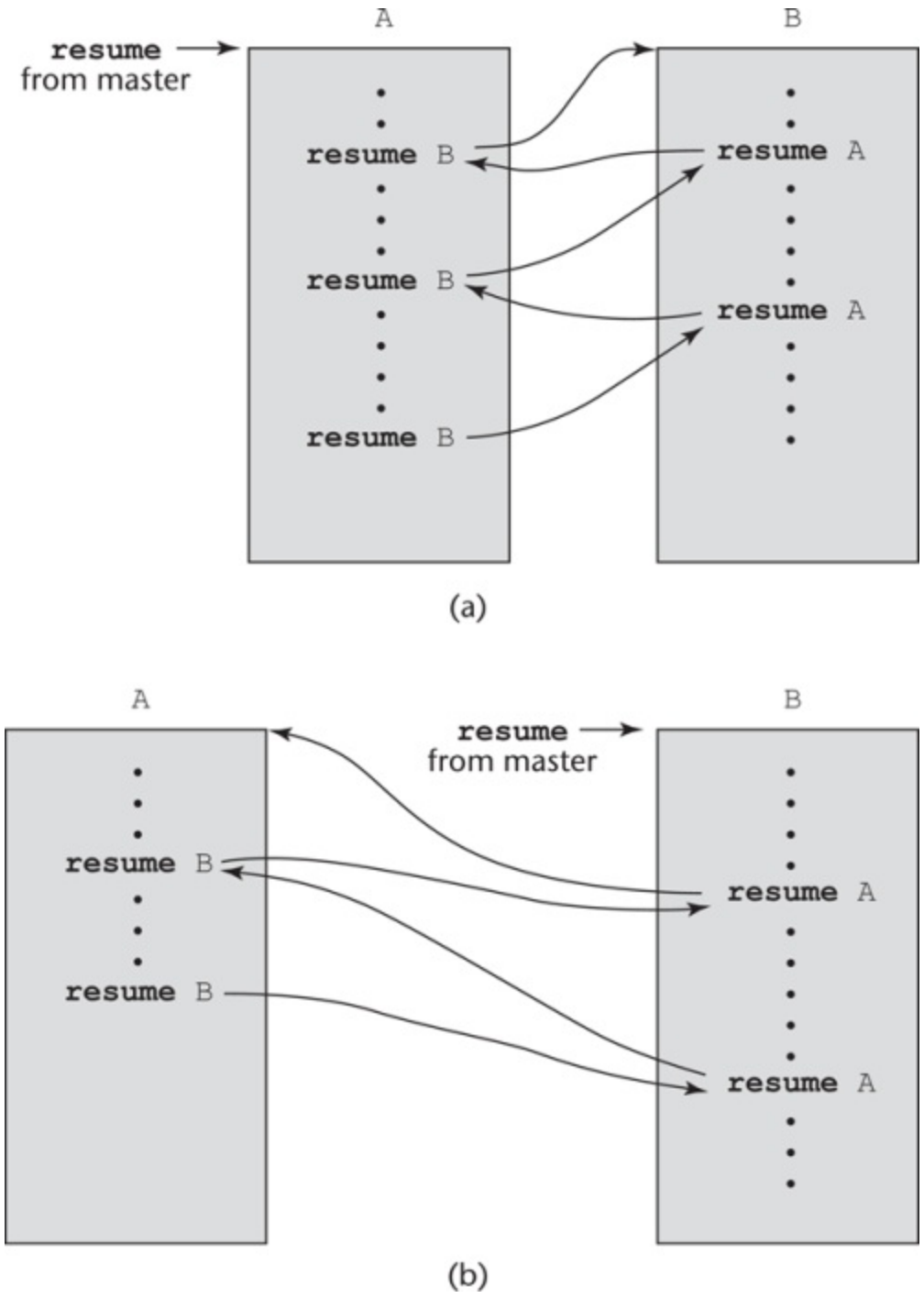


Figure 9.3 Two possible execution control sequences for

two coroutines without loops

Figure 9.3 Full Alternative Text

In Figure 9.3a, the execution of coroutine A is started by the master unit. After some execution, A starts B. When coroutine B in Figure 9.3a first causes control to return to coroutine A, the semantics is that A continues from where it ended its last execution. In particular, its local variables have the values left them by the previous activation. Figure 9.3b shows an alternative execution sequence of coroutines A and B. In this case, B is started by the master unit.

Rather than have the patterns shown in Figure 9.3, a coroutine often has a loop containing a resume. Figure 9.4 shows the execution sequence of this scenario. In this case, A is started by the master unit. Inside its main loop, A resumes B, which in turn resumes A in its main loop.

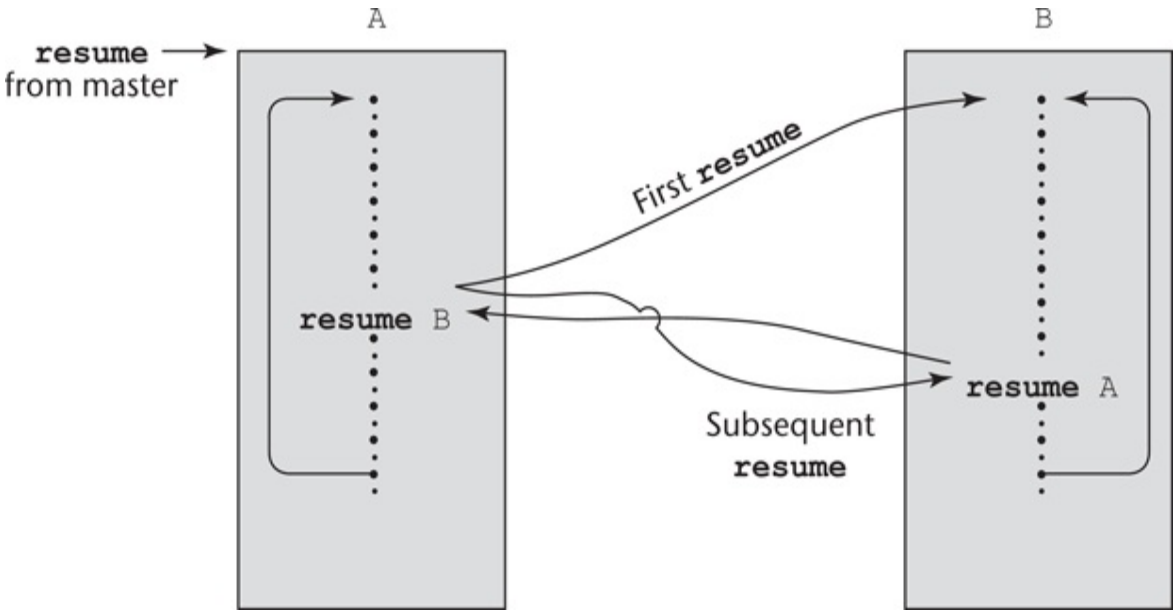


Figure 9.4 Coroutine execution sequence with loops

Figure 9.4 Full Alternative Text

The generators of Python are a form of coroutines.13

13. However, the generators of Python are a form of coroutines.

SUMMARY

Process abstractions are represented in programming languages by subprograms. A subprogram definition describes the actions represented by the subprogram. A subprogram call enacts those actions. A subprogram header identifies a subprogram definition and provides its interface, which is called its protocol.

Formal parameters are the names that subprograms use to refer to the actual parameters given in subprogram calls. In Python and Ruby, array and hash formal parameters are used to support variable numbers of parameters. JavaScript also supports variable numbers of parameters. Actual parameters can be associated with formal parameters by position or by keyword. Parameters can have default values.

Subprograms can be either functions, which model mathematical functions and are used to define new operations, or procedures, which define new statements.

Local variables in subprograms can be stack dynamic, providing support for recursion, or static, providing efficiency and history-sensitive local variables.

JavaScript, Python, Ruby, and Swift allow subprogram definitions to be nested.

There are three fundamental semantics models of parameter passing—in mode, out mode, and inout mode—and a number of approaches to implement them. These are pass-by-value, pass-by-result, pass-by-value-result, pass-by-reference, and pass-by-name. In most languages, parameters are passed in the run-time stack.

Aliasing can occur when pass-by-reference parameters are used, both among two or more parameters and between a parameter and an accessible nonlocal variable.

Parameters that are multidimensioned arrays pose some issues for the

language designer, because the called subprogram needs to know how to compute the storage mapping function for them. This requires more than just the name of the array.

Parameters that are subprogram names provide a necessary service but can be difficult to understand. The opacity lies in the referencing environment that is available when a subprogram that has been passed as a parameter is executed.

C and C++ support pointers to functions. C# has delegates, which are objects that can store references to methods. Delegates can support multicast calls by storing more than one method reference.

Ada, C++, C#, Ruby, and Python allow both subprogram and operator overloading. Subprograms can be overloaded as long as the various versions can be disambiguated by the types of their parameters or returned values. Function definitions can be used to build additional meanings for operators.

Subprograms in C++, Java 5.0, and C# 2005 can be generic, using parametric polymorphism, so the desired types of their data objects can be passed to the compiler, which then can construct units for the requested types.

The designer of a function facility in a language must decide what restrictions will be placed on the returned values, as well as the number of return values.

A closure is a subprogram and its referencing environment. Closures are useful in languages that allow nested subprograms, are static-scoped, and allow subprograms to be returned from functions and assigned to variables.

A coroutine is a special subprogram that has multiple entries. It can be used to provide interleaved execution of subprograms.

REVIEW QUESTIONS

1. What are the three general characteristics of subprograms?
2. What does it mean for a subprogram to be active?
3. What is given in the header of a subprogram?
4. What characteristic of Python subprograms sets them apart from those of other languages?
5. What languages allow a variable number of parameters?
6. What is a Ruby array formal parameter?
7. What is a parameter profile? What is a subprogram protocol?
8. What are formal parameters? What are actual parameters?
9. What are the advantages and disadvantages of keyword parameters?
10. What are the differences between a function and a procedure?
11. What are the design issues for subprograms?
12. What are the advantages and disadvantages of dynamic local variables?
13. What are the advantages and disadvantages of static local variables?
14. What languages allow subprogram definitions to be nested?
15. What are the three semantics models of parameter passing?
16. What are the modes, the conceptual models of transfer, the advantages, and the disadvantages of pass-by-value, pass-by-result, pass-by-value-result, and pass-by-reference parameter-passing methods?

17. Describe the ways that aliases can occur with pass-by-reference parameters.

18. What is the difference between the way original C and C89 deal with an actual parameter whose type is not identical to that of the corresponding formal parameter?

19. What are two fundamental design considerations for parameter-passing methods?

20. Describe the problem of passing multidimensioned arrays as parameters.

21. What is the name of the parameter-passing method used in Ruby?

22. What are the two issues that arise when subprogram names are parameters?

23. Define *shallow* and *deep binding* for referencing environments of subprograms that have been passed as parameters.

24. What is an overloaded subprogram?

25. What is parametric polymorphism?

26. What causes a C++ template function to be instantiated?

27. In what fundamental ways do the generic parameters to a Java 5.0 generic method differ from those of C++ methods?

28. If a Java 5.0 method returns a generic type, what type of object is actually returned?

29. If a Java 5.0 generic method is called with three different generic parameters, how many versions of the method will be generated by the compiler?

30. What are the design issues for functions?

31. Name two languages that allow multiple values to be returned from a

function.

32. What exactly is a delegate?
33. What is the main drawback of generic functions in F#?
34. What is a closure?
35. What are the language characteristics that make closures useful?
36. What languages allow the user to overload operators?
37. In what ways are coroutines different from conventional subprograms?

PROBLEM SET

1. What are arguments for and against a user program building additional definitions for existing operators, as can be done in Python and C++? Do you think such user-defined operator overloading is good or bad? Support your answer.

2. In most Fortran IV implementations, all parameters were passed by reference, using access path transmission only. State both the advantages and disadvantages of this design choice.

3. Argue in support of the Ada 83 designers' decision to allow the implementor to choose between implementing **`inout-`**mode parameters by copy or by reference.

4. Suppose you want to write a method that prints a heading on a new output page, along with a page number that is 1 in the first activation and that increases by 1 with each subsequent activation. Can this be done without parameters and without reference to nonlocal variables in Java? Can it be done in C#?

5. Consider the following program written in C syntax:

```
void swap(int a, int b) {
  int temp;
  temp = a;
  a = b;
  b = temp;
}
void main() {
  int value = 2, list[5] = {1, 3, 5, 7, 9};
  swap(value, list[0]);
  swap(list[0], list[1]);
  swap(value, list[value]);
}
```

   For each of the following parameter-passing methods, what are all of the values of the variables `value` and `list` after each of the three calls to

`swap`?

   1. Passed by value

   2. Passed by reference

   3. Passed by value-result

6. Present one argument against providing both static and dynamic local variables in subprograms.

7. Consider the following program written in C syntax:

```
void fun (int first, int second) {
  first += first;
  second += second;
}
void main() {
  int list[2] = {1, 3};
  fun(list[0], list[1]);
}
```

   For each of the following parameter-passing methods, what are the values of the `list` array after execution?

   1. Passed by value

   2. Passed by reference

   3. Passed by value-result

8. Argue against the C design of providing only function subprograms.

9. From a textbook on Fortran, learn the syntax and semantics of statement functions. Justify their existence in Fortran.

10. Study the methods of user-defined operator overloading in C++ and Ada, and write a report comparing the two using our criteria for evaluating languages.

11. C# supports out-mode parameters, but neither Java nor C++ does.

Explain the difference.

12. Research Jensen's Device, which was a use of pass-by-name parameters, and write a short description of what it is and how it can be used.

13. Study the iterator mechanisms of Ruby and CLU and list their similarities and differences.

14. Speculate on the issue of allowing nested subprograms in programming languages—why are they not allowed in many contemporary languages?

15. What are at least two arguments against the use of pass-by-name parameters?

16. Write a detailed comparison of the generic subprograms of Java 5.0 and C# 2005.

PROGRAMMING EXERCISES

1. Write a program in a language that you know to determine the ratio of the time required to pass a large array by reference and the time required to pass the same array by value. Make the array as large as possible on the machine and implementation you use. Pass the array as many times as necessary to get reasonably accurate timings of the passing operations.

2. Write a C# or Ada program that determines when the address of an out-mode parameter is computed (at the time of the call or at the time the execution of the subprogram finishes).

3. Write a Perl program that passes by reference a literal to a subprogram, which attempts to change the parameter. Given the overall design philosophy of Perl, explain the results.

4. Repeat Programming Exercise 3 in C#.

5. Write a program in some language that has both static and stack-dynamic local variables in subprograms. Create six large (at least 100×100) matrices in the subprogram—three static and three stack dynamic. Fill two of the static matrices and two of the stack-dynamic matrices with random numbers in the range of 1 to 100. The code in the subprogram must perform a large number of matrix multiplication operations on the static matrices and time the process. Then it must repeat this with the stack-dynamic matrices. Compare and explain the results.

6. Write a C# program that includes two methods that are called a large number of times. Both methods are passed a large array, one by value and one by reference. Compare the times required to call these two methods and explain the difference. Be sure to call them a sufficient number of times to illustrate a difference in the required time.

7. Write a program, using the syntax of whatever language you like, that produces different behavior depending on whether pass-by-reference or pass-by-value-result is used in its parameter passing.

8. Write a generic C++ function that takes an array of generic elements and a scalar of the same type as the array elements. The type of the array elements and the scalar is the generic parameter. The function must search the given array for the given scalar and return the subscript of the scalar in the array. If the scalar is not in the array, the function must return `-1`. Test the function for `int` and `float` types.

9. Devise a subprogram and calling code in which pass-by-reference and pass-by-value-result of one or more parameters produces different results.

10 Implementing Subprograms

The purpose of this chapter is to explore the implementation of subprograms. The discussion will provide the reader with some knowledge of how subprogram linkage works, and also why ALGOL 60 was a challenge to the unsuspecting compiler writers of the early 1960s. We begin with the simplest situation, nonnestable subprograms with static local variables, advance to more complicated subprograms with stack-dynamic local variables, and conclude with nested subprograms with stack-dynamic local variables and static scoping. The increased difficulty of implementing subprograms in languages with nested subprograms is caused by the need to include mechanisms to access nonlocal variables.

The static chain method of accessing nonlocals in static-scoped languages is discussed in detail. Then, techniques for implementing blocks are described. Finally, several methods of implementing nonlocal variable access in a dynamic-scoped language are discussed.

10.1 The General Semantics of Calls and Returns

The subprogram call and return operations are together called **subprogram linkage**. The implementation of subprograms must be based on the semantics of the subprogram linkage of the language being implemented.

A subprogram call in a typical language has numerous actions associated with it. The call process must include the implementation of whatever -parameter-passing method is used. If local variables are not static, the call process must allocate storage for the locals declared in the called subprogram and bind those variables to that storage. It must save the execution status of the calling program unit. The execution status is everything needed to resume execution of the calling program unit. This includes register values, CPU status bits, and the environment pointer (EP). The EP, which is discussed further in Section 10.3, is used to access parameters and local variables during the execution of a subprogram. The calling process also must arrange to transfer control to the code of the subprogram and ensure that control can return to the proper place when the subprogram execution is completed. Finally, if the language supports nested subprograms, the call process must create some mechanism to provide access to nonlocal variables that are visible to the called subprogram.

The required actions of a subprogram return are less complicated than those of a call. If the subprogram has parameters that are out mode or inout mode and are implemented by copy, the first action of the return process is to move the local values of the associated formal parameters to the actual parameters. Next, it must deallocate the storage used for local variables and restore the execution status of the calling program unit. Finally, control must be returned to the calling program unit.

10.2 Implementing “Simple” Subprograms

We begin with the task of implementing simple subprograms. By “simple” we mean that subprograms cannot be nested and all local variables are static. Early versions of Fortran were examples of languages that had this kind of subprograms.

The semantics of a call to a “simple” subprogram requires the following actions:

1. Save the execution status of the current program unit.
2. Compute and pass the parameters.
3. Pass the return address to the called.
4. Transfer control to the called.

The semantics of a return from a simple subprogram requires the following actions:

1. If there are pass-by-value-result or out-mode parameters, the current values of those parameters are moved to or made available to the corresponding actual parameters.
2. If the subprogram is a function, the functional value is moved to a place accessible to the caller.
3. The execution status of the caller is restored.
4. Control is transferred back to the caller.

The call and return actions require storage for the following:

- Status information about the caller
- Parameters
- Return address
- Return value for functions
- Temporaries used by the code of the subprograms

These, along with the local variables and the subprogram code, form the complete collection of information a subprogram needs to execute and then return control to the caller.

The question now is the distribution of the call and return actions to the caller and the called. For simple subprograms, the answer is obvious for most of the parts of the process. The last three actions of a call clearly must be done by the caller. Saving the execution status of the caller could be done by either. In the case of the return, the first, third, and fourth actions must be done by the called. Once again, the restoration of the execution status of the caller could be done by either the caller or the called. In general, the linkage actions of the called can occur at two different times, either at the beginning of its execution or at the end. These are sometimes called the **prologue** and **epilogue** of the subprogram linkage. In the case of a simple subprogram, all of the linkage actions of the callee occur at the end of its execution, so there is no need for a prologue.

A simple subprogram consists of two separate parts: the actual code of the subprogram, which is constant, and the local variables and data listed previously, which can change when the subprogram is executed. In the case of simple subprograms, both of these parts have fixed sizes.

The format, or layout, of the noncode part of a subprogram is called an **activation record**, because the data it describes are relevant only during the activation or execution of the subprogram. The form of an activation record is static. An **activation record instance** is a concrete example of an activation record, a collection of data in the form of an activation record.

Because languages with simple subprograms do not support recursion, there can be only one active version of a given subprogram at a time. Therefore, there can be only a single instance of the activation record for a subprogram. One possible layout for activation records is shown in Figure 10.1. The saved execution status of the caller is omitted here and in the remainder of this chapter because it is simple and not relevant to the discussion.

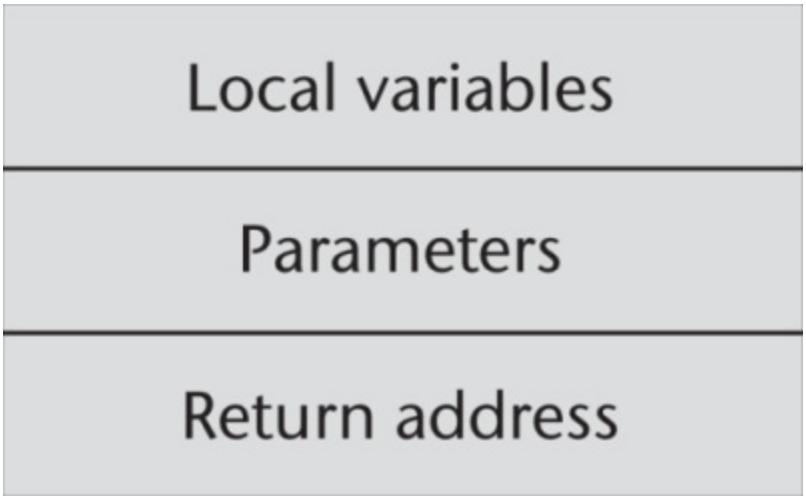


Figure 10.1 An activation record for simple subprogram

Because an activation record instance for a “simple” subprogram has fixed size, it can be statically allocated. In fact, it could be attached to the code part of the subprogram.

Figure 10.2 shows a program consisting of a main program and three subprograms: A, B, and C. Although the figure shows all the code segments separated from all the activation record instances, in some cases, the activation record instances are attached to their associated code segments.

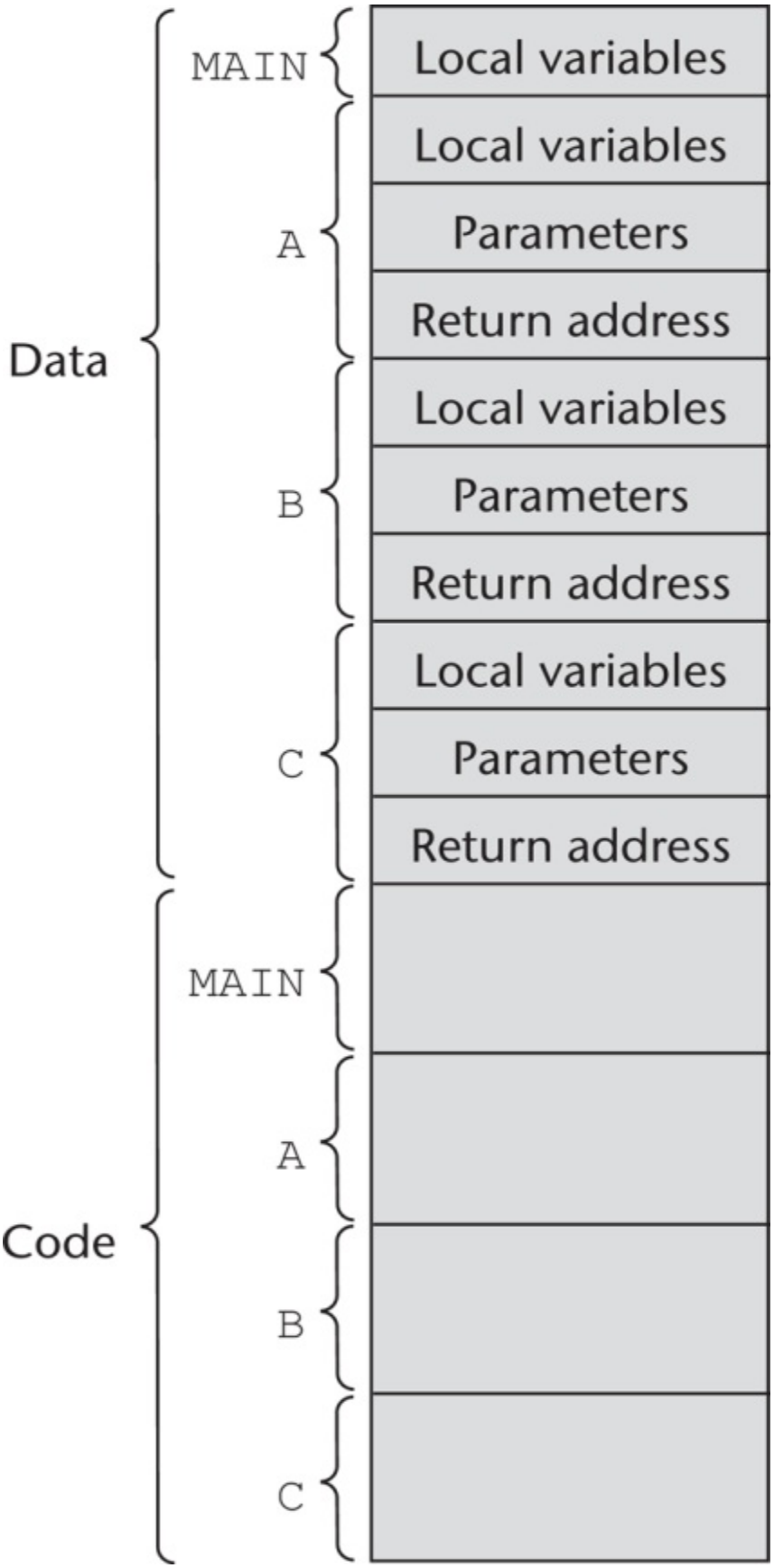

Data
Code
MAIN
A
B
C
MAIN
A
B
C
Local variables
Local variables
Parameters
Return address
Local variables
Parameters
Return address
Local variables
Parameters
Return address

Figure 10.2 The code and - activation records of a program with simple subprograms

Figure 10.2 Full Alternative Text

The construction of the complete program shown in Figure 10.2 is not done entirely by the compiler. In fact, if the language allows independent compilation, the four program units—`MAIN`, `A`, `B`, and `C`—may have been compiled on different days, or even in different years. At the time each unit is compiled, the machine code for it, along with a list of references to external subprograms, is written to a file. The executable program shown in Figure 10.2 is put together by the **linker**, which is part of the operating system. (Sometimes linkers are called *loaders*, *linker/loaders*, or *link editors*.) When the linker is called for a main program, its first task is to find the files that contain the translated subprograms referenced in that program and load them into memory. Then, the linker must set the target addresses of all calls to those subprograms in the main program to the entry addresses of those subprograms. The same must be done for all calls to subprograms in the loaded subprograms and all calls to library subprograms. In the previous example, the linker was called for `MAIN`. The linker had to find the machine code programs for `A`, `B`, and `C`, along with their activation record instances, and load them into memory with the code for `MAIN`. Then, it had to patch in the target addresses for all calls to `A`, `B`, `C`, and any library subprograms called in `A`, `B`, `C`, and `MAIN`.

10.3 Implementing Subprograms with Stack-Dynamic Local Variables

We now examine the implementation of the subprogram linkage in languages in which locals are stack dynamic, again focusing on the call and return operations.

One of the most important advantages of stack-dynamic local variables is support for recursion. Therefore, languages that use stack-dynamic local variables also support recursion.

A discussion of the additional complexity required when subprograms can be nested is postponed until Section 10.4.

10.3.1 More Complex Activation Records

Subprogram linkage in languages that use stack-dynamic local variables are more complex than the linkage of simple subprograms for the following reasons:

- The compiler must generate code to cause the implicit allocation and deallocation of local variables.
- Recursion adds the possibility of multiple simultaneous activations of a subprogram, which means that there can be more than one instance (incomplete execution) of a subprogram at a given time, with at least one call from outside the subprogram and one or more recursive calls. The number of activations is limited only by the memory size of the

machine. Each activation requires its own activation record instance.

The format of an activation record for a given subprogram in most languages is known at compile time. In many cases, the size is also known for activation records because all local data are of a fixed size. That is not the case in some other languages, such as Ada, in which the size of a local array can depend on the value of an actual parameter. In those cases, the format is static, but the size can be dynamic. In languages with stack-dynamic local variables, activation record instances must be created dynamically. The typical activation record for such a language is shown in Figure 10.3.

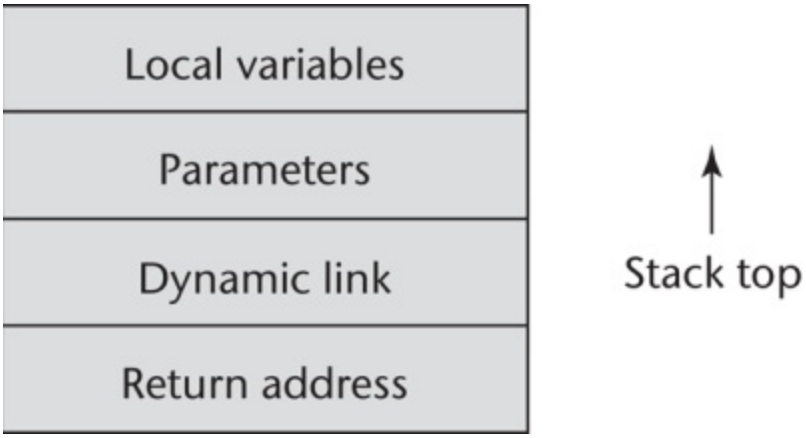


Figure 10.3 A typical activation record for a language with stack-dynamic local variables

Because the return address, dynamic link, and parameters are placed in the activation record instance by the caller, these entries must appear first.

The return address usually consists of a pointer to the instruction following the call in the code segment of the calling program unit. The **dynamic link** is a pointer to the base of the activation record instance of the caller. In static-scoped languages, this link is used to provide traceback information when a run-time error occurs. In dynamic-scoped languages, the dynamic link is used to access nonlocal variables. The actual parameters in the activation record

are the values or addresses provided by the caller.

Local scalar variables are bound to storage within an activation record instance. Local variables that are structures are sometimes allocated elsewhere, and only their descriptors and a pointer to that storage are part of the activation record. Local variables are allocated and possibly initialized in the called subprogram, so they appear last.

Consider the following skeletal C function:

```
void sub(float total, int part) {
  int list[5];
  float sum;
  . . .
}
```

The activation record for `sub` is shown in Figure 10.4.

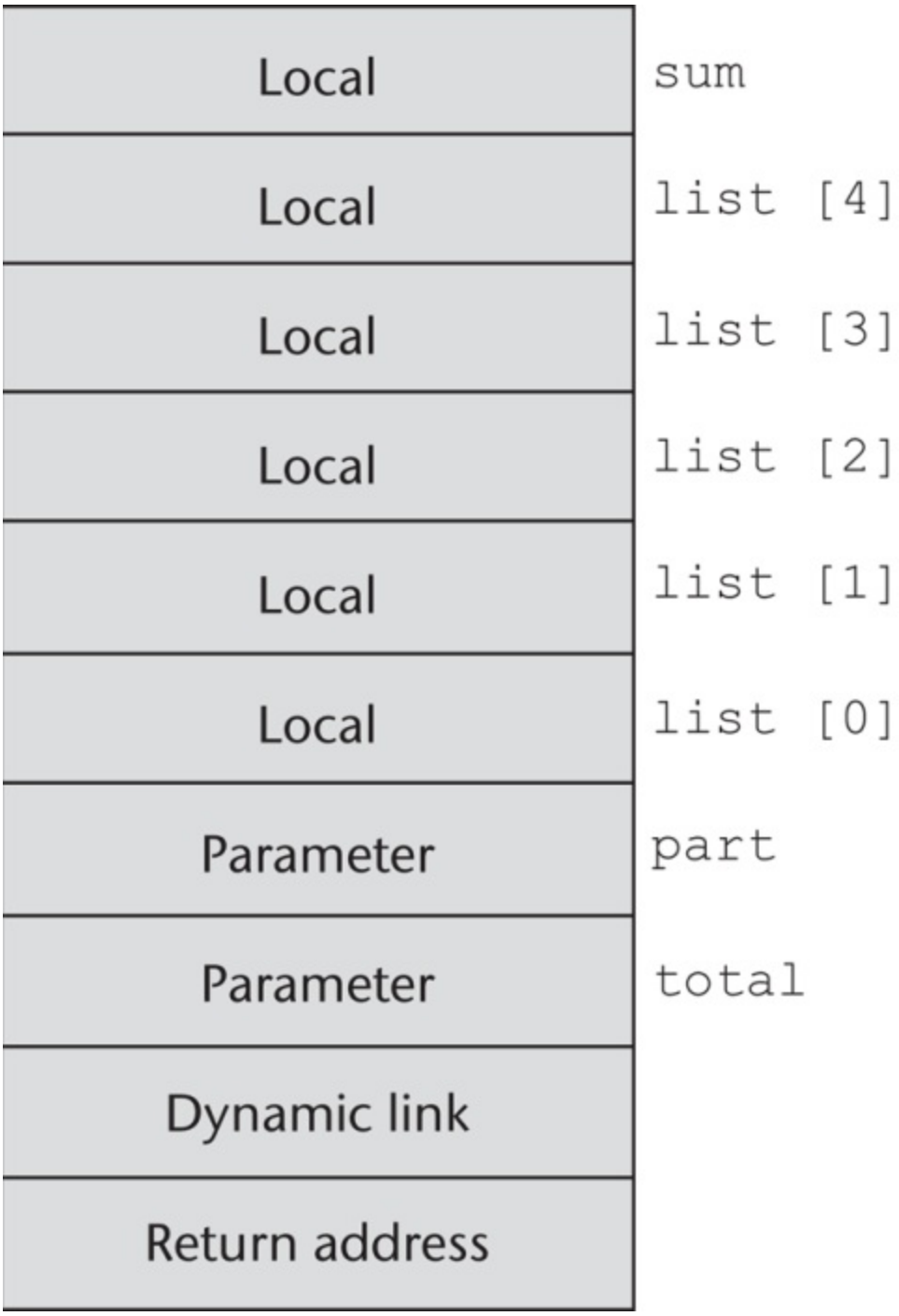


Figure 10.4 The activation record for function sub

Figure 10.4 Full Alternative Text

Activating a subprogram requires the dynamic creation of an instance of the activation record for the subprogram. As stated earlier, the format of the activation record is fixed at compile time, although its size may depend on the call in some languages. Because the call and return semantics specify that the subprogram last called is the first to complete, it is reasonable to create instances of these activation records on a stack. This stack is part of the run-time system and therefore is called the **run-time stack**, although we will usually just refer to it as the stack. Every subprogram activation, whether recursive or nonrecursive, creates a new instance of an activation record on the stack. This provides the required separate copies of the parameters, local variables, and return address.

One more thing is required to control the execution of a subprogram—the EP. Initially, the EP points at the base, or first address of the activation record instance of the main program. The run-time system must ensure that it always points at the base of the activation record instance of the currently executing program unit. When a subprogram is called, the current EP is saved in the new activation record instance as the dynamic link. The EP is then set to point at the base of the new activation record instance. Upon return from the subprogram, the stack top is set to the value of the current EP minus one and the EP is set to the dynamic link from the activation record instance of the subprogram that has completed its execution. Resetting the stack top effectively removes the top activation record instance.

The EP is used as the base of the offset addressing of the data contents of the activation record instance—parameters and local variables.

Note that the EP currently being used is not stored in the run-time stack. Only saved versions are stored in the activation record instances as the dynamic links.

We have now discussed several new actions in the linkage process. The lists given in Section 10.2 must be revised to take these into account. Using the activation record form given in this section, the new actions are as follows:

The caller actions are as follows:

1. Create an activation record instance.
2. Save the execution status of the current program unit.
3. Compute and pass the parameters.
4. Pass the return address to the called.
5. Transfer control to the called.

The prologue actions of the called are as follows:

1. Save the old EP in the stack as the dynamic link and create the new value.
2. Allocate local variables.

The epilogue actions of the called are as follows:

1. If there are pass-by-value-result or out-mode parameters, the current values of those parameters are moved to the corresponding actual parameters.
2. If the subprogram is a function, the functional value is moved to a place accessible to the caller.
3. Restore the stack pointer by setting it to the value of the current EP minus one and set the EP to the old dynamic link.
4. Restore the execution status of the caller.
5. Transfer control back to the caller.

Recall from Chapter 9, that a subprogram is **active** from the time it is called until the time that execution is completed. At the time it becomes inactive, its local scope ceases to exist and its referencing environment is no longer meaningful. Therefore, at that time, its activation record instance can be destroyed.

Parameters are not always transferred in the stack. In many compilers for RISC machines, parameters are passed in registers. This is because RISC machines normally have many more registers than CISC machines. In the remainder of this chapter, however, we assume that parameters are passed in the stack. It is straightforward to modify this approach for parameters being passed in registers.

10.3.2 An Example without Recursion

Consider the following skeletal C program:

```
void fun1(float r) {
  int s, t;
  . . .    <---------- 1
  fun2(s);
  . . .
}
void fun2(int x) {
  int y;
  . . .    <---------- 2
  fun3(y);
  . .
}
void fun3(int q) {
  . . .    <---------- 3
}
void main() {
  float p;
  . . .
  fun1(p);
  . . .
}
```

The sequence of function calls in this program is

```
main calls fun1
fun1 calls fun2
fun2 calls fun3
```

The stack contents for the points labeled 1, 2, and 3 are shown in Figure 10.5.

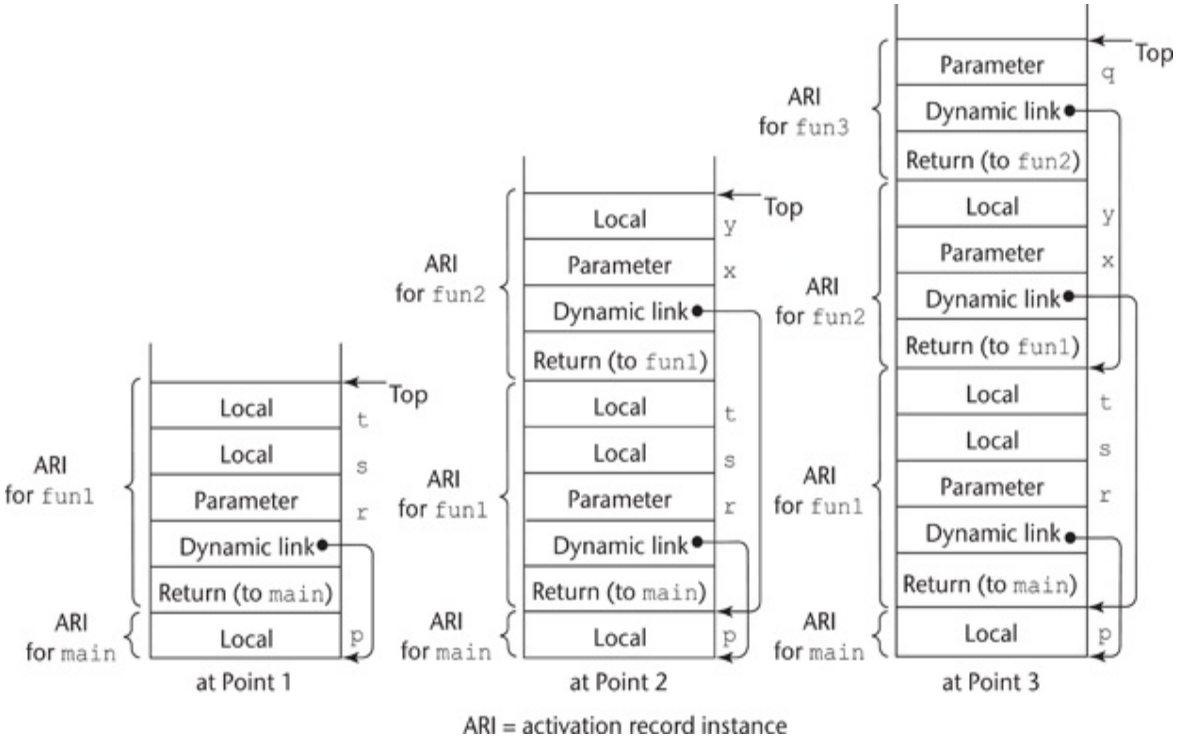


Figure 10.5 Stack contents for three points in a program

Figure 10.5 Full Alternative Text

At point 1, only the activation record instances for function `main` and function `fun1` are on the stack. When `fun1` calls `fun2`, an instance of `fun2`'s activation record is created on the stack. When `fun2` calls `fun3`, an instance of `fun3`'s activation record is created on the stack. When `fun3`'s execution ends, the instance of its activation record is removed from the stack, and the EP is used to reset the stack top pointer. Similar processes take place when functions `fun2` and `fun1` terminate. After the return from the call to `fun1` from `main`, the stack has only the instance of the activation record of `main`. Note that some implementations do not actually use an activation record instance on the stack for main functions, such as the one shown in the figure. However, it can be done this way, and it simplifies both the implementation and our discussion.

In this example and in all others in this chapter, we assume that the stack grows from lower addresses to higher addresses, although in a particular implementation, the stack may grow in the opposite direction.

The collection of dynamic links present in the stack at a given time is called the **dynamic chain**, or **call chain**. It represents the dynamic history of how execution got to its current position, which is always in the subprogram code whose activation record instance is on top of the stack. References to local variables can be represented in the code as offsets from the beginning of the activation record of the local scope, whose address is stored in the EP. Such an offset is called a **local_offset**.

The local_offset of a variable in an activation record can be determined at compile time, using the order, types, and sizes of variables declared in the subprogram associated with the activation record. To simplify the discussion, we assume that all variables take one position in the activation record. The first local variable declared in a subprogram would be allocated in the activation record two positions plus the number of parameters from the bottom (the first two positions are for the return address and the dynamic link). The second local variable declared would be one position nearer the stack top and so forth. For example, consider the preceding example program. In `fun1`, the local_offset of `s` is 3; for `t` it is 4. Likewise, in `fun2`, the local_offset of `y` is 3. To get the address of any local variable, the local_offset of the variable is added to the EP.

10.3.3 Recursion

Consider the following example C program, which uses recursion to compute the factorial function:

```
int factorial(int n) {
    <---------- 1
  if (n <= 1)
    return 1;
  else return (n * factorial(n - 1));
    <---------- 2
 }
void main() {
```

```
  int value;
  value = factorial(3);
    <---------- 3
}
```

The activation record format for the function `factorial` is shown in Figure 10.6. Notice that it has an additional entry for the return value of the function.

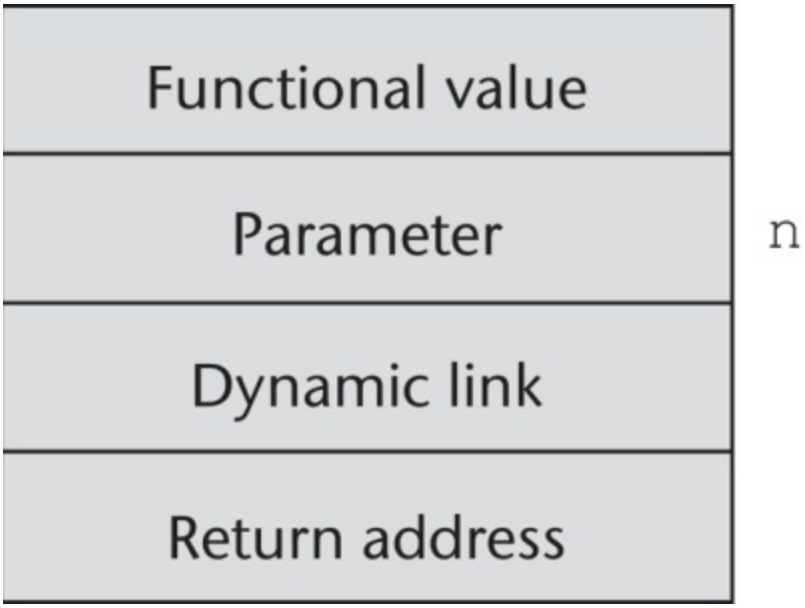


Figure 10.6 The activation record for `factorial`

Figure 10.7 shows the contents of the stack for the three times execution reaches position 1 in the function `factorial`. Each shows one more activation of the function, with its functional value undefined. The first activation record instance has the return address to the calling function, `main`. The others have a return address to the function itself; these are for the recursive calls.

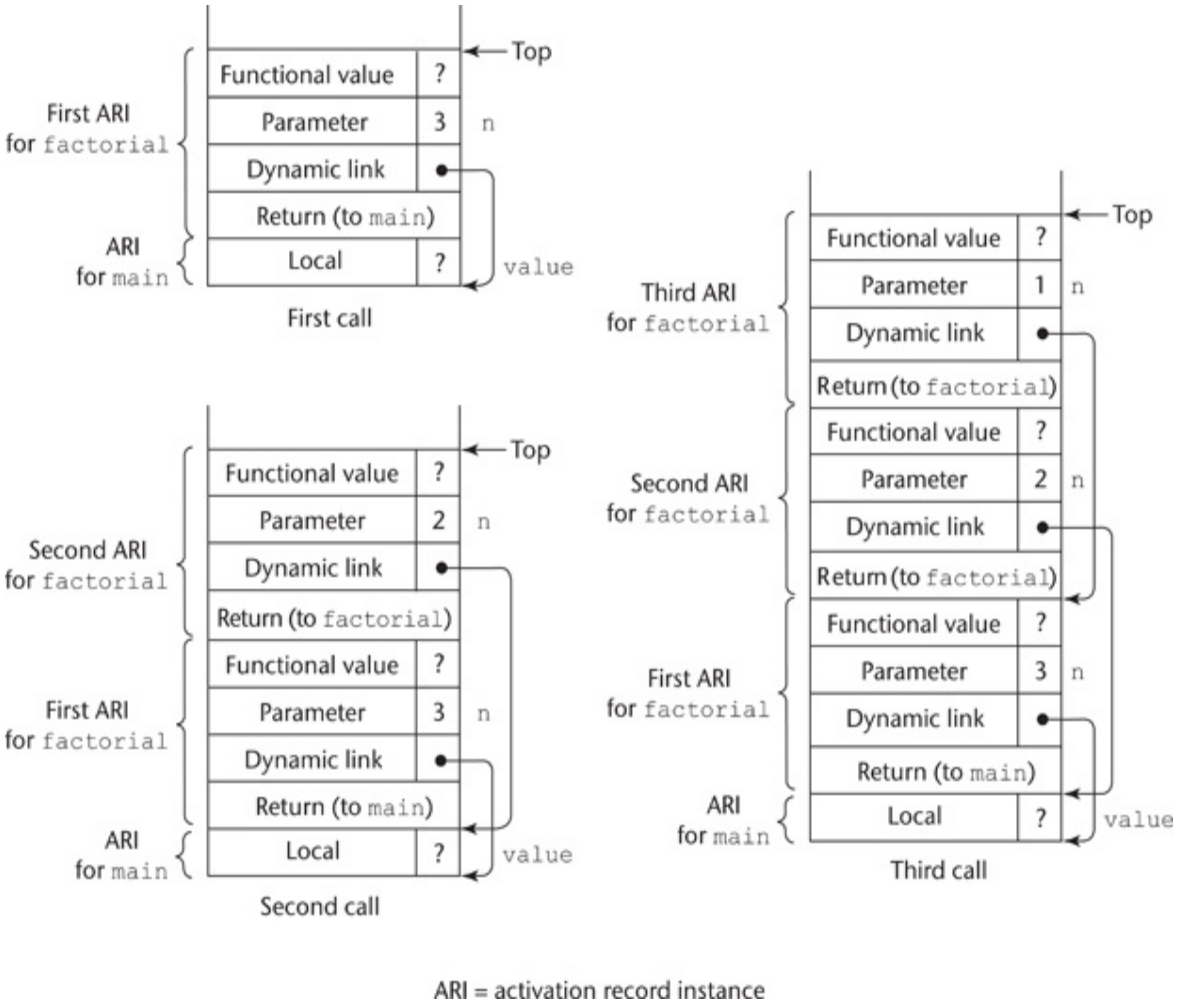


Figure 10.7 Stack contents at position 1 in `factorial`

Figure 10.7 Full Alternative Text

Figure 10.8 shows the stack contents for the three times that execution reaches position 2 in the function `factorial`. Position 2 is meant to be the time after the **`return`** is executed but before the activation record has been removed from the stack. Recall that the code for the function multiplies the current value of the parameter `n` by the value returned by the recursive call to the function. The first return from `factorial` returns the value 1. The activation record instance for that activation has a value of 1 for its version of

the parameter `n`. The result from that multiplication, 1, is returned to the second activation of `factorial` to be multiplied by its parameter value for `n`, which is 2. This step returns the value 2 to the first activation of `factorial` to be multiplied by its parameter value for `n`, which is 3, yielding the final functional value of 6, which is then returned to the first call to `factorial` in `main`.

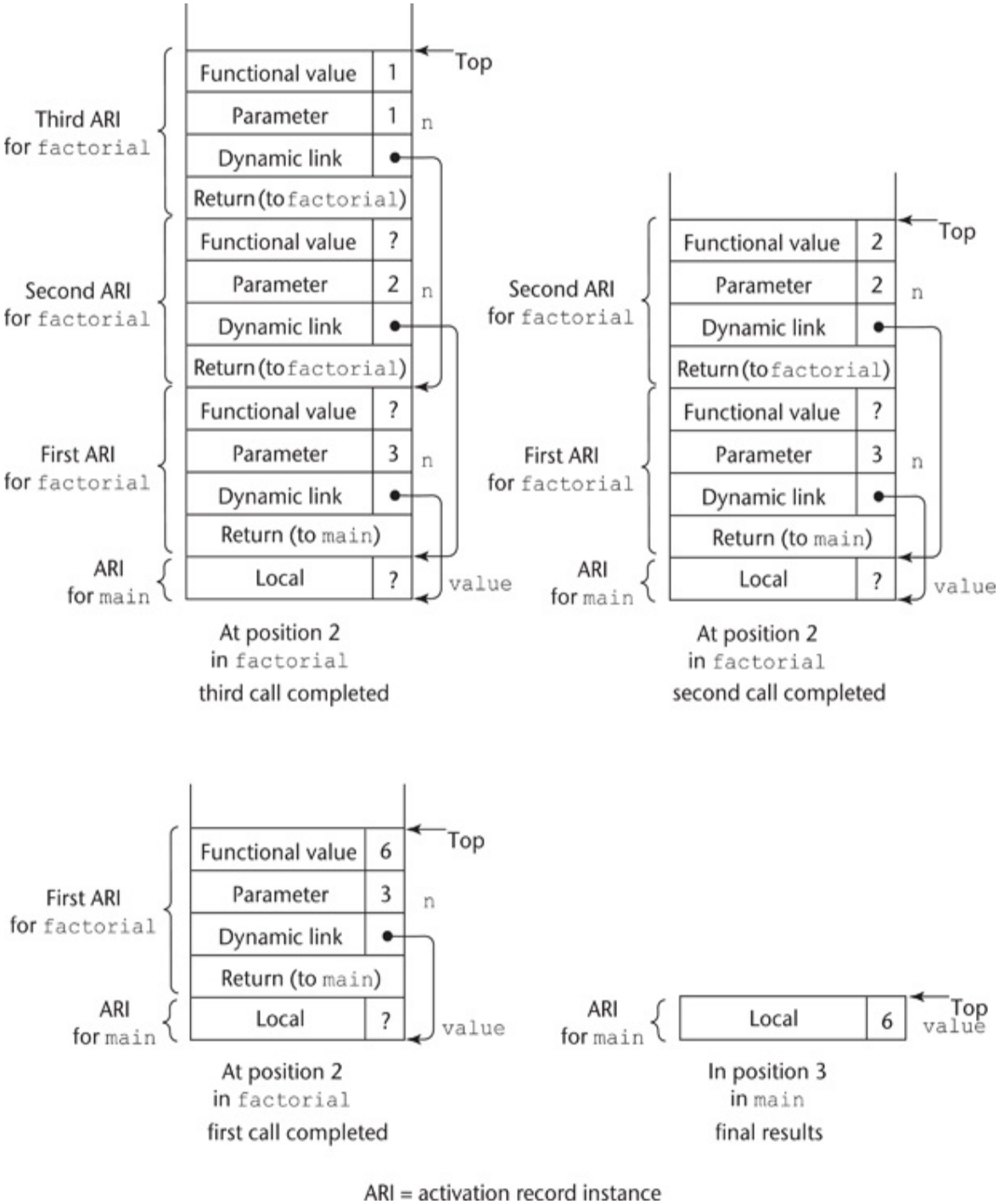


Figure 10.8 Stack contents during execution of `main` and `factorial`

Figure 10.8 Full Alternative Text

10.4 Nested Subprograms

Some of the non–C-based static-scoped programming languages use stack--dynamic local variables and allow subprograms to be nested. Among these are Fortran 95+, Ada, Python, JavaScript, Ruby, and Swift, as well as the functional languages. In this section, we examine the most commonly used approach to implementing subprograms that may be nested. Until the very end of this section, we ignore closures.

10.4.1 The Basics

A reference to a nonlocal variable in a static-scoped language with nested subprograms requires a two-step access process. All nonstatic variables that can be nonlocally accessed are in existing activation record instances and therefore are somewhere in the stack. The first step of the access process is to find the instance of the activation record in the stack in which the variable was allocated. The second part is to use the local_offset of the variable (within the activation record instance) to access it.

Finding the correct activation record instance is the more interesting and more difficult of the two steps. First, note that in a given subprogram, only variables that are declared in static ancestor scopes are visible and can be accessed. Also, activation record instances of all of the static ancestors are always on the stack when variables in them are referenced by a nested subprogram. This is guaranteed by the static semantic rules of the static-scoped languages: A subprogram is callable only when all of its static ancestor subprograms are active.1 If a particular static ancestor were not active, its local variables would not be bound to storage, so it would be nonsense to allow access to them.

1. Closures, of course, violate this rule.

The semantics of nonlocal references dictates that the correct declaration is the first one found when looking through the enclosing scopes, most closely

nested first. So, to support nonlocal references, it must be possible to find all of the instances of activation records in the stack that correspond to those static ancestors. This observation leads to the implementation approach described in the following subsection.

We do not address the issue of blocks until Section 10.5, so in the remainder of this section, all scopes are assumed to be defined by subprograms. Because functions cannot be nested in the C-based languages (the only static scope in those languages are those created with blocks), the discussions of this section do not apply to those languages directly.

10.4.2 Static Chains

The most common way to implement static scoping in languages that allow nested subprograms is static chaining. In this approach, a new pointer, called a static link, is added to the activation record. The **static link**, which is sometimes called a *static scope pointer*, points to the bottom of the activation record instance of an activation of the static parent. It is used for accesses to nonlocal variables. Typically, the static link appears in the activation record below the parameters. The addition of the static link to the activation record requires that local offsets differ from when the static link is not included. Instead of having two activation record elements before the parameters, there are now three: the return address, the static link, and the dynamic link.

A **static chain** is a chain of static links that connect certain activation record instances in the stack. During the execution of a subprogram `P`, the static link of its activation record instance points to an activation record instance of `P`'s static parent program unit. That instance's static link points in turn to `P`'s static grandparent program unit's activation record instance, if there is one. So, the static chain connects all the static ancestors of an executing subprogram, in order of static parent first. This chain can obviously be used to implement the accesses to nonlocal variables in static-scoped languages.

Finding the correct activation record instance of a nonlocal variable using static links is relatively straightforward. When a reference is made to a nonlocal variable, the activation record instance containing the variable can

be found by searching the static chain until a static ancestor activation record instance is found that contains the variable. However, it can be much easier than that. Because the nesting of scopes is known at compile time, the compiler can determine not only that a reference is nonlocal but also the length of the static chain that must be followed to reach the activation record instance that contains the nonlocal object.

Let **static_depth** be an integer associated with a static scope that indicates how deeply it is nested in the outermost scope. A program unit that is not nested inside any other unit has a static_depth of 0. If subprogram `A` is defined in a nonnested program unit, its static_depth is 1. If subprogram `A` contains the definition of a nested subprogram `B`, then `B`'s static_depth is 2.

The length of the static chain needed to reach the correct activation record instance for a nonlocal reference to a variable `X` is exactly the difference between the static_depth of the subprogram containing the reference to `X` and the static_depth of the subprogram containing the declaration for `X`. This difference is called the **nesting_depth**, or **chain_offset**, of the reference. The actual reference can be represented by an ordered pair of integers (chain_offset, local_offset), where chain_offset is the number of links to the correct activation record instance (local_offset is described in Section 10.3.2). For example, consider the following skeletal Python program:

```
# Global scope
. . .
def f1():
  def f2():
    def f3():
      . . .
    # end of f3
    . . .
  # end of f2
  . .
# end of f1
```

The static_depths of the global scope, `f1`, `f2`, and `f3` are 0, 1, 2, and 3, respectively. If procedure `f3` references a variable declared in `f1`, the chain_offset of that reference would be 2 (static_depth of `f3` minus the static_depth of `f1`). If procedure `f3` references a variable declared in `f2`, the chain_offset of that reference would be 1. References to locals can be

handled using the same mechanism, with a chain_offset of 0, but instead of using the static pointer to the activation record instance of the subprogram where the variable was declared as the base address, the EP is used.

To illustrate the complete process of nonlocal accesses, consider the - following skeletal JavaScript program:

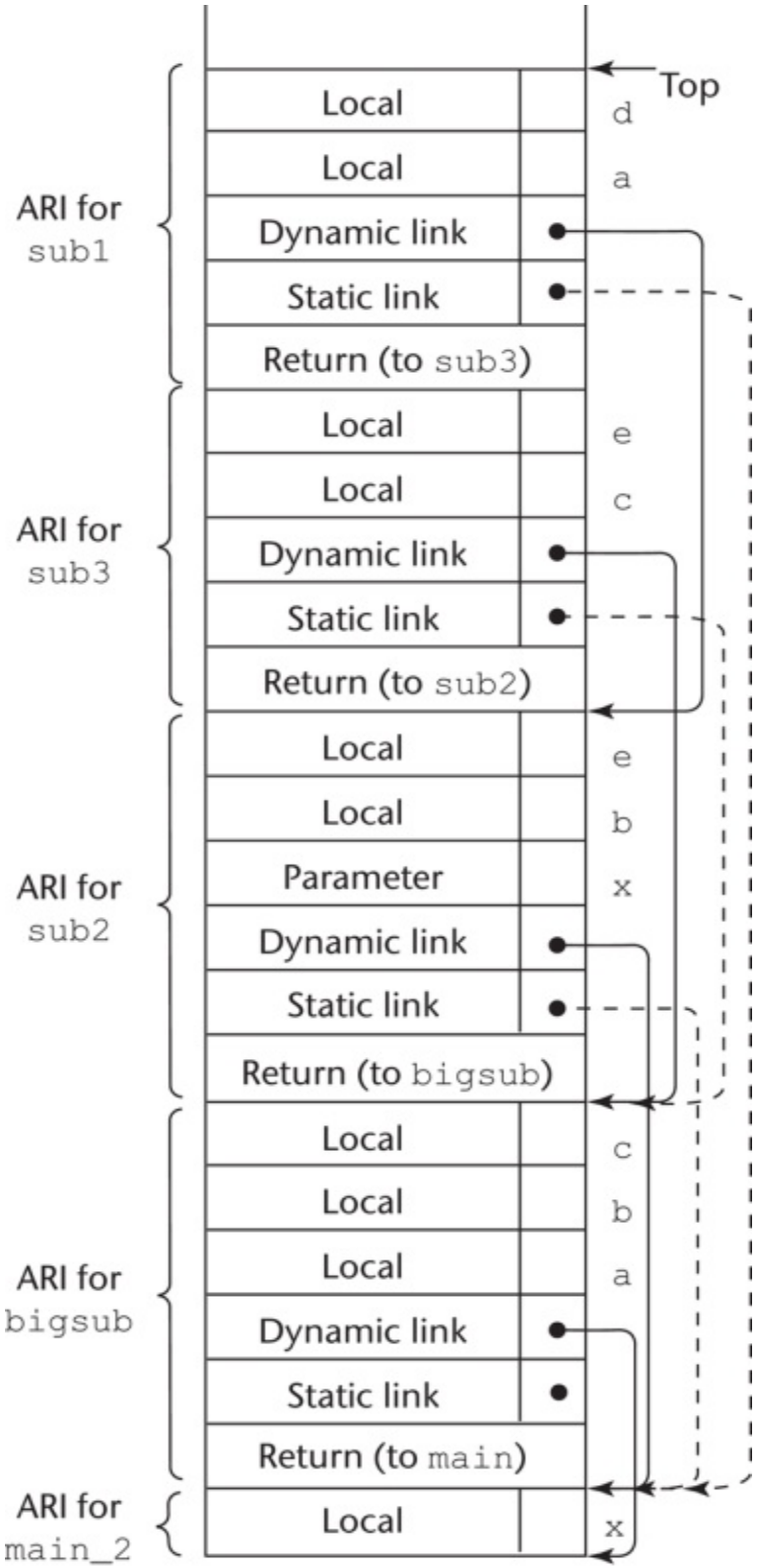
```
function main(){
  var x;
  function bigsub() {
    var a, b, c;
    function sub1 {
      var a, d;
      ...
      a = b + c; <---------------------------------1
      ...
    }  // end of sub1
    function sub2(x) {
      var b, e;
      function sub3() {
        var c, e;
        ...
        sub1();
        ...
        e = b + a; <---------------------------------2
      } // end of sub3
      ...
      sub3();
      ...
      a = d + e; <---------------------------------3
    } // end of sub2
    ...
    sub2(7);
    ...
  } // end of bigsub
  ...
  bigsub();
  ...
} // end of main
```

The sequence of procedure calls is

- `main` calls `bigsub`
- `bigsub` calls `sub2`

- `sub2` calls `sub3`
- `sub3` calls `sub1`

The stack situation when execution first arrives at point 1 in this program is shown in Figure 10.9.



ARI = activation record instance

Figure 10.9 Stack contents at - position 1 in the program `main`

Figure 10.9 Full Alternative Text

At position 1 in procedure `sub1`, the reference is to the local variable, `a`, not to the nonlocal variable `a` from `bigsub`. This reference to `a` has the chain_offset/local_offset pair (0, 3). The reference to `b` is to the nonlocal `b` from `bigsub`. It can be represented by the pair (1, 4). The local_offset is 4, because a 3 offset would be the first local variable (`bigsub` has no parameters). Notice that if the dynamic link were used to do a simple search for an activation record instance with a declaration for the variable `b`, it would find the variable `b` declared in `sub2`, which would be incorrect. If the (1, 4) pair were used with the dynamic chain, the variable `e` from `sub3` would be used. The static link, however, points to the activation record for `bigsub`, which has the correct version of `b`. The variable `b` in `sub2` is not in the referencing environment at this point and is (correctly) not accessible. The reference to `c` at point 1 is to the `c` defined in `bigsub`, which is represented by the pair (1, 5).

After `sub1` completes its execution, the activation record instance for `sub1` is removed from the stack, and control returns to `sub3`. The reference to the variable `e` at position 2 in `sub3` is local and uses the pair (0, 4) for access. The reference to the variable `b` is to the one declared in `sub2`, because that is the nearest static ancestor that contains such a declaration. It is accessed with the pair (1, 4). The local_offset is 4 because `b` is the first variable declared in `sub1`, and `sub2` has one parameter. The reference to the variable `a` is to the `a` declared in `bigsub`, because neither `sub3` nor its static parent `sub2` has a declaration for a variable named `a`. It is referenced with the pair (2, 3).

After `sub3` completes its execution, the activation record instance for `sub3` is removed from the stack, leaving only the activation record instances for `main`, `bigsub`, and `sub2`. At position 3 in `sub2`, the reference to the variable `a` is to the `a` in `bigsub`, which has the only declaration of `a` among the active

routines. This access is made with the pair (1, 3). At this position, there is no visible scope containing a declaration for the variable `d`, so this reference to `d` is a static semantics error. The error would be detected when the compiler attempted to compute the chain_offset/local_offset pair. The reference to `e` is to the local `e` in `sub2`, which can be accessed with the pair (0, 5).

In summary, the references to the variable `a` at points 1, 2, and 3 would be represented by the following points:

- (0, 3) (local)
- (2, 3) (two levels away)
- (1, 3) (one level away)

It is reasonable at this point to ask how the static chain is maintained during program execution. If its maintenance is too complex, the fact that it is simple and effective would be unimportant. We assume here that parameters that are subprograms are not implemented.

The static chain must be modified for each subprogram call and return. The return part is trivial: When the subprogram terminates, its activation record instance is removed from the stack. After this removal, the new top activation record instance is that of the unit that called the subprogram whose execution just terminated. Because the static chain from this activation record instance was never changed, it works correctly just as it did before the call to the other subprogram. Therefore, no other action is required.

The action required at a subprogram call is more complex. Although the correct parent scope is easily determined at compile time, the most recent activation record instance of the parent scope must be found at the time of the call. This can be done by looking at activation record instances on the dynamic chain until the first one of the parent scope is found. However, this search can be avoided by treating subprogram declarations and references exactly like variable declarations and references. When the compiler encounters a subprogram call, among other things, it determines the subprogram that declared the called subprogram, which must be a static ancestor of the calling routine. It then computes the nesting_depth, or number

of enclosing scopes between the caller and the subprogram that declared the called subprogram. This information is stored and can be accessed by the subprogram call during execution. At the time of the call, the static link of the called subprogram's activation record instance is found by moving down the static chain of the caller. The number of links in this move is equal to the nesting_depth, which was computed at compile time.

Consider again the program `main` and the stack situation shown in Figure 10.9. At the call to `sub1` in `sub3`, the compiler determines the nesting_depth of `sub3` (the caller) to be two levels inside the procedure that declared the called procedure `sub1`, which is `bigsub`. When the call to `sub1` in `sub3` is executed, this information is used to set the static link of the activation record instance for `sub1`. This static link is set to point to the activation record instance that is pointed to by the second static link in the static chain from the caller's activation record instance. In this case, the caller is `sub3`, whose static link points to its parent's activation record instance (that of `sub2`). The static link of the activation record instance for `sub2` points to the activation record instance for `bigsub`. So, the static link for the new activation record instance for `sub1` is set to point to the activation record instance for `bigsub`.

This method works for all subprogram linkage, except when parameters that are subprograms are involved.

One criticism of using the static chain approach to access nonlocal variables is that references to variables in scopes beyond the static parent cost more than references to locals. The static chain must be followed, one link per enclosing scope from the reference to the declaration. Fortunately, in practice, references to distant nonlocal variables are rare, so this is not a serious problem. Another criticism of the static-chain approach is that it is difficult for a programmer working on a time-critical program to estimate the costs of nonlocal references, because the cost of each reference depends on the depth of nesting between the reference and the scope of declaration. Further complicating this problem is that subsequent code modifications may change nesting depths, thereby changing the timing of some references, both in the changed code and possibly in code far from the changes.

Some alternatives to static chains have been developed, most notably an

approach that uses an auxiliary data structure called a **display**. However, none of the alternatives has been found to be superior to the static-chain method, which is still the most widely used approach. Therefore, none of the alternatives are discussed here.

The processes and data structures described in this section correctly implement closures in languages that do not permit functions to return functions and do not allow functions to be assigned to variables. However, they are inadequate for languages that do allow one or both of those operations. Several new mechanisms are needed to implement access to nonlocals in such languages. First, if a subprogram accesses a variable from a nesting but not global scope, that variable cannot be stored only in the activation record of its home scope. That activation record could be deallocated before the subprogram that needs it is activated. Such variables could also be stored in the heap and given unlimited extend (their lifetimes are the lifetime of the whole program). Second, subprograms must have mechanisms to access the nonlocals that are stored in the heap. Third, the heap-allocated variables that are nonlocally accessed must be updated every time their stack versions are updated. Clearly, these are nontrivial extensions to the implementation static scoping using static chains.

10.5 Blocks

Recall from Chapter 5, that a number of languages, including the C-based languages, provide for user-specified local scopes for variables called **blocks**. As an example of a block, consider the following code segment:

```
{ int temp;
  temp = list[upper];
  list[upper] = list[lower];
  list[lower] = temp;
}
```

A block is specified in the C-based languages as a compound statement that begins with one or more data definitions. The lifetime of the variable `temp` in the preceding block begins when control enters the block and ends when control exits the block. The advantage of using such a local is that it cannot interfere with any other variable with the same name that is declared elsewhere in the program, or more specifically, in the referencing environment of the block.

Blocks can be implemented by using the static-chain process described in Section 10.4 for implementing nested subprograms. Blocks are treated as parameterless subprograms that are always called from the same place in the program. Therefore, every block has an activation record. An instance of its activation record is created every time the block is executed.

Blocks can also be implemented in a different and somewhat simpler and more efficient way. The maximum amount of storage required for block variables at any time during the execution of a program can be statically determined, because blocks are entered and exited in strictly textual order. This amount of space can be allocated after the local variables in the activation record. Offsets for all block variables can be statically computed, so block variables can be addressed exactly as if they were local variables.

For example, consider the following skeletal program:

```
void main() {
```

```
  int x, y, z;
  while ( . . ) {
    int a, b, c;
    .
    while ( . . . ) {
      int d, e;
      .
    }
  }
  while ( . . . ) {
    int f, g;
    . . .
  }
  . .
}
```

For this program, the static-memory layout shown in Figure 10.10 could be used. Note that `f` and `g` occupy the same memory locations as `a` and `b`, because `a` and `b` are popped off the stack when their block is exited (before `f` and `g` are allocated).

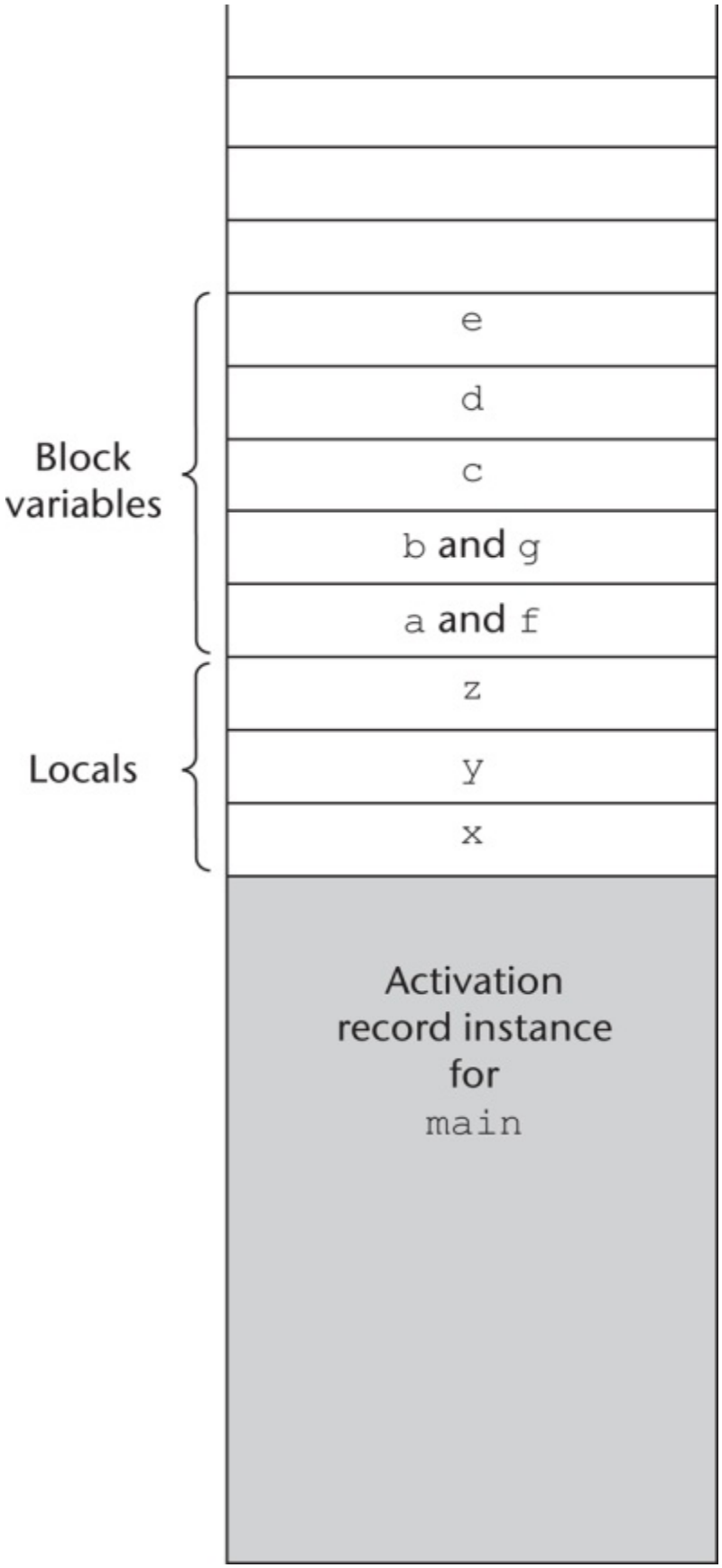

Block variables
Locals
e
d
c
b and g
a and f
z
y
x
Activation record instance for main

Figure 10.10 Block variable - storage when blocks are not treated as parameterless procedures

Figure 10.10 Full Alternative Text

10.6 Implementing Dynamic Scoping

There are at least two distinct ways in which local variables and nonlocal references to them can be implemented in a dynamic-scoped language: deep access and shallow access. Note that deep access and shallow access are not concepts related to deep and shallow binding. An important difference between binding and access is that deep and shallow bindings result in different semantics; deep and shallow accesses do not.

10.6.1 Deep Access

If local variables are stack dynamic and are part of the activation records in a dynamic-scoped language, references to nonlocal variables can be resolved by searching through the activation record instances of the other subprograms that are currently active, beginning with the one most recently activated. This concept is similar to that of accessing nonlocal variables in a static-scoped language with nested subprograms, except that the dynamic—rather than the static—chain is followed. The dynamic chain links together all subprogram activation record instances in the reverse of the order in which they were activated. Therefore, the dynamic chain is exactly what is needed to reference nonlocal variables in a dynamic-scoped language. This method is called **deep access**, because access may require searches deep into the stack.

Consider the following example skeletal program:

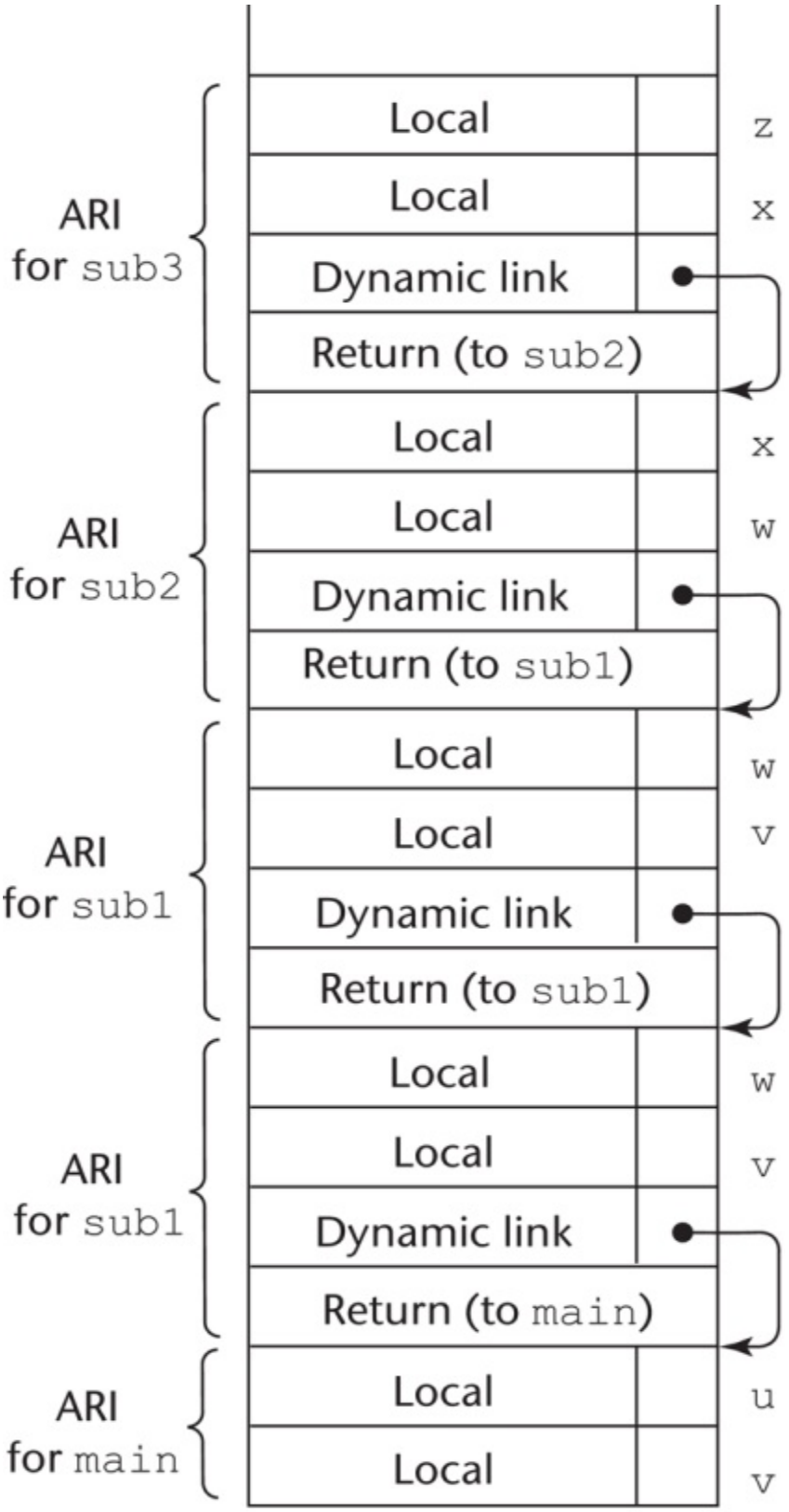
```
void sub3() {
  int x, z;
  x = u + v;
  . . .
}
void sub2() {
  int w, x;
  . . .
```

```
}
void sub1() {
  int v, w;
  . . .
}
void main() {
  int v, u;
  . . .
}
```

This program is written in a syntax that gives it the appearance of a program in a C-based language, but it is not meant to be in any particular language. Suppose the following sequence of function calls occurs:

- `main` calls `sub1`
- `sub1` calls `sub1`
- `sub1` calls `sub2`
- `sub2` calls `sub3`

Figure 10.11 shows the stack during the execution of function `sub3` after this calling sequence. Notice that the activation record instances do not have static links, which would serve no purpose in a dynamic-scoped language.



ARI = activation record instance

Figure 10.11 Stack contents for a dynamic-scoped program

Figure 10.11 Full Alternative Text

Consider the references to the variables `x`, `u`, and `v` in function `sub3`. The reference to `x` is found in the activation record instance for `sub3`. The reference to `u` is found by searching *all* of the activation record instances on the stack, because the only existing variable with that name is in `main`. This search involves following four dynamic links and examining 10 variable names. The reference to `v` is found in the most recent (nearest on the dynamic chain) activation record instance for the subprogram `sub1`.

There are two important differences between the deep-access method for nonlocal access in a dynamic-scoped language and the static-chain method for static-scoped languages. First, in a dynamic-scoped language, there is no way to determine at compile time the length of the chain that must be searched. Every activation record instance in the chain must be searched until the first instance of the variable is found. This is one reason why dynamic-scoped languages typically have slower execution speeds than static-scoped languages. Second, activation records must store the names of variables for the search process, whereas in static-scoped language implementations only the values are required. (Names are not required for static scoping, because all variables are represented by the chain_offset/local_offset pairs.)

10.6.2 Shallow Access

Shallow access is an alternative implementation method, not an alternative semantics. As stated previously, the semantics of deep access and shallow access are identical. In the shallow-access method, variables declared in subprograms are not stored in the activation records of those subprograms. Because with dynamic scoping there is at most one visible version of a

variable of any specific name at a given time, a very different approach can be taken. One variation of shallow access is to have a separate stack for each variable name in a complete program. Every time a new variable with a particular name is created by a declaration at the beginning of a subprogram that has been called, the variable is given a cell at the top of the stack for its name. Every reference to the name is to the variable on top of the stack associated with that name, because the top one is the most recently created. When a subprogram terminates, the lifetimes of its local variables end, and the stacks for those variable names are popped. This method allows fast references to variables, but maintaining the stacks at the entrances and exits of subprograms is costly.

Figure 10.12 shows the variable stacks for the earlier example program in the same situation as shown with the stack in Figure 10.11.

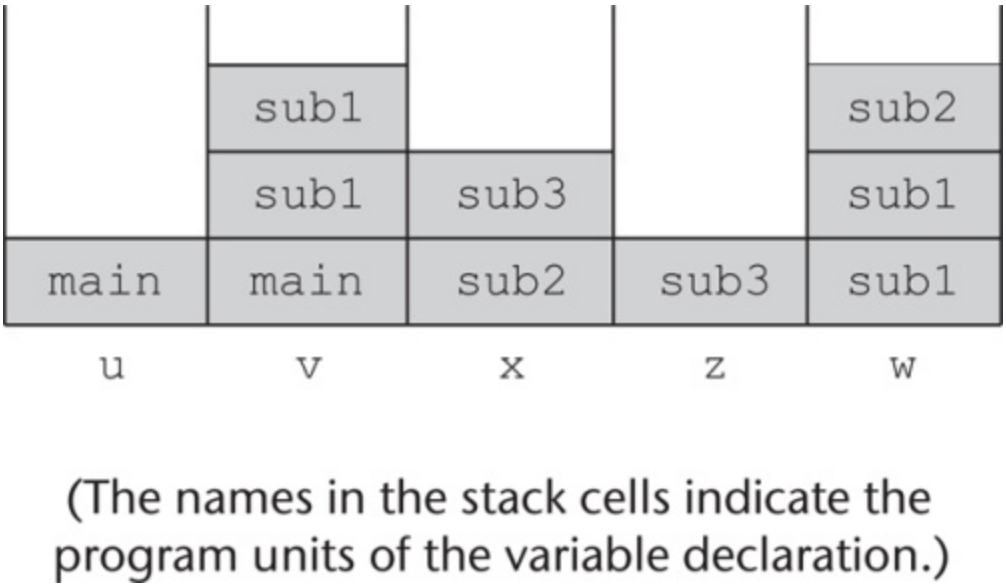


Figure 10.12 One method of using shallow access to - implement dynamic scoping

Figure 10.12 Full Alternative Text

Another option for implementing shallow access is to use a central table that has a location for each different variable name in a program. Along with each entry, a bit called **active** is maintained that indicates whether the name has a current binding or variable association. Any access to any variable can then be to an offset into the central table. The offset is static, so the access can be fast. SNOBOL implementations use the central table implementation technique.

Maintenance of a central table is straightforward. A subprogram call requires that all of its local variables be logically placed in the central table. If the position of the new variable in the central table is already active—that is, if it contains a variable whose lifetime has not yet ended (which is indicated by the active bit)—that value must be saved somewhere during the lifetime of the new variable. Whenever a variable begins its lifetime, the active bit in its central table position must be set.

There have been several variations in the design of the central table and in the way values are stored when they are temporarily replaced. One variation is to have a “hidden” stack on which all saved objects are stored. Because subprogram calls and returns, and thus the lifetimes of local variables, are nested, this works well.

The second variation is perhaps the cleanest and least expensive to implement. A central table of single cells is used, storing only the current version of each variable with a unique name. Replaced variables are stored in the activation record of the subprogram that created the replacement variable. This is a stack mechanism, but it uses the stack that already exists, so the new overhead is minimal.

The choice between shallow and deep access to nonlocal variables depends on the relative frequencies of subprogram calls and nonlocal references. The deep-access method provides fast subprogram linkage, but references to nonlocals, especially references to distant nonlocals (in terms of the call chain), are costly. The shallow-access method provides much faster references to nonlocals, especially distant nonlocals, but is more costly in terms of subprogram linkage.

SUMMARY

Subprogram linkage semantics requires many actions by the implementation. In the case of “simple” subprograms, these actions are relatively uncomplicated. At the call, the status of execution must be saved, parameters and the return address must be passed to the called subprogram, and control must be transferred. At the return, the values of pass-by-result and pass-by-value-result parameters must be transferred back, as well as the return value if it is a function, execution status must be restored, and control transferred back to the caller. In languages with stack-dynamic local variables and nested subprograms, subprogram linkage is more complex. There may be more than one activation record instance, those instances must be stored on the run-time stack, and static and dynamic links must be maintained in the activation record instances. The static link is to allow references to nonlocal variables in static-scoped languages.

Subprograms in languages with stack-dynamic local variables and nested subprograms have two components: the actual code, which is static, and the activation record, which is stack dynamic. Activation record instances contain the formal parameters and local variables, among other things. Access to nonlocal variables is implemented with a chain of static parent pointers.

Access to nonlocal variables in a dynamic-scoped language can be implemented by use of the dynamic chain or through some central variable table method. Dynamic chains provide slow accesses but fast calls and returns. The central table methods provide fast accesses but slow calls and returns.

REVIEW QUESTIONS

1. What is the definition used in this chapter for “simple” subprograms?
2. Which of the caller or callee saves execution status information?
3. What must be stored for the linkage to a subprogram?
4. What is the task of a linker?
5. What are the two reasons why implementing subprograms with stack-dynamic local variables is more difficult than implementing simple subprograms?
6. What is the difference between an activation record and an activation record instance?
7. Why are the return address, dynamic link, and parameters placed in the bottom of the activation record?
8. What kind of machines often use registers to pass parameters?
9. What are the two steps in locating a nonlocal variable in a static-scoped language with stack-dynamic local variables and nested subprograms?
10. Define *static chain*, *static_depth*, *nesting_depth*, and *chain_offset*.
11. What is an EP, and what is its purpose?
12. How are references to variables represented in the static-chain method?
13. Name three widely used programming languages that do not allow nested subprograms.
14. What are the two potential problems with the static-chain method?

15. Explain the two methods of implementing blocks.

16. Describe the deep-access method of implementing dynamic scoping.

17. Describe the shallow-access method of implementing dynamic scoping.

18. What are the two differences between the deep-access method for nonlocal access in dynamic-scoped languages and the static-chain method for static-scoped languages?

19. Compare the efficiency of the deep-access method to that of the shallow-access method, in terms of both calls and nonlocal accesses.

PROBLEM SET

1. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume `bigsub` is at level 1.

```
function bigsub() {
  function a() {
     function b() {
        ... <----------------------------1
     } // end of b
     function  c() {
       ...
       b();
       ...
     } // end of c
     ...
     c();
     ...
  } // end of a
  ...
  a();
  ...
} // end of bigsub
```

2. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume `bigsub` is at level 1.

```
function bigsub() {
  var mysum;
  function a() {
   var x;
    function b(sum) {
     var y, z;
     ...
     c(z);
     ...
    } // end of b
    ...
    b(x);
    ...
```

```
   } // end of a
   function c(plums) {
     ... <-----------------------------1
   } // end of c
   var l;
   ...
   a();
   ...
} // end of bigsub
```

3. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume `bigsub` is at level 1.

```
function bigsub() {
function a(flag) {
  function b() {
    ...
    a(false);
    ...
 } // end of b
 ...
  if (flag)
    b();
  else c();
 ...
} // end of a
 function c() {
   function d() {
     ... <------------------------1
   } // end of d
 ...
   d();
    ...
  } // end of c
  ...
  a(true);
  ...
} // end of bigsub
```

The calling sequence for this program for execution to reach `d` is

`bigsub` calls `a`

`a` calls `b`

`b` calls `a`

`a` calls `c`

`c` calls `d`

4. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. This program uses the deep-access method to implement dynamic scoping.

```
void fun1() {
  float a;
  . . .
}
void fun2() {
  int b, c;
  . . .
}
void fun3() {
  float d;
  . . . <--------- 1
}
void main() {
  char e, f, g;
   . . .
}
```

The calling sequence for this program for execution to reach `fun3` is

`main` calls `fun2`

`fun2` calls `fun1`

`fun1` calls `fun1`

`fun1` calls `fun3`

5. Assume that the program of Problem 4 is implemented using the shallow-access method using a stack for each variable name. Show the stacks for the time of the execution of `fun3`, assuming execution found its way to that point through the sequence of calls shown in Problem 4.

6. Although local variables in Java methods are dynamically allocated at the beginning of each activation, under what circumstances could the value of a local variable in a particular activation retain the value of the previous activation?

7. It is stated in this chapter that when nonlocal variables are accessed in a dynamic-scoped language using the dynamic chain, variable names must be stored in the activation records with the values. If this were actually done, every nonlocal access would require a sequence of costly string comparisons on names. Design an alternative to these string comparisons that would be faster.

8. Pascal allows gotos with nonlocal targets. How could such statements be handled if static chains were used for nonlocal variable access? *Hint*: Consider the way the correct activation record instance of the static parent of a newly enacted procedure is found (see Section 10.4.2).

9. The static-chain method could be expanded slightly by using two static links in each activation record instance where the second points to the static grandparent activation record instance. How would this approach affect the time required for subprogram linkage and nonlocal references?

10. Design a skeletal program and a calling sequence that results in an activation record instance in which the static and dynamic links point to different activation-recorded instances in the run-time stack.

11. If a compiler uses the static chain approach to implementing blocks, which of the entries in the activation records for subprograms are needed in the activation records for blocks?

12. Examine the subprogram call instructions of three different architectures, including at least one CISC machine and one RISC machine, and write a short comparison of their capabilities. (The design of these instructions usually determines at least part of the compiler writer's design of subprogram linkage.)

PROGRAMMING EXERCISES

1. Write a program that includes two subprograms, one that takes a single parameter and performs some simple operation on that parameter and one that takes 20 parameters and uses all of the parameters, but only for one simple operation. The main program must call these two subprograms a large number of times. Include in the program timing code to output the run time of the calls to each of the two subprograms. Run the program on a RISC machine and on a CISC machine and compare the ratios of the time required by the two subprograms. Based on the results, what can you say about the speed of parameter passing on the two machines?

11 Abstract Data Types and Encapsulation Constructs

In this chapter, we explore programming language constructs that support data abstraction. Among the new ideas of the last 50 years in programming methodologies and programming language design, data abstraction is one of the most profound.

We begin by discussing the general concept of abstraction in programming and programming languages. Data abstraction is then defined and illustrated with an example. This topic is followed by descriptions of the support for data abstraction in C++, Java, C#, and Ruby. To illuminate the similarities and differences in the design of the language facilities that support data abstraction, implementations of the same example data abstraction are given in C++, Java, and Ruby. Next, the capabilities of C++, Java 5.0, and C# 2005 to build parameterized abstract data types are discussed.

All the languages used in this chapter to illustrate the concepts and constructs of abstract data types support object-oriented programming, because virtually all contemporary languages support object-oriented programming and nearly all of those that do not, and yet support abstract data types, have faded into

obscurity.

Constructs that support abstract data types are encapsulations of the data and operations on objects of the type. Encapsulations that contain multiple types are required for the construction of larger programs. These encapsulations and the associated namespace issues are also discussed in this chapter.

Some programming languages support logical, as opposed to physical, encapsulations, which are actually used to encapsulate names. These are discussed in Section 11.7.

11.1 The Concept of Abstraction

An **abstraction** is a view or representation of an entity that includes only the most significant attributes. In a general sense, abstraction allows one to collect instances of entities into groups in which their common attributes need not be considered. For example, suppose we define birds to be creatures with the following attributes: two wings, two legs, a tail, and feathers. Then, if we say a crow is a bird, a description of a crow need not include those attributes. The same is true for robins, sparrows, and yellow-bellied sapsuckers. The common attributes in the descriptions of specific species of birds can be abstracted away, because all species have them. Within a particular species, only the attributes that distinguish that species need be considered. For example, crows have the attributes of being black, being of a particular size, and being noisy. A description of a crow needs to provide those attributes, but not the others that are common to all birds. This results in significant simplification of the descriptions of members of the species. A less abstract view of a species, that of a bird, may be considered when it is necessary to see a higher level of detail, rather than just the special attributes.

In the world of programming languages, abstraction is a weapon against the complexity of programming; its purpose is to simplify the programming process. It is an effective weapon because it allows programmers to focus on essential attributes, while ignoring subordinate attributes.

The two fundamental kinds of abstraction in contemporary programming languages are process abstraction and data abstraction.

The concept of **process abstraction** is among the oldest in programming language design (Plankalkül supported process abstraction in the 1940s). All subprograms are process abstractions because they provide a way for a program to specify a process, without providing the details of how it performs its task (at least in the calling program). For example, when a program needs to sort an array of numeric data of some type, it usually uses a subprogram for the sorting process. At the point where the sorting process is required, a statement such as

```
sortInt(list, listLen)
```

is placed in the program. This call is an abstraction of the actual sorting process, whose algorithm is not specified. The call is independent of the algorithm implemented in the called subprogram.

In the case of the subprogram `sortInt`, the only essential attributes are the name of the array to be sorted, the type of its elements, the array's length, and the fact that the call to `sortInt` will result in the array being sorted. The particular algorithm that `sortInt` implements is an attribute that is not essential to the user. The user needs to see only the name and protocol of the sorting subprogram to be able to use it.

The widespread use of data abstraction necessarily followed that of process abstraction because an integral and essential part of every data abstraction is its operations, which are defined as process abstractions.

11.2 Introduction to Data Abstraction

The evolution of data abstraction began in 1960 with the first version of COBOL, which included the record data structure.1 The C-based languages have structs, which are also records. An abstract data type is a data structure, in the form of a record, but which includes subprograms that manipulate its data.

1. Recall from Chapter 2 that a record is a data structure that stores fields, which have names and can be of different types.

Syntactically, an abstract data type is an enclosure that includes only the data representation of one specific data type and the subprograms that provide the operations for that type. Through access controls, unnecessary details of the type can be hidden from units outside the enclosure that use the type. Program units that use an abstract data type can declare variables of that type, even though the actual representation is hidden from them. An instance of an abstract data type is called an **object**.

One of the motivations for data abstraction is similar to that of process abstraction. It is a weapon against complexity; a means of making large and/or complicated programs more manageable. Other motivations for and advantages of abstract data types are discussed later in this section.

Object-oriented programming, which is described in Chapter 12, is an outgrowth of the use of data abstraction in software development, and data abstraction is one of its fundamental components.

11.2.1 Floating-Point as an Abstract Data Type

The concept of an abstract data type, at least in terms of built-in types, is not a recent development. All built-in data types, even those of Fortran I, are abstract data types, although they are rarely called that. For example, consider a floating-point data type. Most programming languages include at least one of these. A floating-point type provides the means to create variables to store floating-point data and also provides a set of arithmetic operations for manipulating objects of the type.

Floating-point types in high-level languages employ a key concept in data abstraction: information hiding. The actual format of the floating-point data value in a memory cell is hidden from the user, and the only operations available are those provided by the language. The user is not allowed to create new operations on data of the type, except those that can be constructed using the built-in operations. The user cannot directly manipulate the parts of the actual representation of values because that representation is hidden. It is this feature that allows program portability between implementations of a particular language, even though the implementations may use different representations for particular data types. For example, before the IEEE 754 standard floating-point representations appeared in the mid-1980s, there were several different representations being used by different computer architectures. However, this variation did not prevent programs that used floating-point types from being portable among the various architectures.

11.2.2 User-Defined Abstract Data Types

A user-defined abstract data type should provide the same characteristics as those of language-defined types, such as a floating-point type: (1) a type definition that allows program units to declare variables of the type but hides the representation of objects of the type; and (2) a set of operations for manipulating objects of the type.

We now formally define an abstract data type in the context of user-defined types. An **abstract data type** is a data type that satisfies the following

conditions:

- The representation of objects of the type is hidden from the program units that use the type, so the only direct operations possible on those objects are those provided in the type's definition.

- The declarations of the type and the protocols of the operations on objects of the type, which provide the type's interface, are contained in a single syntactic unit. The type's interface does not depend on the representation of the objects or the implementation of the operations. Also, other program units are allowed to create variables of the defined type.

There are several benefits of information hiding. One of these is increased reliability. Program units that use a specific abstract data type are called **clients** of that type. Clients cannot manipulate the underlying representations of objects directly, either intentionally or by accident, thus increasing the integrity of such objects. Objects can be changed only through the provided operations.

Another benefit of information hiding is it reduces the range of code and number of variables of which a programmer must be aware when writing or reading a part of the program. The value of a particular variable can be changed only by code in a restricted range, making the code easier to understand and making it less challenging to find sources of incorrect changes.

Information hiding also makes name conflicts less likely, because the scopes of variables is smaller.

Finally, consider the following advantage of information hiding: Suppose that the original implementation of the stack abstraction uses a linked list representation. At a later time, because of memory management problems with that representation, the stack abstraction is changed to use a contiguous representation (one that implements a stack in an array). Because data abstraction was used, this change can be made in the code that defines the stack type, but no changes will be required in any of the clients of the stack abstraction. Of course, a change in protocol of any of the operations would

require changes in the clients.

Although the definition of an abstract data type specifies that data members of objects must be hidden from clients, many situations arise in which clients need to access these data members. The common solution is to provide accessor methods, sometimes called **getters** and **setters**, that allow clients indirect access to the so-called hidden data—a better solution than simply making the data public, which would provide direct access. There are three reasons why accessors are better:

1. Read-only access can be provided by having a getter method but no corresponding setter method.

2. Constraints can be included in setters. For example, if the data value should be restricted to a particular range, the setter can enforce that.

3. The actual implementation of the data member can be changed without affecting the clients if getters and setters are the only access.

Both specifying data in an abstract data type to be public and providing accessor methods for that data are violations of the principles of abstract data types. Some believe these are simply loopholes that make an imperfect design usable. As we will see in Section 11.4.4.2, Ruby disallows making instance data public. However, Ruby also makes it very easy to create accessor functions. It is a challenge for developers to design abstract data types in which all of the data is actually hidden.

The primary advantage of packaging the declarations of the type and its operations in a single syntactic unit is that it provides a method of organizing a program into logical units that can be compiled separately. In some cases, the implementation is included with the type declaration; in other cases, it is in a separate syntactic unit. The advantage of having the implementation of the type and its operations in different syntactic units is that it increases the program's modularity and it is a clear separation of design and implementation. If both the declarations and the definitions of types and operations are in the same syntactic unit, there must be some means of hiding from client program units the parts of the unit that specify the definitions.

11.2.3 An Example

A stack is a widely applicable data structure that stores some number of data elements and only allows access to the data element at one of its ends, the top. Suppose an abstract data type is to be constructed for a stack that has the following abstract operations:

create(stack)	Creates and possibly initializes a stack object
destroy(stack)	Deallocates the storage for the stack
empty(stack)	A predicate (or Boolean) function that returns true if the specified stack is empty and false otherwise
push(stack, element)	Pushes the specified element on the specified stack
pop(stack)	Removes the top element from the specified stack
top(stack)	Returns a copy of the top element from the specified stack

Note that some implementations of abstract data types do not require the create and destroy operations. For example, simply defining a variable to be of an abstract data type may implicitly create the underlying data structure and initialize it. The storage for such a variable may be implicitly deallocated at the end of the variable's scope.

A client of the stack type could have a code sequence such as the following:

```
. . .
create(stk1);
push(stk1, color1);
push(stk1, color2);
temp = top(stk1);
. . .
```

11.3 Design Issues for Abstract Data Types

A facility for defining abstract data types in a language must provide a syntactic unit that encloses the declaration of the type and the prototypes of the subprograms that implement the operations on objects of the type. It must be possible to make these visible to clients of the abstraction. This allows clients to declare variables of the abstract type and manipulate their values. Although the type name must have external visibility, the type representation must be hidden. The type representation and the definitions of the subprograms that implement the operations may appear inside or outside this syntactic unit.

Few, if any, general built-in operations should be provided for objects of abstract data types, other than those provided with the type definition. There simply are not many operations that apply to a broad range of abstract data types. Among these are assignment and comparisons for equality and inequality. If the language does not allow users to overload assignment, the assignment operation must be included in the abstraction. Comparisons for equality and inequality should be predefined in the abstraction in some cases but not in others. For example, if the type is implemented as a pointer, equality may mean pointer equality, but the designer may want it to mean equality of the structures referenced by the pointers.

Some operations are required by many abstract data types, but because they are not universal, they often must be provided by the designer of the type. Among these are iterators, accessors, constructors, and destructors. Iterators were discussed in Chapter 8. Accessors provide a form of access to data that is hidden from direct access by clients. Constructors are used to initialize parts of newly created objects. Destructors are often used to reclaim heap storage that may be used by parts of abstract data type objects in languages that do not do implicit storage reclamation.

As stated earlier, the enclosure for an abstract data type defines a single data

type and its operations. Many contemporary languages, including C++, Java, and C#, directly support abstract data types.

The first design issue is whether abstract data types can be parameterized. For example, if the language supports parameterized abstract data types, one could design an abstract data type for some structure that could store elements of any type. Parameterized abstract data types are discussed in Section 11.5. The second design issue is what access controls are provided and how such controls are specified. Finally, the language designer must decide whether the specification of the type is physically separate from its implementation (or whether that is a developer choice).

11.4 Language Examples

The concept of data abstraction had its origins in SIMULA 67, although that language did not provide complete support for abstract data types, because it did not include a way to hide implementation details. In this section, we describe the support for data abstraction provided by C++, Java, C#, and Ruby.

11.4.1 Abstract Data Types in C++

C++, which was first released in 1985, was created by adding features to C. The first important additions were those to support object-oriented programming. Because one of the primary components of object-oriented programming is abstract data types, C++ obviously is required to support them.

C++ provides two constructs that are very similar to each other, the class and the struct, which directly support abstract data types. Because structs are most commonly used when only data is included, we do not discuss them further here.

interview

C++: Its Birth, Its Ubiquitousness, and Common Criticisms

BJARNE STROUSTRUP

Bjarne Stroustrup is the designer and original implementer of C++ and the author of A Tour of C++, Programming—Principles and Practice using C++, The C++ Programming Language, The Design and Evolution of C++, and many other publications. His research interests include distributed systems, design, programming techniques, software development tools, and programming languages. He is actively involved in the ANSI/ISO standardization of C++. Dr. Stroustrup is a Managing Director in the technology division of Morgan Stanley in New York City, a Visiting Professor in Computer Science at Columbia University, and a Distinguished Research Professor in Computer Science at Texas A&M University. He is a member of the National Academy of Engineering, an ACM Fellow, and an IEEE fellow. In 1993, Stroustrup received the ACM Grace Murray Hopper Award “for his early work laying the foundations for the C++ programming language. Based on the foundations and Dr. Stroustrup’s continuing efforts, C++ has become one of the most influential programming languages in the history of computing.”

(year of interview: 2002)

A BRIEF HISTORY OF YOU AND COMPUTING

What were you working on, and where, before you joined Bell Labs in the early 1980s? At Bell Labs, I was doing research in the general area of distributed systems. I joined in 1979. Before that, I was finishing my Ph.D. in that field in Cambridge University.

Did you immediately start on "C with Classes" (which would later become C ++)? I worked on a few projects related to distributed computing before starting on C with Classes and during the development of that and of C++. For example, I was trying to find a way to distribute the UNIX kernel across several computers and helped a lot of projects build simulators.

Was it an interest in mathematics that got you into this profession? I signed up for a degree in "mathematics with computer science" and my master's degree is officially a math degree. I—wrongly—thought that computing was some kind of applied math. I did a couple of years of math and rate myself a poor mathematician, but that's still much better than not knowing math. At the time I signed up, I had never even seen a computer. What I love about computing is the programming rather than the more mathematical fields.

DISSECTING A SUCCESSFUL LANGUAGE

I'd like to work backward, listing some items I think make C++ ubiquitous, and get your reaction. It's "open source," nonproprietary, and standardized by ANSI/ISO. The ISO C++ standard is important. There are many independently developed and evolving C++ implementations. Without a standard for them to adhere to and a standards process to help coordinate the evolution of C++, a chaos of dialects would erupt.

It is also important that there are both open-source and commercial implementations available. In addition, for many users, it is crucial that the standard provides a measure of protection from manipulation by implementation providers.

The ISO standards process is open and democratic. The C++ committee rarely meets with fewer than 50 people present and typically more than eight nations are represented at each meeting. It is not just a vendors' forum.

It's ideal for systems programming (which, at the time C++ was born, was the largest sector of the market developing code).

Yes, C++ is a strong contender for any systems-programming project. It is also effective for embedded systems programming, which is currently the fastest-growing sector. Yet another growth area for C++ is high-performance numeric/engineering/scientific programming.

Its object-oriented nature and inclusion of classes/libraries make programming more efficient and transparent. C++ is a multiparadigm programming language. That is, it supports several fundamental styles of programming (including object-oriented programming) and combinations of those styles. When used well, this leads to cleaner, more flexible, and more efficient libraries than can be provided using just one paradigm. The C++ standard library containers and algorithms, which is basically a generic programming framework, is an example. When used together with (object-oriented) class hierarchies, the result is an unsurpassed combination of type safety, efficiency, and flexibility.

Its incubation in the AT&T development environment AT&T Bell Labs provided an environment that was crucial for C++'s development. The labs were an exceptionally rich source of challenging problems and a uniquely supportive environment for practical research. C++ emerged from the same research lab as C did and benefited from the same intellectual tradition, experience, and exceptional people. Throughout, AT&T supported the standardization of C++. However, C++ was not the beneficiary of a massive marketing campaign, like many modern languages. That's simply not the way the labs work.

Did I miss anything on your top list? Undoubtedly.

Now, let me paraphrase from the C++ critiques and get your reactions: It's huge/unwieldy. The "hello world" problem is 10 times larger in C++ than in C. C++ is certainly not a small language, but then few modern languages are. If a language is small, you tend to need huge libraries to get work done and often have to rely on conventions and extensions. I prefer to have key parts of the inevitable complexity in the language where it can be seen, taught, and effectively standardized rather than hidden elsewhere in a system. For most purposes, I don't consider C++ unwieldy. The C++ "hello world" program isn't larger than its C equivalent on my machine, and it shouldn't be on yours.

In fact, the object code for the C++ version of the "hello world" program is smaller than the C version on my machine. There is no language reason why the one version should be larger than the other. It is all an issue of how the implementor organized the libraries. If one version is significantly larger than the other, report the problem to the implementor of the larger version.

It's tougher to program in C++ (compared with C). (Something the critics say.) Even you once admitted it, saying something about shooting yourself in the foot with C versus C ++. Yes, I did say something along the lines of "C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, C++ blows your whole leg off." What people tend to miss is that what I said about C++ is to a varying extent true for all powerful languages. As you protect people from simple dangers, they get themselves into new and less obvious problems. Someone who avoids the simple problems may simply be heading for a not-so-simple one. One problem with very supporting and protective environments is that the hard problems may be discovered too late or be too hard to remedy once discovered. Also, a rare problem is harder to find than a frequent one because you don't suspect it.

It's appropriate for embedded systems of today but not for the Internet software of today. C++ is suitable for embedded systems today. It is also suitable—and widely used—for "Internet software" today. For example, have a look at my "C++ applications" Web page. You'll notice that some of the major Web service providers, such as Amazon, Adobe, Google, Quicken, and Microsoft, critically rely on C++. Gaming is a related area in which you find

heavy C++ use.

Did I miss another one that you get a lot? Sure.

C++ classes are types. A C++ program unit that declares an instance of a class can also access any of the public entities in that class, but only through an instance of the class.

11.4.1.1 Encapsulation

The data defined in a C++ class are called **data members**; the functions (methods) defined in a class are called **member functions**. Data members and member functions appear in two categories: class and instance. Class members are associated with the class; instance members are associated with the instances of the class. In this chapter, only the instance members of a class are discussed. All of the instances of a class share a single set of member functions, but each instance has its own set of the class's data members. Class instances can be static, stack dynamic, or heap dynamic. If static or stack dynamic, they are referenced directly with value variables. If heap dynamic, they are referenced through pointers. Stack dynamic instances of classes are always created by the elaboration of an object declaration. Furthermore, the lifetime of such a class instance ends when the end of the scope of its declaration is reached. Heap dynamic class instances are created with the `new` operator and destroyed with the `delete` operator. Both stack- and heap-dynamic classes can have pointer data members that reference heap dynamic data, so that even though a class instance is stack dynamic, it can include data members that reference heap dynamic data.

A member function of a class can be defined in two distinct ways: The complete definition can appear in the class, or only in its header. When both the header and the body of a member function appear in the class definition, the member function is implicitly inlined. Recall that this means that its code is placed in the caller's code, rather than requiring the usual call and return linkage. If only the header of a member function appears in the class definition, its complete definition appears outside the class and is separately compiled. The rationale for allowing member functions to be inlined was to

save function call overhead in real-time applications, in which run-time efficiency is of utmost importance. The downside of inlining member functions is that it clutters the class definition interface, resulting in a reduction in readability.

Placing member function definitions outside the class definition separates specification from implementation, a common goal of modern programming.

11.4.1.2 Information Hiding

A C++ class can contain both hidden and visible entities (meaning they are either hidden from or visible to clients of the class). Entities that are to be hidden are placed in a **`private`** clause, and visible, or public, entities appear in a **`public`** clause. The **`public`** clause therefore describes the interface to class instances. There is also a third category of visibility, **`protected`**, which makes a member visible to subclasses, but not to clients.

11.4.1.3 Constructors and Destructors

C++ allows the user to include **constructor** functions in class definitions, which are used to initialize the data members of newly created objects. A constructor may also allocate the heap-dynamic data that are referenced by the pointer members of the new object. Constructors are implicitly called when an object of the class type is created. A constructor has the same name as the class whose objects it initializes. Constructors can be overloaded, but of course each constructor of a class must have a unique parameter profile.

A C++ class can also include a function called a **destructor**, which is implicitly called when the lifetime of an instance of the class ends. As stated earlier, stack-dynamic class instances can contain pointer members that reference heap-dynamic data. The destructor function for such an instance can include a **`delete`** operator on the pointer members to deallocate the heap

space they reference. Destructors are often used as a debugging aid, in which case they display or print the values of some or all of the object's data members before those members are deallocated. The name of a destructor is the class's name, preceded by a tilde (~).

Neither constructors nor destructors have return types, and neither use **`return`** statements. Both constructors and destructors can be explicitly called.

11.4.1.4 An Example

Our example of a C++ abstract data type is a stack:

```
#include <iostream.h>
class Stack {
  private:  //** These members are visible only to other
            //** members and friends (see Section 11.6.4)
    int *stackPtr;
    int maxLen;
    int topSub;
  public:  //** These members are visible to clients
    Stack() { //** A constructor
      stackPtr = new int [100];
      maxLen = 99;
      topSub = -1;
    }
    ~Stack() {delete  [] stackPtr;};  //** A destructor
    void push(int number) {
      if (topSub == maxLen)
        cerr << "Error in push--stack is full\n";
      else stackPtr[++topSub] = number;
    }
    void pop() {
      if (empty())
        cerr << "Error in pop--stack is empty\n";
      else topSub--;
    }
    int top() {
       if  (empty())
         cerr << "Error in top--stack is empty\n";
       else
         return  (stackPtr[topSub]);
    }
    int  empty() {return  (topSub == -1);}
```

```
}
```

We discuss only a few aspects of this class definition, because it is not necessary to understand all of the details of the code. Objects of the `Stack` class are stack dynamic but include a pointer that references heap-dynamic data. The `Stack` class has three data members—`stackPtr`, `maxLen`, and `topSub`—all of which are private. `stackPtr` is used to reference the heap-dynamic data, which is the array that implements the stack. The class also has four public member functions—`push`, `pop`, `top`, and `empty`—as well as a constructor and a destructor. All of the member function definitions are included in this class, although they could have been externally defined. Because the bodies of the member functions are included, they are all implicitly inlined. The constructor uses the **`new`** operator to allocate an array of 100 **`int`** elements from the heap. It also initializes `maxLen` and `topSub`.

The following is an example program that uses the `Stack` abstract data type:

```
void main() {
  int topOne;
  Stack stk;  //** Create an instance of the Stack class
  stk.push(42);
  stk.push(17);
  topOne = stk.top();
  stk.pop();
  . . .
}
```

Following is a definition of the `Stack` class with only prototypes of the member functions. This code is stored in a header file with the `.h` file name extension. The definitions of the member functions follow the class definition. These use the scope resolution operator, `::`, to indicate the class to which they belong. These definitions are stored in a code file with the file name extension `.cpp`.

```
// Stack.h - the header file for the Stack class
#include <iostream.h>
class Stack {
  private:  //** These members are visible only to other
            //** members and friends (see Section 11.6.3)
    int *stackPtr;
    int maxLen;
    int topSub;
```

```
  public:    //** These members are visible to clients
    Stack();  //** A constructor
    ~Stack();  //** A destructor
    void push(int);
    void pop();
    int top();
    int empty();
}
// Stack.cpp - the implementation file for the Stack class
#include <iostream.h>
#include "Stack.h"
using std::cout;
Stack::Stack() {  //** A constructor
  stackPtr =  new int  [100];
  maxLen = 99;
  topSub = -1;
}
Stack::~Stack() {delete  [] stackPtr;};  //** A destructor
void Stack::push(int  number) {
  if (topSub == maxLen)
    cerr << "Error in push--stack is full\n";
  else  stackPtr[++topSub] = number;
}
void Stack::pop() {
  if (topSub == -1)
    cerr << "Error in pop--stack is empty\n";
  else  topSub--;
}
int top() {
      if  (topSub == -1)
        cerr << "Error in top--stack is empty\n";
      else
        return  (stackPtr[topSub]);
    }
int Stack::empty() {return  (topSub == -1);}
```

11.4.2 Abstract Data Types in Java

Java support for abstract data types is similar to that of C++. There are, however, a few important differences. All objects are allocated from the heap and accessed through reference variables. Methods in Java must be defined

completely in a class. A method body must appear with its corresponding method header.2 Therefore, a Java abstract data type is both declared and defined in a single syntactic unit. A Java compiler can inline any method that is not overridden. Definitions are hidden from clients by declaring them to be private.

2. Java interfaces are an exception to this—an interface has method headers but cannot include their bodies.

One important advantage of Java's classes over the classes of C++ is that it uses implicit garbage collection of all objects. This allows the programmer to ignore the issue of deallocation of objects and the clutter of deallocation code in the implementations of abstract data types.

Rather than having private and public clauses in its class definitions, in Java access modifiers can be attached to method and variable definitions. If an instance variable or method does not have an access modifier, it has package access, which is discussed in Section 11.7.2.

11.4.2.1 An Example

The following is a Java class definition for our stack example:

```
class StackClass {
  private int [] stackRef;
  private int maxLen,
              topIndex;
  public StackClass() {  // A constructor
    stackRef = new int [100];
    maxLen = 99;
    topIndex = -1;
  }
  public void push(int number) {
    if (topIndex == maxLen)
      System.out.println("Error in push–stack is full");
    else stackRef[++topIndex] = number;
  }
  public void pop() {
    if (empty())
      System.out.println("Error in pop–stack is empty");
```

```
    else --topIndex;
  }
  public int top() {
    if (empty()) {
      System.out.println("Error in top–stack is empty");
      return  9999;
    }
    else
      return (stackRef[topIndex]);
  }
  public boolean empty() {return (topIndex == -1);}
}
```

An example class that uses `StackClass` follows:

```
public class TstStack {
  public static void main(String[] args) {
    StackClass myStack = new StackClass();
    myStack.push(42);
    myStack.push(29);
    System.out.println("29 is: " + myStack.top());
    myStack.pop();
    System.out.println("42 is: " + myStack.top());
    myStack.pop();
    myStack.pop(); // Produces an error message
  }
}
```

One obvious difference between the Java and the C++ implementations of the stack is the lack of a destructor in the Java version, obviated by Java's implicit garbage collection.3

3. In Java, the **`finalize`** method serves as a kind of destructor.

11.4.2.2 Evaluation

Although different in some primarily cosmetic ways, Java's support for abstract data types is similar to that of C++. Java clearly provides for what is necessary to design abstract data types.

11.4.3 Abstract Data Types in C#

Recall that C# is based on both C++ and Java and that it also includes some new constructs. Like Java, all C# class instances are heap dynamic. Default constructors, which provide initial values for instance data, are predefined for all classes. These constructors provide typical initial values, such as `0` for **`int`** types and **`false`** for **`boolean`** types. A user can furnish one or more constructors for any class he or she defines. Such constructors can assign initial values to some or all of the instance data of the class. Any instance variable that is not initialized in a user-defined constructor is assigned a value by the default constructor.

Although C# allows destructors to be defined, because it uses garbage collection for most of its heap objects, destructors are rarely used.

11.4.3.1 Encapsulation

C++ includes both classes and structs, which are nearly identical constructs. The only difference is that the default access modifier for class is **`private`**, whereas for structs it is **`public`**. C# also has structs, but they are very different from those of C++. In C#, structs are, in a sense, lightweight classes. They can have constructors, properties, methods, and data fields and can implement interfaces but do not support inheritance. One other important difference between structs and classes in C# is that structs are value types, as opposed to reference types. They are allocated on the run-time stack, rather than the heap. If they are passed as parameters, like other value types, by default they are passed by value. All C# value types, including all of its primitive types, are actually structs. Structs can be created by declaring them, like other predefined value types, such as **`int`** or **`float`**. They can also be created with the **`new`** operator, which calls a constructor to initialize them.

Structs are used in C# primarily to implement relatively small simple types that need never be base types for inheritance. They are also used when it is convenient for the objects of the type to be stack as opposed to heap allocated.

11.4.3.2 Information Hiding

C# uses the **`private`** and **`protected`** access modifiers exactly as they are used in Java.

C# provides properties, which it inherited from Delphi, as a way of implementing getters and setters without requiring explicit method calls by the client. Properties provide implicit access to specific private instance data. For example, consider the following simple class and client code:

```
public class  Weather {
  public int  DegreeDays {  //** DegreeDays is a property
    get {
       return  degreeDays;
    }
     set {
       if(value < 0 || value > 30)
         Console.WriteLine(
              "Value is out of range: {0}",  value);
       else
         degreeDays = value;
    }
  }
  private int degreeDays;
   . . .
  }
. . .
Weather w =  new Weather();
int  degreeDaysToday, oldDegreeDays;
. . .
w.DegreeDays = degreeDaysToday;
. . .
oldDegreeDays = w.DegreeDays;
```

In the class `Weather`, the property `DegreeDays` is defined. This property provides a getter method and a setter method for access to the private data member, `degreeDays`. In the client code following the class definition, `degreeDays` is treated as if it were a public-member variable, although access to it is available through the property only. Notice the use of the implicit variable **`value`** in the setter method. This is the mechanism by which the new value of the property is referenced.

The stack example is not shown here in C#. The only difference between the Java version in Section 11.4.2.1 and the C# version is the output method calls and the use of **bool** instead of **boolean** for the return type of the empty method.

11.4.4 Abstract Data Types in Ruby

Ruby provides support for abstract data types through its classes. In terms of capabilities, Ruby classes are similar to those in C++ and Java.

11.4.4.1 Encapsulation

In Ruby, a class is defined in a compound statement opened with the **class** reserved word and closed with **end**. The names of instance variables have a special syntactic form, they must begin with at signs (@). Instance methods have the same syntax as functions in Ruby: They begin with the **def** reserved word and end with **end**. Class methods are distinguished from instance methods by having the class name appended to the beginning of their names with a period separator. For example, in a class named Stack, a class method's name would begin with Stack. Constructors in Ruby are named initialize. Because the constructor cannot be overloaded, there only can be one per class.

Classes in Ruby are dynamic in the sense that members can be added at any time. This is done by simply including additional class definitions that specify the new members. Moreover, even predefined classes of the language, such as String, can be extended. For example, consider the following class definition:

```
class myClass
  def meth1
    . . .
  end
```

```
end
```

This class could be extended by adding a second method, `meth2`, with a second class definition:

```
class myClass
  def meth2
    . . .
  end
end
```

Methods can also be removed from a class. This is done by providing another class definition in which the method to be removed is sent to the method `remove_method` as a parameter. The dynamic classes of Ruby is another example of a language designer trading readability (and as a consequence, reliability) for flexibility. Allowing dynamic changes to classes clearly adds flexibility to the language, while harming readability. To determine the behavior of a class at a particular point in a program, one must find all of its definitions in the program and consider all of them.

11.4.4.2 Information Hiding

Access control for methods in Ruby is dynamic, so access violations are detected only during execution. The default method access is public, but it can also be protected or private. There are two ways to specify the access control, both of which use functions with the same names as the access levels, **`private`** and **`public`**. One way is to call the appropriate function without parameters. This resets the default access for subsequently defined methods in the class. For example,

```
class MyClass
  def meth1
  . . .
  end
  . . .
private
  def meth7
  . . .
  end
  . . .
```

```
end   # of class MyClass
```

The alternative is to call the access control functions with the names of the specific methods as parameters. For example, the following is semantically equivalent to the previous class definition:

```
class MyClass
  def meth1
  . . .
  end
  . . .
  def meth7
  . . .
  end
  private :meth7, . . .
   end # of class MyClass
```

In Ruby, all data members of a class are private, and that cannot be changed. So, data members can be accessed only by the methods of the class, some of which may be accessor methods. In Ruby, instance data that are accessible through accessor methods are called **attributes**.

For an instance variable named `@sum`, the getter and setter methods would be as follows:

```
def sum
  @sum
end
def sum=(new_sum)
  @sum = new_sum
end
```

Notice that getters are given the name of the instance variable minus the `@`. The names of setter methods are the same as those of the corresponding getters, except they have an equal sign (`=`) attached.

Getters and setters can be implicitly generated by the Ruby system by including calls to `attr_reader` and `attr_writer`, respectively, in the class definition. The parameters to these are the symbols of the attribute’s names, as is illustrated in the following:

```
attr_reader :sum, :total
```

```
attr_writer :sum
```

11.4.4.3 An Example

Following is the stack example written in Ruby:

```
# Stack.rb - defines and tests a stack of maximum length
#             100, implemented in an array
class  StackClass
# Constructor
  def initialize
    @stackRef = Array.new(100)
    @maxLen = 100
    @topIndex = -1
  end
# push method
   def push(number)
     if @topIndex == @maxLen
       puts "Error in push - stack is full"
     else
       @topIndex = @topIndex + 1
       @stackRef[@topIndex] = number
      end
      end
# pop method
  def pop
     if empty
       puts "Error in pop - stack is empty"
     else
       @topIndex = @topIndex - 1
     end
   end
# top method
   def top
     if empty
       puts "Error in top - stack is empty"
     else
       @stackRef[@topIndex]
     end
   end
# empty method
  def empty
    @topIndex == -1
  end
```

```
end  # of Stack class
# Test code for StackClass
myStack = StackClass.new
myStack.push(42)
myStack.push(29)
puts "Top element is (should be 29): #{myStack.top}"
myStack.pop
puts "Top element is (should be 42): #{myStack.top}"
myStack.pop
# The following pop should produce an
# error message - stack is empty
myStack.pop
```

Recall that the notation `#{variable}` converts the value of the variable to a string, which is then inserted into the string in which it appears. This class defines a stack structure that can store objects of any type.

11.4.4.4 Evaluation

Recall that in Ruby, everything is an object and arrays are actually arrays of references to objects. That clearly makes this stack more flexible than the similar examples in C++ and Java. Furthermore, simply by passing the desired maximum length to the constructor, objects of this class could have any given maximum length. Of course, because arrays in Ruby have dynamic length, the class could be modified to implement stack objects that are not restricted to any length, except that imposed by the machine’s memory capacity. Because the names of class and instance variables have different forms, Ruby has a slight readability advantage over the other languages discussed in this section.

11.5 Parameterized Abstract Data Types

It is often convenient to be able to parameterize abstract data types. For example, we should be able to design a stack abstract data type that can store any scalar type elements rather than be required to write a separate stack abstraction for every different scalar type. Note that this is only an issue for static typed languages. In a dynamic typed language like Ruby, any stack implicitly can store any type elements. In fact, different elements of the stack could be of different types. In the following three subsections, the capabilities of C++, Java 5.0, and C# 2005 to construct parameterized abstract data types are discussed.

11.5.1 C++

To make the example C++ stack class of Section 11.4.1 generic in the stack size, only the constructor function needs to be changed, as in the following:

```
Stack(int size) {
  stackPtr = new int  [size];
  maxLen = size - 1;
  topSub = -1;
}
```

The declaration for a stack object now may appear as follows:

```
Stack stk(150);
```

The class definition for `Stack` can include both constructors, so users can use the default-size stack or specify some other size.

The element type of the stack can be made generic by making the class a templated class. Then, the element type can be a template parameter. The definition of the templated class for a stack type is as follows:

```
#include <iostream.h>
template <typename Type>  // Type is the template parameter
class Stack {
  private:
    Type *stackPtr;
    int maxLen;
    int topSub;
  public:
// A constructor for 100 element stacks
    Stack() {
      stackPtr = new Type [100];
      maxLen = 99;
      topSub = -1;
    }
// A constructor for a given number of elements
    Stack(int size) {
      stackPtr = new Type [size];
      maxLen = size - 1;
      topSub = -1;
    }
    ~Stack() {delete  stackPtr;};  // A destructor
    void push(Type number) {
      if (topSub == maxLen)
        cout << "Error in push–stack is full\n";
      else  stackPtr[++ topSub] = number;
    }
    void pop() {
      if (empty())
        cout << "Error in pop–stack is empty\n";
      else  topSub --;
    }
    Type top() {
      if (empty())
       cerr << "Error in top--stack is empty\n";
      else
        return  (stackPtr[topSub]);
    }
    int  empty() {return  (topSub == -1);}
}
```

C++ templated classes are instantiated to become typed classes at compile time. For example, an instance of the templated `Stack` class, as well as an instance of the typed class, can be created with the following declaration:

```
Stack<int> myIntStack;
```

However, if an instance of the templated `Stack` class has already been created for the **`int`** type, the typed class need not be created.

11.5.2 Java 5.0

Java 5.0 supports a form of parameterized abstract data types in which the generic parameters must be classes. Recall that these were briefly discussed in Chapter 9.

The most common generic types are collection types, such as `LinkedList` and `ArrayList`, which were in the Java class library before support for generics was added. The original collection types stored `Object` class instances, so they could store any objects (but not primitive types). Therefore, the collection types have always been able to store multiple types (as long as they are classes). There were three issues with this: First, every time an object was removed from the collection, it had to be cast to the appropriate type. Second, there was no error checking when elements were added to the collection. This meant that once the collection was created, objects of any class could be added to the collection, even if the collection was meant to store only `Integer` objects. Third, the collection types could not store primitive types. So, to store **`int`** values in an `ArrayList`, the value first had to be put in an `Integer` class instance. For example, consider the following code:

```
//* Create an ArrayList object
ArrayList myArray = new ArrayList();
//* Create an element
myArray.add(0, new Integer(47));
//* Get first object
Integer myInt = (Integer)myArray.get(0);
```

In Java 5.0, the collection classes, the most commonly used of which is `ArrayList`, became a generic class. Such classes are instantiated by calling **`new`** on the class constructor and passing it the generic parameter in pointed - brackets. For example, the `ArrayList` class can be instantiated to store `Integer` objects with the following statement:

```
ArrayList <Integer> myArray = new ArrayList <Integer>();
```

This new class overcomes two of the problems with pre-Java 5.0 collections. Only `Integer` objects can be put into the `myArray` collection. Furthermore, there is no need to cast an object being removed from the collection.

Java 5.0 also includes generic collections for linked-lists, queues, and sets. Users also can define generic classes in Java 5.0. For example, we could have the following:

```
public class MyClass<T> {
  . . .
}
```

This class could be instantiated with the following:

```
MyClass<String> myString;
```

There are some drawbacks to these user-defined generic classes. For one thing, they cannot store primitives. Second, the elements cannot be indexed. Elements must be added to user-defined generic collections with the `add` method. Next, we implement the generic stack example using an `ArrayList`. Note that the last element of an `ArrayList` is found using the `size` method, which returns the number of elements in the structure. Elements are deleted from the structure with the `remove` method. Following is the generic class:

```
import  java.util.*;
public class  Stack2<T> {
  private ArrayList<T> stackRef;
  private int maxLen;
  public Stack2() { // A constructor
    stackRef = new ArrayList<T> ();
    maxLen = 99;
  }
  public void push(T newValue) {
    if (stackRef.size() == maxLen)
      System.out.println("Error in push–stack is full");
    else
      stackRef.add(newValue);
  }
  public void pop() {
    if (empty())
      System.out.println("Error in pop–stack is empty");
    else
     stackRef.remove(stackRef.size() - 1);
```

```
  }
  public T top() {
    if empty()) {
      System.out.println("Error in top–stack is empty");
      return null;  }
    else
      return (stackRef.get(stackRef.size() - 1));
  }
  public boolean empty() {return (stackRef.isEmpty());}
```

This class could be instantiated for the `String` type with the following:

```
Stack2<String> myStack = new Stack2<String>();
```

Recall from Chapter 9, that Java 5.0 supports wildcard classes. For example, `Collection<?>` is a wildcard class for all collection classes. This allows a method to be written that can accept any collection type as a parameter. Because a collection can itself be generic, the `Collection<?>` class is in a sense a generic of a generic class.

Some care must be taken with objects of the wildcard type. For example, because the components of a particular object of this type have a type, other type objects cannot be added to the collection. For example, consider

```
Collection<?> c = new ArrayList<String>();
```

It would be illegal to use the `add` method to put something into this collection unless its type were `String`.

A generic class can easily be defined in Java 5.0 that will work only for a restricted set of types. For example, a class can declare a variable of the generic type and call a method such as `compareTo` through that variable. If the class is instantiated for a type that does not include a `compareTo` method, the class cannot be used. To prevent a generic class from being instantiated for a type that does not support `compareTo`, it could be defined with the following generic parameter:

```
<T extends Comparable>
```

`Comparable` is the interface in which `compareTo` is declared. If this generic type is used on a class definition, the class cannot be instantiated for any type

that does not implement `Comparable`. The choice of the reserved word **`extends`** seems odd here, but its use is related to the concept of a subtype. Apparently, the designers of Java did not want to add another more connotative reserved word to the language.

11.5.3 C# 2005

As was the case with Java, the first version of C# defined collection classes that stored objects of any class. These were `ArrayList`, `Stack`, and `Queue`. These classes had the same problems as the collection classes of pre-Java 5.0.

Generic classes were added to C# in its 2005 version. The five predefined generic collections are `Array`, `List`, `Stack`, `Queue`, and `Dictionary` (the `Dictionary` class implements hashes). Exactly as in Java 5.0, these classes eliminate the problems of allowing mixed types in collections and requiring casts when objects are removed from the collections.

As with Java 5.0, users can define generic classes in C# 2005. One capability of the user-defined C# generic collections is that any of them can be defined to allow its elements to be indexed (accessed through subscripting). Although the indexes are usually integers, an alternative is to use strings as indexes.

One capability that Java 5.0 provides that C# 2005 does not is wildcard classes.

11.6 Encapsulation Constructs

The first five sections of this chapter discussed abstract data types, which are minimal encapsulations. This section describes the multiple-type encapsulations that are needed for larger programs.

11.6.1 Introduction

When the size of a program reaches beyond a few thousand lines, two practical problems become evident. From the programmer's point of view, having such a program appear as a single collection of subprograms or abstract data type definitions does not impose an adequate level of organization on the program to keep it intellectually manageable. The second practical problem for larger programs is recompilation. For relatively small programs, recompiling the whole program after each modification is not costly. But for large programs, the cost of recompilation is significant. So, there is an obvious need to find ways to avoid recompilation of the parts of a program that are not affected by a change. The obvious solution to both of these problems is to organize programs into collections of logically related code and data, each of which can be compiled without recompilation of the rest of the program. An **encapsulation** is such a collection.

Encapsulations are often placed in libraries and made available for reuse in programs other than those for which they were written. People have been writing programs with more than a few thousand lines for at least the last 50 years, so techniques for providing encapsulations have been evolving for some time.

In languages that allow nested subprograms, programs can be organized by nesting subprogram definitions inside the logically larger subprograms that use them. This can be done in Python and Ruby. As discussed in Chapter 5, however, this method of organizing programs, which uses static scoping, is far from ideal. Therefore, even in languages that allow nested subprograms,

they are not used as a primary organizing encapsulation construct.

11.6.2 Encapsulation in C

C does not provide complete support for abstract data types, although both abstract data types and multiple-type encapsulations can be simulated.

In C, a collection of related functions and data definitions can be placed in a file, which can be independently compiled. Such a file, which acts as a library, has an implementation of its entities. The interface to such a file, including data, type, and function declarations, is placed in a separate file called a **header file**. Type representations can be hidden by declaring them in the header file as pointers to struct types. The complete definitions of such struct types need only appear in the implementation file.

The header file, in source form, and the compiled version of the implementation file are furnished to clients. When such a library is used, the header file is included in the client code, using an `#include` preprocessor specification, so that references to functions and data in the client code can be type checked. The `#include` specification also documents the fact that the client program depends on the library implementation file. This approach effectively separates the specification and implementation of an encapsulation.

Although these encapsulations work, they create some insecurities. For example, a user could simply cut and paste the definitions from the header file into the client program, rather than using `#include`. This would work, because `#include` simply copies the contents of its operand file into the file in which the `#include` appears. However, there are two problems with this approach. First, the documentation of the dependence of the client program on the library (and its header file) is lost. Second, suppose a user copies the header file into his or her program. Then the author of the library changes both the header and the implementation files. Following this, the user uses the new implementation file with the old header. For example, a variable `x` could have been defined to be **`int`** type in the old header file, which the client code still uses, although the implementation code has been recompiled with

the new header file, which defines `x` to be **`float`**. So, the implementation code was compiled with `x` as an **`int`** but the client code was compiled with `x` as a **`float`**. The linker does not detect this error.

Thus, it is the user's responsibility to ensure that both the header and implementation files are up-to-date. This is often done with a `make` utility.

One other drawback of this approach is the inherent problems of pointers and the potential confusion with assignment and comparisons of pointers.

11.6.3 Encapsulation in C++

C++ provides two different kinds of encapsulation: header and implementation files can be defined as in C, or class headers and definitions can be defined. Because of the complex interplay of C++ templates and separate compilation, the header files of C++ template libraries often include complete definitions of resources, rather than just data declarations and subprogram protocols; this is due in part to the use of the C linker for C++ programs.

When nontemplated classes are used for encapsulations, the class header file has only the prototypes of the member functions, with the function - definitions provided outside the class in a code file, as in the last example in Section 11.4.1.4. This clearly separates interface from implementation.

One language design problem that results from having classes but no generalized encapsulation construct is that sometimes when operations are defined that use two different classes of objects, the operation does not naturally belong in either class. For example, suppose we have an abstract data type for matrices and one for vectors and need a multiplication operation between a vector and a matrix. The multiplication code must have access to the data members of both the vector and the matrix classes, but neither of those classes is the natural home for the code. Furthermore, regardless of which is chosen, access to the members of the other is a problem. In C++, these kinds of situations can be handled by allowing nonmember functions to be "friends" of a class. Friend functions have access to the private entities of

the class where they are declared to be friends. For the matrix/vector multiplication operation, one C++ solution is to define the operation outside both the matrix and the vector classes but define it to be a friend of both. The following skeletal code illustrates this scenario:

```
class Matrix;  //** A class declaration
class Vector {
  friend Vector multiply(const Matrix&, const Vector&);
  . . .
};
class  Matrix {  //** The class definition
  friend Vector multiply(const Matrix&, const Vector&);
  . . .
};
//** The function that uses both Matrix and Vector objects
Vector multiply(const Matrix& m1, const Vector& v1) {
  . . .
}
```

In addition to functions, whole classes can be defined to be friends of a class; then all the private members of the class are visible to all of the members of the friend class.

11.6.4 C# Assemblies

C# includes a larger encapsulation construct than a class. The construct is the one used by all of the .NET programming languages: the assembly. Assemblies are built by .NET compilers. A .NET application consists of one or more assemblies. An **assembly** is a file[4] that appears to application programs to be a single dynamic link library (`.dll`)[5] or an executable (`.exe`). An assembly defines a module, which can be separately developed. An assembly includes several different components. One of the primary components of an assembly is its programming code, which is in an intermediate language, having been compiled from its source language. In .NET, the intermediate language is named Common Intermediate Language (CIL). It is used by all .NET languages. Because its code is in CIL, an assembly can be used on any architecture, device, or operating system. When executed, the CIL is just-in-time compiled to native code for the architecture on which it is resident.

4. An assembly can consist of any number of files.

5. A **dynamic link library (DLL)** is a collection of classes and methods that are individually linked to an executing program when needed during execution. Therefore, although a program has access to all of the resources in a particular DLL, only the parts that are actually used are ever loaded and linked to the program. DLLs have been part of the Windows programming environment since Windows first appeared. However, the DLLs of .NET are quite different from those of previous Windows systems.

In addition to the CIL code, a .NET assembly includes metadata that describes every class it defines, as well as all external classes it uses. An assembly also includes a list of all assemblies referenced in the assembly and an assembly version number.

In the .NET world, the assembly is the basic unit of deployment of software. Assemblies can be private, in which case they are available to just one application, or public, which means any application can use them.

As mentioned previously, C# has an access modifier, **`internal`**. An **`internal`** member of a class is visible to all classes in the assembly in which it appears.

Java has a file structure that is similar to an assembly called a Java Archive (JAR). It is also used for deployment of Java software systems. JARs are built with the Java utility `jar`, rather than a compiler.

11.7 Naming Encapsulations

We have considered encapsulations to be syntactic containers for logically related software resources—in particular, abstract data types. The purpose of these encapsulations is to provide a way to organize programs into logical units for compilation. This allows parts of programs to be recompiled after isolated changes. There is another kind of encapsulation that is necessary for constructing large programs: a naming encapsulation.

A large program is usually written by many developers, working somewhat independently, perhaps even in different geographic locations. This requires the logical units of the program to be independent, while still able to work together. It also creates a naming problem: How can independently working developers create names for their variables, methods, and classes without accidentally using names already in use by some other programmer developing a different part of the same software system?

Libraries are the origin of the same kind of naming problems. Over the past two decades, large software systems have become progressively more dependent on libraries of supporting software. Nearly all software written in contemporary programming languages requires the use of large and complex standard libraries, in addition to application-specific libraries. This widespread use of multiple libraries has necessitated new mechanisms for managing names. For example, when a developer adds new names to an existing library or creates a new library, he or she must not use a new name that conflicts with a name already defined in a client's application program or in some other library the program uses. Without some language processor assistance, this is virtually impossible, because there is no convenient way for the library author to know what names a client's program uses or what names are defined by the other libraries the client program might use.

Naming encapsulations define name scopes that assist in avoiding these name conflicts. Each library can create its own naming encapsulation to prevent its names from conflicting with the names defined in other libraries or in client code. Each logical part of a software system can create a naming

encapsulation with the same purpose.

Naming encapsulations are logical encapsulations, in the sense that they need not be physically contiguous. Several different collections of code can be placed in the same namespace, even though they are stored in different places. In the following sections, we briefly describe the uses of naming encapsulations in C++, Java, and Ruby.

11.7.1 C++ Namespaces

C++ includes a specification, **`namespace`**, that helps programs manage the problem of global namespaces. One can place each library in its own namespace and qualify the names in the program with the name of the namespace when the names are used outside that namespace. For example, suppose there is an abstract data type header file that implements stacks. If there is concern that some other library file may define a name that is used in the stack abstract data type, the file that defines the stack could be placed in its own namespace. This is done by placing all of the declarations for the stack in a namespace block, as in the following:

```
namespace myStackSpace {
  // Stack declarations
}
```

The implementation file for the stack abstract data type could reference the names declared in the header file with the scope resolution operator, `::`, as in

```
myStackSpace::topSub
```

The implementation file could also appear in a namespace block specification identical to the one used on the header file, which would make all of the names declared in the header file directly visible. This is definitely simpler, but slightly less readable, because it is less obvious where a specific name in the implementation file is declared.

Client code can gain access to the names in the namespace of the header file of a library in three different ways. One way is to qualify the names from the

library with the name of the namespace. For example, a reference to the variable `topSub` could appear as follows:

```
myStackSpace::topSub
```

This is exactly the way the implementation code could reference it if the implementation file was not in the same namespace.

The other two approaches use the **`using`** directive. This directive can be used to qualify individual names from a namespace, as with

```
using myStackSpace::topSub;
```

which makes `topSub` visible, but not any other names from the `myStackSpace` namespace.

The **`using`** directive also can be used to qualify all of the names from a namespace, as in the following:

```
using namespace myStackSpace;
```

Code that includes this directive can directly access the names defined in the namespace, as in

```
p = topSub;
```

Be aware that namespaces are a complicated feature of C++, and we have introduced only the simplest part of the story here.

C# includes namespaces that are much like those of C++.

11.7.2 Java Packages

Java includes a naming encapsulation construct: the package. Packages can contain more than one type[6] definition, and the types in a package are partial friends of one another. *Partial* here means that the entities defined in a type in a package that either are public or protected (see Chapter 12) or have no access specifier are visible to all other types in the package.

6. By type here we mean either a class or an interface.

Entities without access modifiers are said to have **package scope**, because they are visible throughout the package. Java therefore has less need for explicit friend declarations and does not include the friend functions or friend classes of C++.

The resources defined in a file are specified to be in a particular package with a package declaration, as in

```
package stkpkg;
```

The package declaration must appear as the first line of the file. The resources of every file that does not include a package declaration are implicitly placed in the same unnamed package.

The clients of a package can reference the types defined in the package using fully qualified names. For example, if the package `stkpkg` has a class named `myStack`, that class can be referenced in a client of `stkpkg` as `stkpkg.myStack`. Likewise, a variable in the `myStack` object named `topSub` could be referenced as `stkpkg.myStack.topSub`. Because this approach can quickly become cumbersome when packages are nested, Java provides the **`import`** declaration, which allows shorter references to type names defined in a package. For example, suppose the client includes the following:

```
import stkpkg.myStack;
```

Now, the class `myStack` can be referenced by just its name. To be able to access all of the type names in the package, an asterisk can be used on the import statement in place of the type name. For example, if we wanted to import all of the types in `stkpkg`, we could use the following:

```
import stkpkg.*;
```

Note that Java's **`import`** is only an abbreviation mechanism. No otherwise hidden external resources are made available with **`import`**. In fact, in Java nothing is implicitly hidden if it can be found by the compiler or class loader (using the package name and the `CLASSPATH` environment variable).

Java's **import** documents the dependencies of the package in which it appears on the packages named in the **import**. These dependencies are less obvious when **import** is not used.

11.7.3 Ruby Modules

Ruby classes serve as namespace encapsulations, as do the classes of other languages that support object-oriented programming. Ruby has an additional naming encapsulation, called a **module**. Modules typically define collections of methods and constants. So, modules are convenient for encapsulating libraries of related methods and constants, whose names are in a separate namespace so there are no name conflicts with other names in a program that uses the module. Modules are unlike classes in that they cannot be instantiated or subclassed and do not define variables. Methods that are defined in a module include the module's name in their names. For example, consider the following skeletal module definition:

```
module MyStuff
  PI = 3.14159265
  def MyStuff.mymethod1(p1)
  . . .
  end
  def MyStuff.mymethod2(p2)
  . . .
  end
end
```

Assuming the `MyStuff` module is stored in its own file, a program that wants to use the constant and methods of `MyStuff` must first gain access to the module. This is done with the `require` method, which takes the file name in the form of a string literal as a parameter. Then, the constants and methods of the module can be accessed through the module's name. Consider the following code that uses our example module, `MyStuff`, which is stored in the file named `myStuffMod`:

```
  require 'myStuffMod'
  . . .
  MyStuff.mymethod1(x)
  . . .
```

Modules are further discussed in Chapter 12.

SUMMARY

The concept of abstract data types, and their use in program design, was a milestone in the development of programming as an engineering discipline. Although the concept is relatively simple, its use did not become convenient and safe until languages were designed to support it.

The two primary features of abstract data types are the packaging of data objects with their associated operations and information hiding. A language may support abstract data types directly or simulate them with more general encapsulations.

C++ data abstraction is provided by classes. Classes are types, and instances can be either stack or heap dynamic. A member function (method) can have its complete definition appear in the class or have only the protocol given in the class and the definition placed in another file, which can be separately compiled. C++ classes can have two clauses, each prefixed with an access modifier: private or public. Both constructors and destructors can be given in class definitions. Heap-allocated objects must be explicitly deallocated with **`delete`**.

Java data abstractions are similar to those of C++, except all Java objects are allocated from the heap and are accessed through reference variables. Also, all objects are garbage collected. Rather than having access modifiers attached to clauses, in Java the modifiers appear on individual declarations (or definitions).

C# supports abstract data types with both classes and structs. Its structs are value types and do not support inheritance. C# classes are similar to those of Java.

Ruby supports abstract data types with its classes. Ruby's classes differ from those of most other languages in that they are dynamic—members can be added, deleted, or changed during execution.

C++, Java 5.0, and C# 2005 allow their abstract data types to be -parameterized—Ada through its generic packages, C++ through its templated classes, and Java 5.0 and C# through their collection classes and interfaces and user-defined generic classes.

To support the construction of large programs, some contemporary languages include multiple-type encapsulation constructs, which can contain a collection of logically related types. An encapsulation may also provide access control to its entities. Encapsulations provide the programmer with a method of organizing programs that also facilitates recompilation.

C++, C#, Java, and Ruby provide naming encapsulations. For Ada and Java, they are named packages; for C++ and C#, they are namespaces; for Ruby, they are modules. Partially because of the availability of packages, Java does not have friend functions or friend classes.

REVIEW QUESTIONS

1. What are the two kinds of abstractions in programming languages?
2. Define *abstract data type*.
3. What are the advantages of the two parts of the definition of *abstract data type*?
4. What are the language design requirements for a language that supports abstract data types?
5. What are the language design issues for abstract data types?
6. From where are C++ objects allocated?
7. In what different places can the definition of a C++ member function appear?
8. What is the purpose of a C++ constructor?
9. What are the legal return types of a constructor?
10. Where are all Java methods defined?
11. How are C++ class instances created?
12. From where are Java class instances allocated?
13. Why does Java not have destructors?
14. Where are all Java methods defined?
15. Where are Java classes allocated?
16. Why are destructors not as frequently needed in Java as they are in C++?

17. What is a friend function? What is a friend class?

18. What is one reason Java does not have friend functions or friend classes?

19. Describe the fundamental differences between C# structs and its classes.

20. How is a struct object in C# created?

21. Explain the three reasons accessors to private types are better than making the types public.

22. What are the differences between a C++ struct and a C# struct?

23. What is the name of all Ruby constructors?

24. What is the fundamental difference between the classes of Ruby and those of C++ and Java?

25. How are instances of C++ template classes created?

26. Describe the two problems that appear in the construction of large programs that led to the development of encapsulation constructs.

27. What problems can occur using C to define abstract data types?

28. What is a C++ namespace, and what is its purpose?

29. What is a Java package, and what is its purpose?

30. Describe a .NET assembly.

31. What elements can appear in a Ruby module?

PROBLEM SET

1. Some software engineers believe that all imported entities should be qualified by the name of the exporting program unit. Do you agree? Support your answer.

2. Suppose someone designed a stack abstract data type in which the function `top` returned an access path (or pointer) rather than returning a copy of the top element. This is not a true data abstraction. Why? Give an example that illustrates the problem.

3. Write an analysis of the similarities of and differences between Java packages and C++ namespaces.

4. Discuss the advantages of C# properties, relative to writing accessor methods in C++ or Java.

5. Explain the dangers of C's approach to encapsulation.

6. Why didn't C++ eliminate the problems discussed in Problem 5?

7. Why are destructors rarely used in Java but essential in C++?

8. What are the arguments for and against the C++ policy on inlining of methods?

9. Describe a situation where a C# struct is preferable to a C# class.

10. Explain why naming encapsulations are important for developing large programs.

11. Describe the three ways a client can reference a name from a namespace in C++.

12. The namespace of the C# standard library, `System`, is not implicitly available to C# programs. Do you think this is a good idea? Defend your

answer.

13. What are the advantages and disadvantages of the ability to change objects in Ruby?

14. Compare Java’s packages with Ruby’s modules.

PROGRAMMING EXERCISES

1. Design an abstract data type for a matrix with integer elements in a language that you know, including operations for addition, subtraction, and matrix multiplication.

2. Design a queue abstract data type for float elements in a language that you know, including operations for enqueue, dequeue, and empty. The dequeue operation removes the element and returns its value.

3. Modify the C++ class for the abstract stack type shown in Section 11.4.1 to use a linked list representation and test it with the same code that appears in this chapter.

4. Modify the Java class for the abstract stack type shown in Section 11.4.2 to use a linked list representation and test it with the same code that appears in this chapter.

5. Write an abstract data type for complex numbers, including operations for addition, subtraction, multiplication, division, extraction of each of the parts of a complex number, and construction of a complex number from two floating-point constants, variables, or expressions. Use C++, Java, C#, or Ruby.

6. Write an abstract data type for queues whose elements store 10-character names. The queue elements must be dynamically allocated from the heap. Queue operations are enqueue, dequeue, and empty. Use either C++, Java, C#, or Ruby.

7. Write an abstract data type for a queue whose elements can be any primitive type. Use Java 5.0, C# 2005, or C++.

8. Write an abstract data type for a queue whose elements include both a 20-character string and an integer priority. This queue must have the following methods: enqueue, which takes a string and an integer as parameters; dequeue, which returns the string from the queue that has

the highest priority; and empty. The queue is not to be maintained in priority order of its elements, so the dequeue operation must always search the whole queue.

9. A deque is a double-ended queue, with operations adding and removing elements from either end. Modify the solution to Programming Exercise 7 to implement a deque.

10. Write an abstract data type for rational numbers (a numerator and a denominator). Include a constructor and methods for getting the numerator, getting the denominator, addition, subtraction, multiplication, division, equality testing, and display. Use Java, C#, C++, or Ruby.

12 Support for Object-Oriented Programming

This chapter begins with a brief introduction to object-oriented programming, followed by an extended discussion of the primary design issues for inheritance and dynamic binding. Next, the support for object-oriented programming in Smalltalk, C++, Java, C#, and Ruby is discussed. Following this is a short overview of the implementation of dynamic bindings of method calls to methods in object-oriented languages. The last section discusses reflection.

12.1 Introduction

Languages that support object-oriented programming now are firmly entrenched in the mainstream. From COBOL to LISP, including virtually every language in between, dialects that support object-oriented programming have appeared. C++ supports procedural and data-oriented programming, in addition to object-oriented programming. CLOS, an object-oriented version of LISP (Paepeke, 1993), also supports functional programming. Some of the newer languages that were designed to support object-oriented programming do not support other programming paradigms but still employ some of the basic imperative structures and have the appearance of the older imperative languages. Among these are Java and C#. Ruby is challenging to categorize: It is a pure object-oriented language in the sense that all data are objects, but it is a hybrid language in that one can use it for procedural programming. Finally, there is the pure object-oriented but somewhat unconventional language: Smalltalk. Smalltalk was the first language to offer complete support for object-oriented programming. The details of support for object-oriented programming vary widely among languages, and that is the primary topic of this chapter.

This chapter relies heavily on Chapter 11. It is in fact a continuation of that chapter. This relationship reflects the reality that object-oriented programming is, in essence, an application of the principle of abstraction to abstract data types. Specifically, in object-oriented programming, the commonality of a collection of similar abstract data types is factored out and put in a new type. The members of the collection inherit these common parts from that new type. This feature is inheritance, which is at the center of object-oriented programming and the languages that support it.

The other characterizing feature of object-oriented programming, dynamic binding of method calls to methods, is also extensively discussed in this chapter.

Although object-oriented programming is supported by some of the functional languages, for example, CLOS, OCaml, and F#, those languages

are not discussed in this chapter.

12.2 Object-Oriented Programming

12.2.1 Introduction

The concept of **object-oriented programming** had its roots in SIMULA 67 but was not fully developed1 until the evolution of Smalltalk resulted in Smalltalk 80 (in 1980, of course). Indeed, some consider Smalltalk to be the base model for a purely object-oriented programming language. A language that is object oriented must provide support for three key language features: abstract data types, inheritance, and dynamic binding of method calls to methods. Abstract data types were discussed in detail in Chapter 11, so this chapter focuses on inheritance and dynamic binding.

1. Although SIMULA 67 had classes, the members defined inside them were not hidden from outside code.

12.2.2 Inheritance

There has long been pressure on software developers to increase their productivity. This pressure has been intensified by the continuing reduction in the cost of computer hardware. By the middle to late 1980s, it became apparent to many software developers that one of the most promising opportunities for increased productivity in their profession was in software reuse. Abstract data types, with their encapsulation and access controls, are obvious candidates for reuse. The problem with the reuse of abstract data types is that, in nearly all cases, the features and capabilities of the existing type are not quite right for the new use. The old type requires at least some minor modifications. Such modifications can be difficult, because they require the person doing the modification to understand part, if not all, of the existing code. In many cases, the person doing the modification is not the program's original author. Furthermore, the modifications often require changes to all client programs.

A second problem with programming with abstract data types is that the type definitions are all independent and are at the same level.2 This design often makes it impossible to organize a program to match the problem space being addressed by the program. In many cases, the underlying problem has categories of objects that are related, both as siblings (being similar to each other) and as parents and children (having a descendant relationship).

2. This is similar to the functions in a C program, which are also independent and at a single level.

Inheritance offers a solution to both the modification problem posed by abstract data type reuse and the program organization problem. If a new abstract data type can inherit the data and functionality of some existing type, and is also allowed to modify some of those entities and add new entities, reuse is greatly facilitated without requiring changes to the reused abstract data type. Programmers can begin with an existing abstract data type and design a modified descendant of it to fit a new problem requirement. Furthermore, inheritance provides a framework for the definition of hierarchies of related classes that can reflect the descendant relationships in the problem space.

The abstract data types in object-oriented languages, following the lead of SIMULA 67, are usually called **classes**. As with instances of abstract data types, class instances are called **objects**. A class that is defined through inheritance from another class is a **derived class**, a **subclass**, or a **child class**. A class from which the new class is derived is its **base class**, **superclass**, or **parent class**. The subprograms that define the operations on objects of a class are called **methods**. The calls to methods are sometimes called **messages**. The entire collection of methods of a class is called the **message protocol**, or **message interface**, of the class. Computations in an object-oriented program are specified by messages sent from objects to other objects, or in some cases, to classes.

Methods are similar to subprograms. Both are collections of code that perform some computation. Both can take parameters and return results.

Passing a message is different from calling a subprogram. A subprogram typically processes data that is either passed to it by its caller as a parameter

or is accessed nonlocally or globally. A message that is sent to an object is a request to execute one of its methods. At least some of the data on which the method is to operate is part of the object itself. Objects have methods that define processes the object can perform on itself. Because the objects are of abstract data types, these should be the only ways to manipulate data of the object. A subprogram defines a process that it can perform on any data sent to it (or made available nonlocally or globally).

As a simple example of inheritance, consider the following: Suppose we have a class named `Vehicles`, which has variables for year, color, and make. A natural specialization, or subclass, of this would be `Truck`, which could inherit the variables from `Vehicle`, but would add variables for hauling capacity and number of wheels. Figure 12.1 shows a simple diagram to indicate the relationship between the `Vehicle` class and the `Truck` class, in which the arrow points to the parent class.

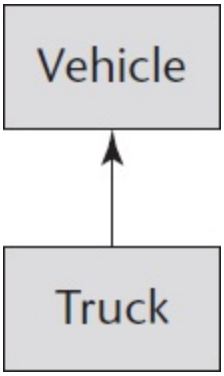


Figure 12.1 A simple example of inheritance

There are several ways a derived class can differ from its parent.3 Following are the most common differences between a parent class and its subclasses:

3. If a subclass does not differ from its parent, it obviously serves no purpose.

1. The subclass can add variables and/or methods to those inherited from the parent class.

2. The subclass can modify the behavior of one or more of its inherited methods. A modified method has the same name, and often the same protocol, as the one of which it is a modification.

3. The parent class can define some of its variables or methods to have private access, which means they will not be visible in the subclass.

The new method is said to **override** the inherited method, which is then called an **overridden method**. The purpose of an overriding method is to provide an operation in the subclass that is similar to one in the parent class, but is customized for objects of the subclass. For example, a parent class, `Bird`, might have a `draw` method that draws a generic bird. A subclass of `Bird` named `Waterfowl` could override the `draw` method inherited from `Bird` to draw a generic waterfowl, perhaps a duck.

Classes can have two kinds of methods and two kinds of variables. The most commonly used methods and variables are called **instance methods** and **instance variables**. Every object of a class has its own set of instance variables, which store the object's state. The only difference between two objects of the same class is the state of their instance variables.4 For example, a class for cars might have instance variables for color, make, model, and year. Instance methods operate only on the objects of the class. **Class variables** belong to the class, rather than their objects, so there is only one copy for the class. For example, if we wanted to count the number of instances of a class, the counter could not be an instance variable—it would need to be a class variable. **Class methods** can perform operations on the class, and possibly also on the objects of the class. They can be called by prefixing their names with either the class name or a variable that references one of their instances. If a class defines a class method, that method can be called even if there are no instances of the class. A class method could be used to create an instance of the class.

4. This is not true in Ruby, which allows different objects of the same class to differ in other ways.

If a new class is a subclass of a single parent class, then the derivation process is called **single inheritance**. If a class has more than one parent class, the process is called **multiple inheritance**. When a number of classes are

related through single inheritance, their relationships to each other can be shown in a derivation tree. The class relationships in a multiple inheritance can be shown in a derivation graph. This is shown in Figure 12.5 in Section 12.4.2.2.

One disadvantage of inheritance as a means of increasing the possibility of reuse is that it creates dependencies among the classes in an inheritance hierarchy. This result works against one of the advantages of abstract data types, which is that they are independent of each other. Of course, not all abstract data types must be completely independent. But in general, the independence of abstract data types is one of their strongest positive characteristics. However, it may be difficult, if not impossible, to increase the reusability of abstract data types without creating dependencies among some of them. Furthermore, in many cases, the dependencies naturally mirror dependencies in the underlying problem space.

In Chapter 11 the access controls for variables and methods, together often called members, in a class are discussed. Private members are visible inside the class, while public members also are visible to clients of the class. Inheritance brings a new category of possible visibility, subclasses. Private members of a base class are not visible to subclasses, but public members are. The third level of accessibility, protected, allows members of a base class to be visible to subclasses, but not clients.

12.2.3 Dynamic Binding

The third essential characteristic (after abstract data types and inheritance) of object-oriented programming languages is a kind of polymorphism[5] provided by the dynamic binding of messages to method definitions. This is sometimes called **dynamic dispatch**. Consider the following situation: There is a base class, `A`, that defines a method `draw` that draws some figure associated with the base class. A second class, `B`, is defined as a subclass of `A`. Objects of this new class also need a `draw` method that is like that provided by `A` but a bit different because the subclass objects are slightly different. So, the subclass overrides the inherited `draw` method. If a client of `A` and `B` has a variable that is a reference to class `A`'s objects, that reference also could point at class `B`'s

objects, making it a **polymorphic** reference. If the method `draw`, which is defined in both classes, is called through the polymorphic reference, the run-time system must determine, during execution, which method should be called, `A`'s or `B`'s (by determining which type object is currently referenced by the reference).6 Figure 12.2 shows this situation.

6. Dynamic binding of method calls to methods is sometimes called *dynamic polymorphism*.

5. Polymorphism is defined in Chapter 9.

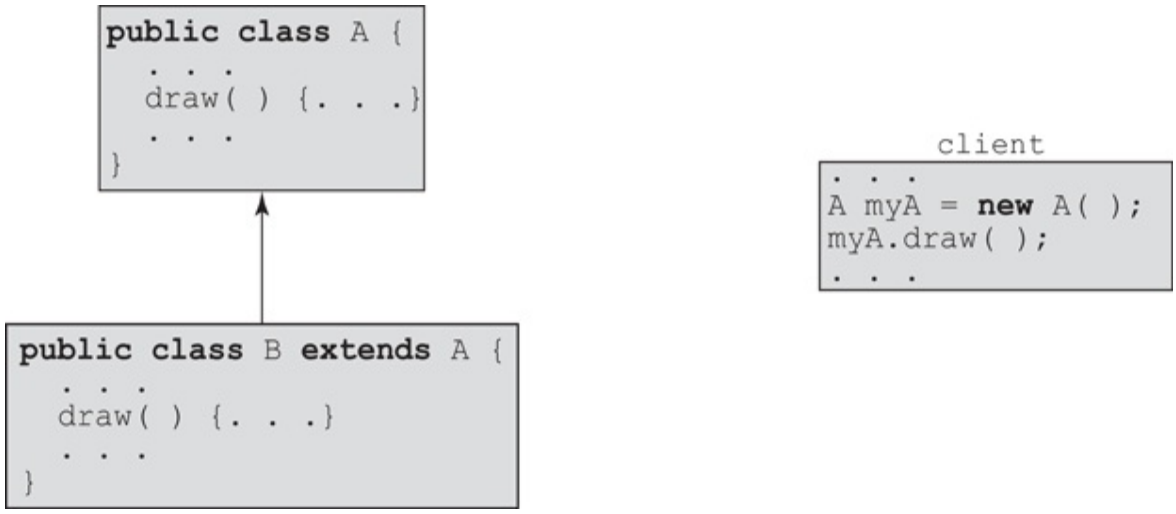


Figure 12.2 Dynamic binding

Figure 12.2 Full Alternative Text

Polymorphism is a natural part of any object-oriented language that is statically typed. In a sense, polymorphism makes a statically typed language a little bit dynamically typed, where the little bit is in some bindings of method calls to methods. The type of a polymorphic variable is indeed dynamic.

One purpose of dynamic binding is to allow software systems to be more easily extended during both development and maintenance. Suppose we have a catalog of used cars that is implemented as a `Car` class and a subclass for

each car in the catalog. The subclasses contain an image of the car and specific information about the car. Users can browse the cars with a program that displays the images and information about each car as the user browses to it. The display of each car (and its information) includes a button that the user can click if he or she is interested in that particular car. After the user gets through the catalog, the system will print the images and information about the cars of interest to the user. One way to implement this system is to place a reference to each car (subclass of `Car`) of interest in a list that can store references to the base class, `Car`. When the user is ready, information about all of the cars of interest could be printed for the user to study and compare the cars in the list. The catalog of cars will of course change frequently. This will necessitate corresponding changes in the subclasses of `Car`. However, changes to the collection of subclasses will not require any other changes to the system.

In some cases, the design of an inheritance hierarchy results in one or more classes that are so high in the hierarchy that an instantiation of them would not make sense. For example, suppose a program defined a `Building` class and a collection of subclasses for specific types of buildings, for instance, `French_Gothic_Cathedrals`. It probably would not make sense to have an implemented `draw` method in `Building`. But because all of its descendant classes should have such methods, the protocol (but not the body) of that method is included in `Building`. Such a method is often called an **abstract method** (*pure virtual method* in C++). A class that includes at least one abstract method is called an **abstract class** (*abstract base class* in C++). Such a class usually cannot be instantiated, because some of its methods are declared but are not defined (they do not have bodies). Any subclass of an abstract class that is to be instantiated must provide implementations (definitions) of all of the inherited abstract methods.

12.3 Design Issues for Object-Oriented Languages

A number of issues must be considered when designing the programming language features to support inheritance and dynamic binding. Those that we consider most important are discussed in this section.

12.3.1 The Exclusivity of Objects

A language designer who is totally committed to the object model of computation designs an object system that subsumes all other concepts of type. Everything, from a simple scalar integer to a complete software system, is an object in this mind-set. The advantage of this choice is the elegance and pure uniformity of the language and its use. The primary disadvantage is that simple operations must be done through the message-passing process, which often makes them slower than similar operations in an imperative model, where single machine instructions may implement such simple operations. In this purest model of object-oriented computation, all types are classes. There is no distinction between predefined and user-defined classes. In fact, all classes are treated the same way and all computation is accomplished through message passing.

One alternative to the exclusive use of objects that is common in imperative languages to which support for object-oriented programming has been added is the following: Retain the complete collection of types from the base imperative language and add the object typing model. This approach results in a larger language whose type structure can be confusing to new users of the language.

Another alternative to the exclusive use of objects is to have an imperative-style type structure for the primitive scalar types, but implement all structured types as objects. This choice provides the speed of operations on primitive

values that is comparable to those expected in the imperative model.

12.3.2 Are Subclasses Subtypes?

If a language allows programs in which a variable of a class can be substituted for a variable of one of its ancestor classes in any situation, without causing type errors and without changing the behavior of the program, that language supports the **principle of substitution**. In such a language, if class `B` is derived from class `A`, then `B` has everything `A` has and the behavior of an object of class `B`, when used in place of an object of class `A`, is identical to that of an object of class `A`.[7] When this is true, `B` is a **subtype** of `A`. Although a subclass that is a subtype of its parent class must expose all of the members that are exposed by its parent class, the subclass can have members that are not in the parent class and still be a subtype.

7. There is a fundamental theoretical problem with this last requirement: In general, it is not possible to determine if the behaviors of two programs are identical.

The subtypes of Ada are examples of predefined subtypes. For example,

```
subtype Small_Int is Integer range -100..100;
```

Variables of `Small_Int` type have all of the operations of `Integer` variables but can store only a subset of the values possible in `Integer`. Furthermore, every `Small_Int` variable can be used anywhere an `Integer` variable can be used. That is, every `Small_Int` variable is, in a sense, an `Integer` variable.

The definition of subtype clearly disallows having public entities in the parent class that are not public in the subclass. So, the derivation process for subtypes must require that public entities of the parent class are inherited as public entities in the subclass.

Not all subclasses are subtypes and not all subtypes are subclasses. For example, a subclass cannot be a subtype if it changes the behavior of one of its overriding methods. Also, a class that is not a subclass of another class can be a subtype of that class by defining the same members, in terms of both

types and behavior. A subtype inherits interfaces and behavior, while a subclass inherits implementation, primarily to promote code reuse.

Most static-typed languages that support object-oriented programming are designed so that subclasses are subtypes, unless the programmer specifically designs a subclass that has behavior that differs from that of its parent class.

One obvious question is: Is the issue of whether subclasses are subtypes a theoretical or practical one? It is probably unusual to define a subclass whose overriding methods preserve the type protocols of their corresponding overridden methods but not their effects. So it is not a frequent practical issue. However, requiring all subclasses to be subtypes, if there were a reasonably simple way to enforce that, would place inheritance on a sounder theoretical base.

12.3.3 Single and Multiple Inheritance

Another simple design issue for object-oriented languages is: Does the language allow multiple inheritance (in addition to single inheritance)? Maybe it’s not so simple. The purpose of multiple inheritance is to allow a new class to inherit from two or more classes.

Because multiple inheritance is sometimes highly useful, why would a language designer not include it? The reasons lie in two categories: complexity and efficiency. The additional complexity is illustrated by several problems. First, note that if a class has two unrelated parent classes and neither defines a name that is defined in the other, there is no problem. However, suppose a subclass named `C` inherits from both class `A` and class `B` and both `A` and `B` define an inheritable method named `display`. If `C` needs to reference both versions of `display`, how can that be done? This ambiguity problem is further complicated when the two parent classes both define identically named methods and one or both of them must be overridden in the subclass.

Another issue arises if both `A` and `B` are derived from a common parent, `Z`, and `C` has both `A` and `B` as parent classes. This situation is called **diamond** or **shared** inheritance. In this case, both `A` and `B` should include `Z`'s inheritable variables. Suppose `Z` includes an inheritable variable named `sum`. The question is whether `C` should inherit both versions of `sum` or just one, and if just one, which one? There may be programming situations in which just one of the two should be inherited, and others in which both should be inherited. A similar problem occurs when both `A` and `B` inherit a method from `Z` and both override that method. If a client of `C`, which inherits both overriding methods, calls the method, which method is called, or are both supposed to be called. Diamond inheritance is shown in Figure 12.3.

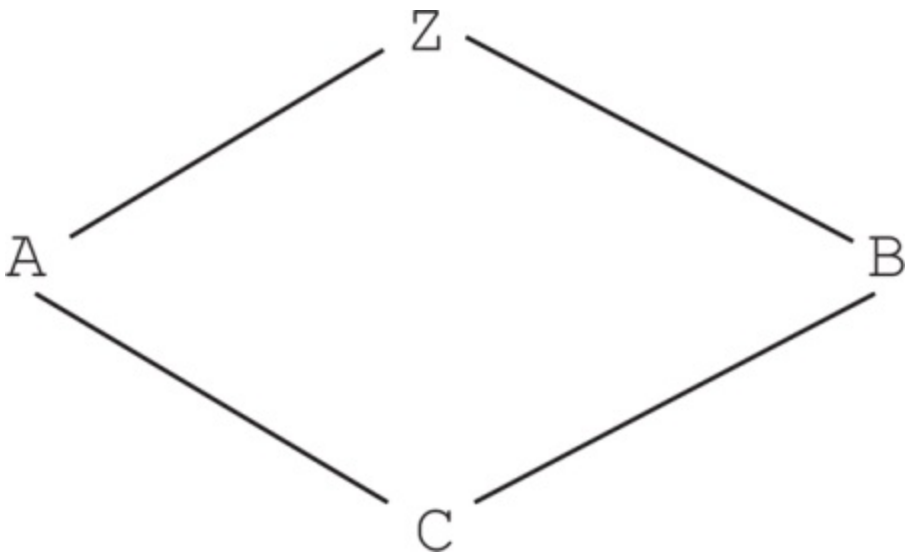


Figure 12.3 An example of diamond inheritance

The question of efficiency may be more perceived than real. In C++, for example, supporting multiple inheritance requires just one additional array access and one extra addition operation for each dynamically bound method call, at least with some machine architectures (Stroustrup, 1994, p. 270). Although this operation is required even if the program does not use multiple inheritance, it is a small additional cost.

The use of multiple inheritance can easily lead to complex program organizations. Many who have attempted to use multiple inheritance have found that designing the classes to be used as multiple parents is difficult. And the difficulties are not restricted to those created by the initial developer. A class might be used by another developer at some later date as one of the parents of a new class. Maintenance of systems that use multiple inheritance can be a more serious problem, for multiple inheritance leads to more complex dependencies among classes. It is not clear to some that the benefits of multiple inheritance are worth the added effort to design and maintain a system that uses it.

An interface is somewhat similar to an abstract class; its methods are declared but not defined. Interfaces cannot be instantiated. They are used as an alternative to multiple inheritance.8 Interfaces provide some of the benefits of multiple inheritance but have fewer disadvantages. For example, the problems of diamond inheritance are avoided when interfaces, rather than multiple inheritance, are used.

8. Interfaces initially appeared in Java, whose designers recognized the difficulties that are raised by the use of multiple inheritance.

12.3.4 Allocation and Deallocation of Objects

There are two design questions concerning the allocation and deallocation of objects. The first of these is the place from which objects are allocated. If they behave like the abstract data types, then they can be allocated from anywhere. This means they could be allocated from the run-time stack or explicitly created on the heap with an operator or function, such as **new**. If they are all heap dynamic, there is the advantage of having a uniform method of creation and access through pointer or reference variables. This design simplifies the assignment operation for objects, making it in all cases only a pointer or reference value change. It also allows references to objects to be implicitly dereferenced, simplifying the access syntax.

If objects are stack dynamic, there is a potential problem with regard to subtypes. If class `B` is a child of class `A` and `B` is a subtype of `A`, then an object of `B` type can be assigned to a variable of `A` type. For example, if `b1` is a variable of `B` type and `a1` is a variable of `A` type, then

```
a1 = b1;
```

is a legal statement. If `a1` and `b1` are references to heap-dynamic objects, there is no problem—the assignment is a simple pointer assignment. However, if `a1` and `b1` are stack dynamic, then they are value variables and, if assigned the value of the object, must be copied to the space of the target object. If `B` adds a data field to what it inherited from `A`, then `a1` will not have sufficient space on the stack for all of `b1`. The excess will simply be truncated, which could be confusing to programmers who write or use the code. This truncation is called **object slicing**. The following example and Figure 12.4 illustrate the problem.

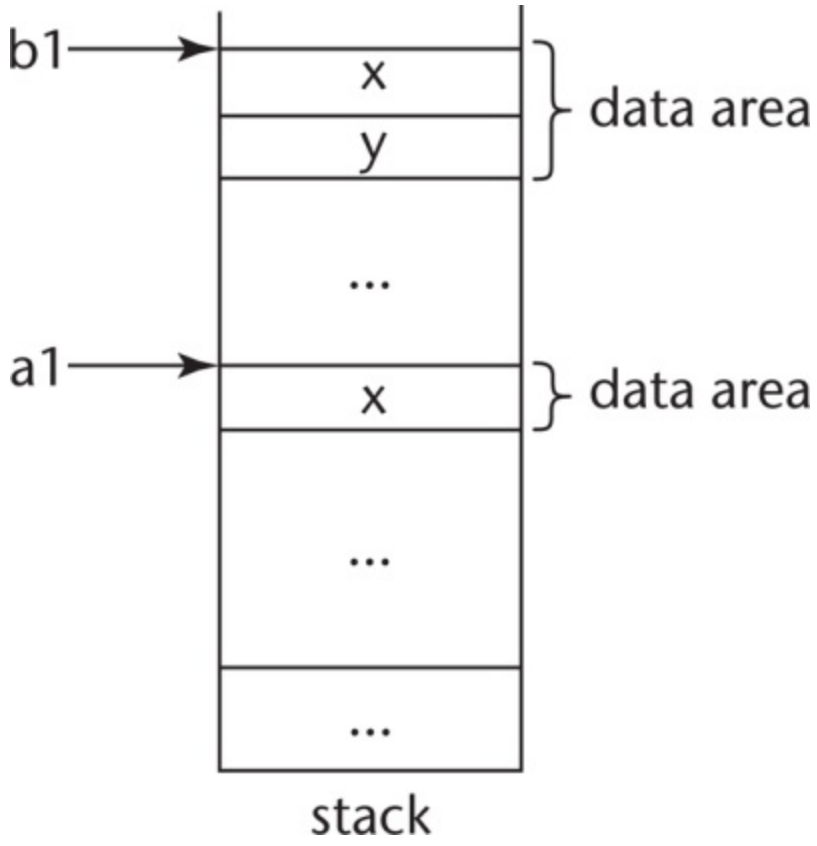


Figure 12.4 An example of object slicing

Figure 12.4 Full Alternative Text

```
class A {
  int x;
  . . .
};
class B : A {
  int y;
  . . .
}
```

The second question here is concerned with those cases where objects are allocated from the heap. The question is whether deallocation is implicit, explicit, or both. If deallocation is implicit, some implicit method of storage reclamation is required. If deallocation can be explicit, that raises the issue of whether dangling pointers or references can be created.

12.3.5 Dynamic and Static Binding

As discussed in Section 12.2.3, dynamic binding of messages to methods is an essential part of object-oriented programming. The question here is whether all bindings of messages to methods are dynamic. The alternative is to allow the user to specify whether a specific binding is to be dynamic or static. The advantage of this is that static bindings are faster. So, if a binding need not be dynamic, why pay the price?

12.3.6 Nested Classes

One of the primary motivations for nesting class definitions is information hiding. If a new class is needed by only one class, there is no reason to define it so it can be seen by other classes. In this situation, the new class can be

nested inside the class that uses it. In some cases, the new class is nested inside a subprogram, rather than directly in another class.

The class in which the new class is nested is called the **nesting class**. The most obvious design issues associated with class nesting are related to visibility. Specifically, one issue is: Which of the members of the nesting class are visible in the nested class? The other important issue is the opposite: Which of the members of the nested class are visible in the nesting class?

12.3.7 Initialization of Objects

The initialization issue is whether and how objects are initialized to values when they are created. This is more complicated than may be first thought. One question is whether objects must be initialized manually or through some implicit mechanism. When an object of a subclass is created, is the associated initialization of the inherited parent class member implicit or must the programmer explicitly deal with it?

12.4 Support for Object-Oriented Programming in Specific Languages

12.4.1 Smalltalk

Many think of Smalltalk as the definitive object-oriented programming language. It was the first language to include complete support for that paradigm. Therefore, it is natural to begin a survey of language support for object-oriented programming with Smalltalk.

12.4.1.1 General Characteristics

In Smalltalk, the concept of an object is truly universal. Virtually everything, from things as simple as the integer constant `2` to a complex file-handling system, is an object. As objects, they are treated uniformly. They all have local memory, inherent processing ability, the capability to communicate with other objects, and the possibility of inheriting methods and instance variables from ancestors. Classes cannot be nested in Smalltalk.

All computation is through messages, even a simple arithmetic operation. For example, the expression `x + 7` is implemented as sending the `+` message to `x` (to enact the `+` method), sending `7` as the parameter. This operation returns a new numeric object with the result of the addition.

Replies to messages have the form of objects and are used to return requested or computed information or only to confirm that the requested service has been completed.

All Smalltalk objects are allocated from the heap and are referenced through reference variables, which are implicitly dereferenced. There is no explicit deallocation statement or operation. So, all deallocation is implicit, using a

garbage collection process for storage reclamation.

In Smalltalk, constructors must be explicitly called when an object is created. A class can have multiple constructors, but each must have a unique name.

Smalltalk classes cannot be nested in other classes.

Unlike a hybrid language such as C++, Smalltalk was designed for just one software development paradigm—object oriented. Furthermore, it adopts none of the appearance of the imperative languages. Its purity of purpose is reflected in its simple elegance and uniformity of design.

There is an example Smalltalk program in Chapter 2.

12.4.1.2 Inheritance

A Smalltalk subclass inherits all of the members of its superclass. The subclass can also have its own instance variables, which must have names that are distinct from the variable names in its ancestor classes. Finally, the subclass can define new methods and redefine methods that already exist in an ancestor class. When a subclass has a method whose name and protocol are the same as an ancestor class, the subclass method hides that of the ancestor class. Access to such a hidden method is provided by prefixing the message with the pseudovariable **`super`**. The prefix causes the method search to begin in the superclass rather than locally.

Because members in a parent class cannot be hidden from subclasses, subclasses can be and usually are subtypes.

Smalltalk does not support multiple inheritance.

12.4.1.3 Dynamic Binding

The dynamic binding of messages to methods in Smalltalk operates as follows: A message to an object causes a search of the class to which the

object belongs for a corresponding method. If the search fails, it is continued in the superclass of that class, and so forth, up to the system class, `Object`, which has no superclass. `Object` is the root of the class derivation tree on which every class is a node. If no method is found anywhere in that chain, an error occurs. It is important to remember that this method search is dynamic —it takes place when the message is sent. Smalltalk does not, under any circumstances, bind messages to methods statically.

The only type checking in Smalltalk is dynamic, and the only type error occurs when a message is sent to an object that has no matching method, either locally or through inheritance. This is a different concept of type checking than that of most other languages. Smalltalk type checking has the simple goal of ensuring that a message matches some method.

Smalltalk variables are not typed; any name can be bound to any object. As a direct result, Smalltalk supports dynamic polymorphism. All Smalltalk code is generic in the sense that the types of the variables are irrelevant, as long as they are consistent. The meaning of an operation (method or operator) on a variable is determined by the class of the object to which the variable is currently bound.

The point of this discussion is that as long as the objects referenced in an expression have methods for the messages of the expression, the types of the objects are irrelevant. This means that no code is tied to a particular type.

12.4.1.4 Evaluation of Smalltalk

Smalltalk is a small language, although the Smalltalk system is large. The syntax of the language is simple and highly regular. It is a good example of the power that can be provided by a small language if that language is built around a simple but powerful concept. In the case of Smalltalk, that concept is that all programming can be done employing only a class hierarchy built using inheritance, objects, and message passing.

In comparison with conventional compiled imperative-language programs, equivalent Smalltalk programs are significantly slower. Although it is

theoretically interesting that array indexing and loops can be provided within the message-passing model, efficiency is an important factor in the evaluation of programming languages. Therefore, efficiency will clearly be an issue in most discussions of the practical applicability of Smalltalk.

Smalltalk's dynamic binding allows type errors to go undetected until run time. A program can be written that includes messages to nonexistent methods and it will not be detected until the messages are sent, which causes a great deal more error repair later in the development than would occur in a static-typed language. However, in practice type errors are not a serious problem with Smalltalk programs.

Overall, the design of Smalltalk consistently came down on the side of language elegance and strict adherence to the principles of object-oriented programming support, often without regard for practical matters, in particular execution efficiency. This is most obvious in the exclusive use of objects and the typeless variables.

The Smalltalk user interface has had an important impact on computing: The integrated use of windows, mouse-pointing devices, and pop-up and pull-down menus, all of which first appeared in Smalltalk, dominate contemporary software systems.

Perhaps the greatest impact of Smalltalk is the advancement of object--oriented programming, now the most widely used design and coding methodology.

12.4.2 C++

Chapter 2 describes how C++ evolved from C and SIMULA 67, with the design goal of support for object-oriented programming while retaining nearly complete backward compatibility with C. C++ classes, as they are used to support abstract data types, are discussed in Chapter 11. C++ support for the other essentials of object-oriented programming is explored in this section. The whole collection of details of C++ classes, inheritance, and dynamic binding is large and complex. This section discusses only the most

important among these topics, specifically, those directly related to the design issues described in Section 12.3.

C++ was the first widely used object-oriented programming language and is still among the most popular. So, naturally, it is the one with which other languages are often compared. For both of these reasons, our coverage of C++ here is more detailed than that of the other example languages discussed in this chapter.

12.4.2.1 General Characteristics

To maintain backward compatibility with C, C++ retains the type system of C and adds classes to it. Therefore, C++ has both traditional imperative--language types and the class structure of an object-oriented language. It supports methods, as well as functions that are not related to specific classes. This makes it a hybrid language, supporting both procedural programming and object-oriented programming.

The objects of C++ can be static, stack dynamic, or heap dynamic. Explicit deallocation using the **`delete`** operator is required for heap-dynamic objects, because C++ does not include implicit storage reclamation.

Many class definitions include a destructor method, which is implicitly called when an object of the class ceases to exist. The destructor is used to deallocate heap-allocated memory that is referenced by data members. It may also be used to record part or all of the state of the object just before it dies, usually for debugging purposes.

12.4.2.2 Inheritance

A C++ class can be derived from an existing class, which is then its parent, or base, class. Unlike Smalltalk and most other languages that support object--oriented programming, a C++ class can also be stand-alone, without a superclass. In the definition of a derived class, the name of the derived class has the name of the base class attached with a colon (:), as in the following

syntactic form:

- **`class`** derived_class_name `:` base_class_name `{ ... }`

The data defined in a class definition are called *data members* of that class, and the functions defined in a class definition are called *member functions* of that class (member functions in other languages are usually called methods). Some or all of the members of the base class may be inherited by the derived class, which can also add new members and modify inherited member functions.

All C++ objects must be initialized before they are used. Therefore, all C++ classes include at least one constructor method that initializes the data members of the new object. Constructor methods are implicitly called when an object is created. If any of the data members are pointers to heap-allocated data, the constructor allocates that storage.

If a class is derived from another class, the inherited data members must be initialized when the derived class object is created. To do this, the base class constructor is implicitly called. When initialization data must be furnished to the base class constructor, it is given in the call to the derived object - constructor. In general, this is done with the following construct:

- subclass`(`subclass parameters`):` base_class`(`superclass parameters`) {`
- `. . .`
- `}`

interview

On Paradigms and Better Programming

BJARNE STROUSTRUP

Bjarne Stroustrup is the designer and original implementer of C++ and the author of A Tour of C++, Programming: Principles and Practice Using C++, The C++ Programming Language, The Design and Evolution of C++, and many other publications. His research interests include distributed systems, design, programming techniques, software development tools, and programming languages. He is actively involved in the ANSI/ISO standardization of C++. Dr. Stroustrup is a managing director in the technology division of Morgan Stanley in New York City, a Visiting Professor in Computer Science at Columbia University, and a Distinguished Research Professor in Computer Science at Texas A&M University. He is a member of the National Academy of Engineering, an ACM Fellow, and an IEEE Fellow. In 1993, Stroustrup received the ACM Grace Murray Hopper Award “for his early work laying the foundations for the C++ programming language. Based on the foundations and Dr. Stroustrup’s continuing efforts, C++ has become one of the most influential programming languages in the history of computing.”

(year of interview: 2002)

PROGRAMMING PARADIGMS

Your thoughts on the object-oriented paradigm: Its pluses and minuses. Let me first say what I mean by OOP—too many people think that "object-oriented" is simply a synonym for "good." If so, there would be no need for other paradigms. The key to OO is the use of class hierarchies providing polymorphic behavior through some rough equivalent of virtual functions. For proper OO, it is important to avoid directly accessing the data in such a hierarchy and to use only a well-designed functional interface.

In addition to its well-documented strengths, object-oriented programming also has obvious weaknesses. In particular, not every concept naturally fits into a class hierarchy, and the mechanisms supporting object-oriented programming can impose significant overheads compared to alternatives. For many simple abstractions, classes that do not rely on hierarchies and run-time binding provide a simpler and more efficient alternative. Furthermore, where no run-time resolution is needed, generic programming relying on (compile-time) parametric polymorphism is a better behaved and more efficient approach.

So, C++: Is it OO or other? C++ supports several paradigms—including OOP, generic programming, and procedural programming—and combinations of these paradigms define multiparadigm programming as supporting more than one programming style ("paradigm") and combinations of those styles.

Do you have a mini-example of multiparadigm programming? Consider this variant of the classic "collection of shapes" examples (originating from the early days of the first language to support object-oriented programming: Simula 67):

```
void draw_all(const vector<Shape*>& vs)
{
   for (int i = 0; i<vs.size(); ++i)
       vs[i]->draw();
}
```

Here, I use the generic container `vector` together with the polymorphic type

`Shape`. The `vector` provides static type safety and optimal run-time performance. The `Shape` provides the ability to handle a `Shape` (i.e., any object of a class derived from `Shape`) without recompilation.

We can easily generalize this to any container that meets the C++ standard library requirements:

```
template<class C>
       void draw_all(const C& c)
{
    typedef typename C::
       const_iterator CI;
    for (CI p = c.begin();
       p!=c.end(); ++p)
       (*p)->draw();
}
```

Using iterators allows us to apply this `draw_all ()` to containers that do not support subscripts, such as a standard library list:

```
vector<Shape*> vs;
list<Shape*> ls;
// . . .
draw_all(vs);
draw_all(ls);
```

We can even generalize this further to handle any sequence of elements defined by a pair of iterators:

```
template<class Iterator> void draw_all(Iterator b, Iterator e)
{
    for_each(b,e,mem_fun(&Shape::draw));
}
```

To simplify the implementation, I used the standard library algorithm `for_each`.

We might call this last version of `draw_all()` for a standard library list and an array:

```
list<Shape*> ls;
Shape* as[100];
// . . .
```

```
draw_all(ls.begin(),ls.end());
draw_all(as,as+100);
```

SELECTING THE "RIGHT" LANGUAGE FOR THE JOB

How useful is it to have this background in numerous paradigms? Or would it be better to invest time in becoming even more familiar with OO languages rather than learning these other paradigms? It is essential for anyone who wants to be considered a professional in the areas of software to know several languages and several programming paradigms. Currently, C++ is the best language for multiparadigm programming and a good language for learning various forms of programming. However, it's not a good idea to know just C++, let alone to know just a single-paradigm language. That would be a bit like being colorblind or monoglot: You would hardly know what you were missing. Much of the inspiration to good programming comes from having learned and appreciated several programming styles and seen how they can be used in different languages.

Furthermore, I consider programming of any nontrivial program a job for professionals with a solid and broad education, rather than for people with a hurried and narrow "training."

If no constructor is included by the developer in a class definition, the compiler includes a trivial constructor. This default constructor calls the constructor of the base class, if there is a base class.

Class members can be private, protected, or public. Private members are accessible only by member functions and friends of the class. Standalone functions, member functions, and classes can be declared to be friends of a class and thereby be given access to its private members. Public members are visible everywhere. Protected members are like private members, except in derived classes, whose access is described next. Derived classes can modify accessibility for their inherited members. The complete syntactic form of a derived class is as follows:

- **`class`** derived_class_name : derivation_mode base_class_name

  {data member and member function declarations};

The derivation_mode can be either **`public`** or **`private`**.9 (Do not confuse public and private derivation with public and private members.) The public and protected members of a base class are also public and protected, respectively, in a public-derived class. In a private-derived class, both the public and protected members of the base class are private. So, in a class hierarchy, a private-derived class cuts off access to all members of all ancestor classes to all successor classes. Private members of a base class are inherited by a derived class, but they are not visible to the members of that derived class and are therefore of no use there. Private derivations provide the possibility that a subclass can have members with different access than the same members in the parent class. Consider the following example:

9. It can also be **`protected`**, but that option is not discussed here.

```
class base_class {
  private:
    int a;
    float  x;
  protected:
    int b;
    float y;
  public:
    int c;
    float z;
};
 class subclass_1 : public base_class {. . .};
 class subclass_2 : private base_class {. . .};
```

In `subclass_1`, `b` and `y` are protected, and `c` and `z` are public. In `subclass_2`, `b`, `y`, `c`, and `z` are private. No derived class of `subclass_2` can have members with access to any member of `base_class`. The data members `a` and `x` in `base_class` are not accessible in either `subclass_1` or `subclass_2`.

Note that private-derived subclasses cannot be subtypes. For example, if the base class has a public data member, under private derivation that data member would be private in the subclass. Therefore, if an object of the subclass were substituted for an object of the base class, accesses to that data

member would be illegal on the subclass object. However, public-derived subclasses can be and usually are subtypes.

Under private class derivation, no member of the parent class is implicitly visible to the instances of the derived class. Any member that must be made visible must be reexported in the derived class. This reexportation in effect exempts a member from being hidden even though the derivation was private. For example, consider the following class definition:

```
class subclass_3 : private base_class {
  base_class :: c;
  . . .
}
```

Now, instances of `subclass_3` can access `c`. As far as `c` is concerned, it is as if the derivation had been public. The double colon (`::`) in this class definition is a scope resolution operator. It specifies the class where its following entity is defined.

The example in the following paragraphs illustrates the purpose and use of private derivation.

Consider the following example of C++ inheritance, in which a general linked-list class is defined and then used to define two useful subclasses:

```
class single_linked_list {
  private:
    class node {
      public:
        node *link;
        int contents;
    };
    node *head;
  public:
    single_linked_list() {head = 0};
    void  insert_at_head(int);
    void  insert_at_tail(int);
    int  remove_at_head();
    int  empty();
};
```

The nested class, `node`, defines a cell of the linked list to consist of an integer

variable and a pointer to a `node` object. The `node` class is in the private clause, which hides it from all other classes. Its members are public, however, so they are visible to the nesting class, `single_linked_list`. If they were private, `node` would need to declare the nesting class to be a friend to make them visible in the nesting class. Note that nested classes have no special access to members of the nesting class. Only static data members of the nesting class are visible to methods of the nested class.10

10. A class can also be defined in a method of a nesting class. The scope rules of such classes are the same as those for classes nested directly in other classes, even for the local variables declared in the method in which they are defined.

The enclosing class, `single_linked_list`, has just a single data member, a pointer to act as the list's header. It contains a constructor function, which sets `head` to the null pointer value. The four member functions allow nodes to be inserted at either end of a list object, nodes to be removed from one end of a list, and lists to be tested for empty.

The following definitions provide stack and queue classes, both based on the `single_linked_list` class:

```
class stack : public single_linked_list {
  public:
    stack() {}
    void push(int value) {
      insert_at_head(value);
    }
    int pop() {
      return remove_at_head();
    }
};
class queue : public single_linked_list {
  public:
    queue() {}
    void enqueue(int value) {
      insert_at_tail(value);
    }
    int dequeue() {
      remove_at_head();
    }
};
```

Note that objects of both the `stack` and `queue` subclasses can access the `empty` function defined in the base class, `single_linked_list` (because it is a public derivation). Both subclasses define constructor functions that do nothing. When an object of a subclass is created, the proper constructor in the subclass is implicitly called. Then, any applicable constructor in the base class is called. So, in our example, when an object of type `stack` is created, the constructor in `single_linked_list` is called, which does the necessary initialization. Then the `stack` constructor is called, which does nothing.

The classes `stack` and `queue` both suffer from the same serious problem: Clients of both can access all of the public members of the parent class, `single_linked_list`. A client of a `stack` object could call `insert_at_tail`, thereby destroying the integrity of its stack. Likewise, a client of a `queue` object could call `insert_at_head`. These unwanted accesses are allowed because both `stack` and `queue` are subtypes of `single_linked_list`. Public derivation is used when one wants the subclass to inherit the entire interface of the base class. The alternative is to use a derivation in which the subclass inherits only the implementation of the base class. Our two example derived classes can be written to make them not subtypes of their parent class by using **`private`**, rather than **`public`**, derivation.11 Then, both will also need to reexport `empty`, because it will become hidden to their instances. This situation illustrates the motivation for the private-derivation option. The new definitions of the stack and queue types, named `stack_2` and `queue_2`, are shown in the following:

11. They would not be subtypes because the public members of the parent class can be seen in a client, but not in a client of the subclass, where those members are private.

```
class stack_2 : private single_linked_list {
  public:
    stack_2() {}
    void push(int value) {
      single_linked_list :: insert_at_head(value);
    }
    int pop() {
      return single_linked_list :: remove_at_head();
    }
    single_linked_list:: empty();
};
```

```
class queue_2 : private single_linked_list {
  public:
    queue_2() {}
    void enqueue(int value) {
      single_linked_list :: insert_at_tail(value);
    }
    int dequeue() {
      single_linked_list :: remove_at_head();
    }
    single_linked_list:: empty();
};
```

These two classes use reexportation to allow access to base class methods for clients. This was not necessary when public derivation was used.

The two versions of stack and queue illustrate the difference between subtypes and derived types that are not subtypes. The linked list is a generalization of both stacks and queues, because both can be implemented as linked lists. So, it is natural to inherit from a linked-list class to define stack and queue classes. However, neither is a subtype of the linked-list class, because both make the public members of the parent class private, which makes them inaccessible to clients.

One of the reasons friends are necessary is that sometimes a subprogram must be written that can access the members of two different classes. For example, suppose a program uses a class for vectors and one for matrices, and a subprogram is needed to multiply a matrix object by a vector object. In C++, the multiply function can be made a friend of both classes.

As previously stated, C++ provides multiple inheritance. As an example, suppose we wanted a class for drawing that needed the behavior of a class written for drawing figures and the methods of the new class needed to run in a separate thread. We might define the following:

```
class Thread { . . . };
class Drawing { . . . };
class DrawThread : public Thread, public Drawing { . . . };
```

Class `DrawThread` inherits all of the members of both `Thread` and `Drawing`. If both `Thread` and `Drawing` happen to include members with the same name, they can be unambiguously referenced in objects of class `DrawThread` by

using the scope resolution operator (::). This example of multiple inheritance is shown in Figure 12.5.

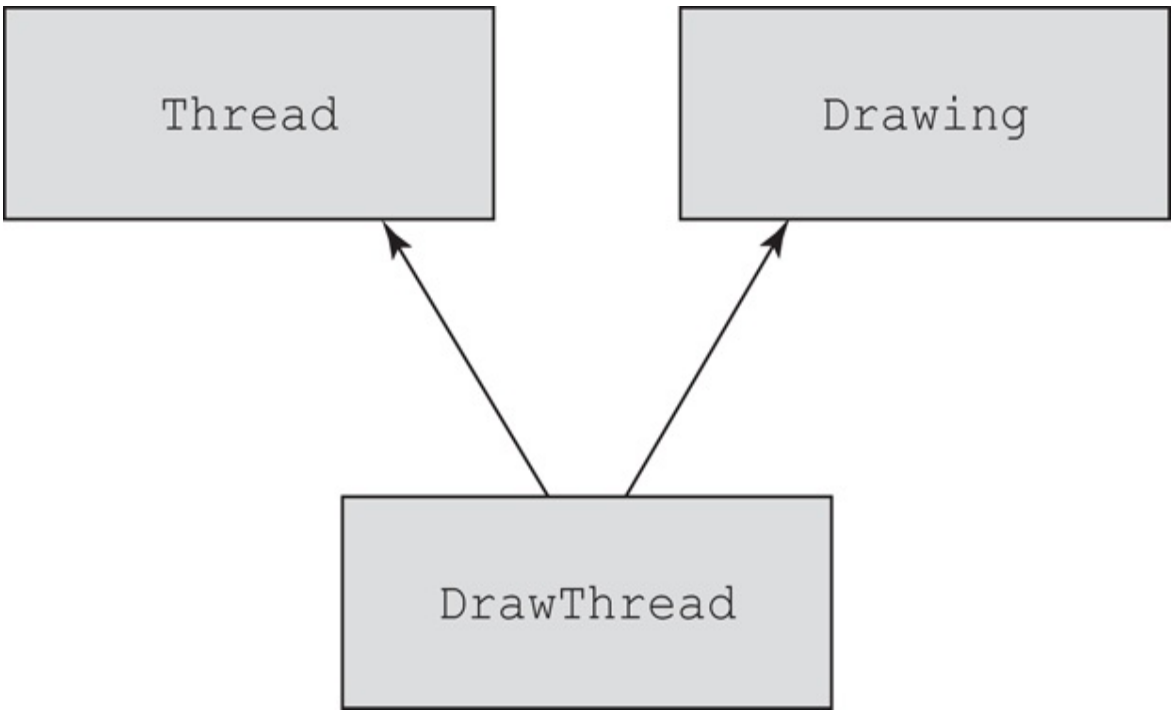


Figure 12.5 Multiple inheritance

Some issues with the C++ implementation of multiple inheritance are discussed in Section 12.5.

Overriding methods12 in C++ must have exactly the same parameter profile as the overridden method. If there is any difference in the parameter profiles, the method in the subclass is considered a new method that is unrelated to the method with the same name in the ancestor class. The return type of the overriding method either must be the same as that of the overridden method or must be a publicly derived type of the return type of the overridden method.

12. Recall that an overriding method is one that is defined in the derived class to replace a virtual method inherited from an ancestor class. Calls to an

overriding method must be dynamically bound.

12.4.2.3 Dynamic Binding

All of the member functions we have defined thus far are statically bound; that is, a call to one of them is statically bound to a function definition. A C++ object could be manipulated through a value variable, rather than a pointer or a reference. (Such an object would be static or stack dynamic.) However, in that case, the object's type is known and static, so dynamic binding is not needed. On the other hand, a pointer variable that has the type of a base class can be used to point to any heap-dynamic object of any class publicly derived from that base class, making it a polymorphic variable. Publicly derived subclasses are subtypes if none of the members of the base class are private. Privately derived subclasses are never subtypes. A pointer to a base class cannot be used to reference a method in a subclass that is not a subtype.

C++ does not allow value variables (as opposed to pointers or references) to be polymorphic. When a polymorphic variable is used to call a member function overridden in one of the derived classes, the call must be dynamically bound to the correct member function definition.

Consider the situation of having a base class named `Shape`, along with a collection of derived classes for different kinds of shapes, such as circles, rectangles, and so forth. If these shapes need to be displayed, then the displaying member function, `draw`, must be unique for each descendant, or kind of shape. These versions of `draw` must be defined to be virtual. When a call to `draw` is made with a pointer to the base class of the derived classes, that call must be dynamically bound to the member function of the correct derived class. The following example has the skeletal definitions for the example situation just described:

```
class Shape {
  public:
    virtual void  draw() = 0;
  . . .
};
```

```
class Circle : public Shape {
  public:
    void draw() { . . . }
  . . .
};
class Rectangle : public Shape {
  public:
    void draw() { . . . }
  . . .
};
```

Given these definitions, the following code has examples of both statically and dynamically bound calls:

```
Circle* circ = new Circle;
Rectangle* rect = new Rectangle;
Shape* ptr_shape;
ptr_shape = circ;          // Now ptr_shape points to a
                           //  Circle object
ptr_shape->draw();      // Dynamically bound to the draw
                           //  in the Circle class
rect->draw();              // Statically bound to the draw
                           //  in the Rectangle class
```

This situation is shown in Figure 12.6.

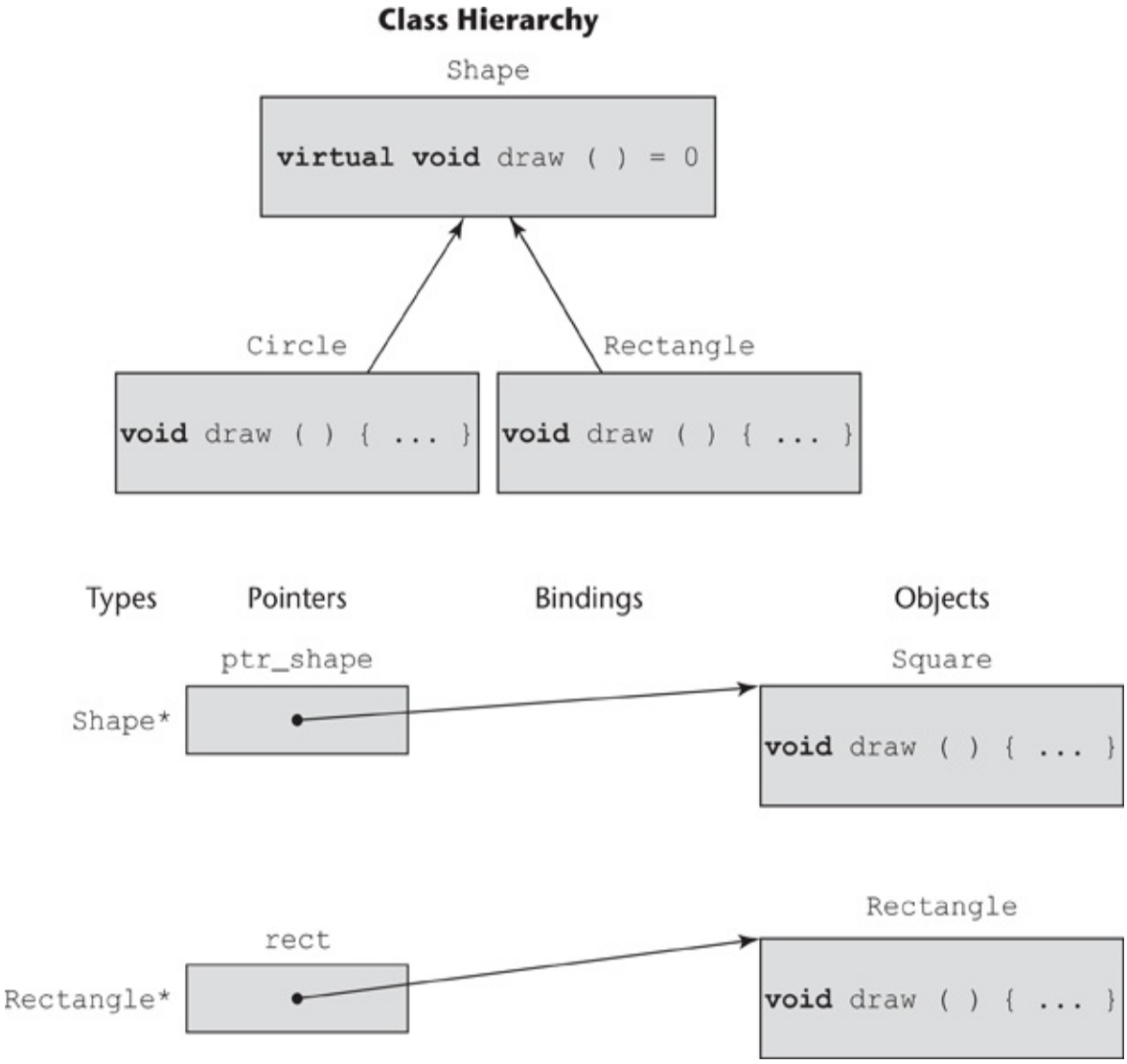


Figure 12.6 Dynamic binding

Figure 12.6 Full Alternative Text

Notice that the `draw` function in the definition of the base class `shape` is set to 0. This peculiar syntax is used to indicate that this member function is a **pure virtual function**, meaning that it has no body and it cannot be called. It must be redefined in derived classes if they call the function. The purpose of a pure virtual function is to provide the interface of a function without giving any of its implementation. Pure virtual functions are usually defined when an actual member function in the base class would not be useful. Recall that in

Section 12.2.3, a base class `Building` was discussed, and each subclass described some particular kind of building. Each subclass had a `draw` method but none of these would be useful in the base class. So, `draw` would be a pure virtual function in the `Building` class.

Any class that includes a pure virtual function is an **abstract class**. In C++, an abstract class is not marked with a reserved word. An abstract class can include completely defined methods. Because of the presence of one or more virtual functions, it is illegal to instantiate an abstract class. In a strict sense, an abstract class is one that is used only to represent the characteristics of a type. C++ provides abstract classes to model these truly abstract classes. If a subclass of an abstract class does not redefine a pure virtual function of its parent class, that function remains as a pure virtual function in the subclass and the subclass is also an abstract class.

Abstract classes and inheritance together support a powerful technique for software development. They allow types to be hierarchically defined so that related types can be subclasses of truly abstract types that define their common abstract characteristics.

Dynamic binding allows the code that uses members like `draw` to be written before all or even any of the versions of `draw` are written. New derived classes could be added years later, without requiring any change to the code that uses such dynamically bound members. This is a highly useful feature of object-oriented languages.

Reference assignments for stack-dynamic objects are different from pointer assignments for heap-dynamic objects. For example, consider the following code, which uses the same class hierarchy as the last example:

```
Circle circ;     // Allocate a Circle object on the stack
Rectangle rect;  // Allocate a Rectangle object on
                 //  the stack
rect = circ;     // Copies the data member values from
                 //  the Circle object
rect.draw();     // Calls the draw from the Rectangle
                 //  object
```

In the assignment `rect = circ`, the member data from the object referenced

by `circ` would be assigned to the data members of the object referenced by `rect`, but `rect` would still reference the `Rectangle` object. Therefore, the call to `draw` through the object referenced by `rect` would be that of the `Rectangle` class. If `rect` and `circ` were pointers to heap-dynamic objects, the same assignment would be a pointer assignment, which would make `rect` point to the `Circle` object, and a call to `draw` through `rect` would be bound dynamically to the `draw` in the `Circle` object.

12.4.2.4 Evaluation

It is natural to compare the object-oriented features of C++ with those of Smalltalk. The inheritance of C++ is more intricate than that of Smalltalk in terms of access control. By using both the access controls within the class definition and the derivation access controls, and also the possibility of friend functions and friend classes, the C++ programmer has highly detailed control over access to class members. Although C++ provides multiple inheritance and Smalltalk does not, there are many who feel that is not an advantage for C++. The downsides of multiple inheritance weigh heavily against its value. In fact, C++ is the only language discussed in this chapter that supports multiple inheritance. On the other hand, languages that provide alternatives to multiple inheritance, such as Java and C#, clearly have an advantage over Smalltalk in that area.

In C++, the programmer can specify whether static binding or dynamic binding is to be used. Because static binding is faster, this is an advantage for those situations where dynamic binding is not necessary. Furthermore, even the dynamic binding in C++ is fast when compared with that of Smalltalk. Binding a virtual member function call in C++ to a function definition has a fixed cost, regardless of how distant in the inheritance hierarchy the definition appears. Calls to virtual functions require only five more memory references than statically bound calls (Stroustrup, 1988). In Smalltalk, however, messages are always dynamically bound to methods, and the farther away in the inheritance hierarchy the correct method is, the longer it takes. The disadvantage of allowing the user to decide which bindings are static and which are dynamic is that the original design must include these decisions, which may have to be changed later.

The static type checking of C++ is an advantage over Smalltalk, in which all type checking is dynamic. A Smalltalk program can be written with messages to nonexistent methods, which are not discovered until the program is executed. A C++ compiler finds such errors. Compiler-detected errors are less expensive to repair than those found in testing.

Smalltalk is essentially typeless, meaning that all code is effectively generic. This provides a great deal of flexibility, but static type checking is sacrificed. C++ provides generic classes through its template facility (as described in Chapter 11), which retains the benefits of static type checking.

The primary advantage of Smalltalk lies in the elegance and simplicity of the language, which results from the single philosophy of its design. It is purely and completely devoted to the object-oriented paradigm, devoid of compromises necessitated by the whims of an entrenched user base. C++, on the other hand, is a large and complex language with no single philosophy as its foundation, except to support object-oriented programming and include the C user base. One of its most significant goals was to preserve the efficiency and flavor of C while providing the advantages of object-oriented programming. Some people feel that the features of this language do not always fit well together and that at least some of the complexity is unnecessary.

According to Chambers and Ungar (1991), Smalltalk ran a particular set of small C-style benchmarks at only 10 percent of the speed of optimized C. C++ programs require only slightly more time than equivalent C programs (Stroustrup, 1988). Given the great efficiency gap between Smalltalk and C++, it is little wonder that the commercial use of C++ is far more widespread than that of Smalltalk. There are other factors in this difference, but efficiency is clearly a strong argument in favor of C++. Of course, all of the compiled languages that support object-oriented programming run approximately 10 times faster than Smalltalk.

12.4.3 Java

Because Java’s design of classes, inheritance, and methods is similar to that

of C++, in this section we focus only on those areas in which Java differs from C++.

12.4.3.1 General Characteristics

As with C++, Java supports both objects and nonobject data. However, in Java, only values of the primitive scalar types (Boolean, character, and the numeric types) are not objects. Java's enumerations and arrays are objects. The reason Java has nonobjects is efficiency.

In Java 5.0+, primitive values are implicitly coerced when they are put in object context. This coercion converts the primitive value to an object of the wrapper class of the primitive value's type. For example, putting an **`int`** value or variable into object context causes the creation of an `Integer` object with the value of the **`int`** primitive. This coercion is called **boxing**.

Whereas C++ classes can be defined to have no parent, that is not possible in Java. All Java classes must be subclasses of the root class, `Object`, or some class that is a descendant of `Object`. One advantage of this is that some commonly needed methods, such as `toString` and `equals`, can be defined in `Object` and inherited and used by all other classes.

All Java objects are explicit heap dynamic. Most are allocated with the **`new`** operator, but there is no explicit deallocation operator. Garbage collection is used for storage reclamation. Like many other language features, although garbage collection avoids some serious problems, such as dangling pointers, it can cause other problems. One such difficulty arises because the garbage collector deallocates, or reclaims the storage occupied by an object, but it does no more. For example, if an object has access to some resource other than heap memory, such as a file or a lock on a shared resource, the garbage collector does not reclaim these. For these situations, Java allows the inclusion of a special method, **`finalize`**, which is related to a C++ destructor function.

A **`finalize`** method is implicitly called when the garbage collector is about to reclaim the storage occupied by the object. The problem with **`finalize`** is

that the time it (and the garbage collector) will run cannot be forced or even predicted. The alternative to using **`finalize`** to reclaim resources held by an object about to be garbage collected is to include a method that does the reclamation. The only problem with this is that all clients of the objects must be aware of this method and remember to call it.

12.4.3.2 Inheritance

In Java, a method can be defined to be **`final`**, which means that it cannot be overridden in any descendant class. When the **`final`** reserved word is specified on a class definition, it means the class cannot be subclassed. All of the methods in a final class are implicitly final, which means that the bindings of method calls to the methods of the class are statically bound.

The advantage of defining a class to be final is that no changes to the class are allowed. For example, `String` is a final class and because of that any method that receives a `String` reference in a parameter can depend on the stability of the meaning of `String`'s methods. The disadvantage is that defining a class to be final disallows reuses that require even minor modifications.

Java includes the annotation `@Override`, which informs the compiler to check to determine whether the following method overrides a method in an ancestor class. If it does not, the compiler issues an error message.

Like C++, Java requires that parent class constructor be called before the subclass constructor is called. If parameters are to be passed to the parent class constructor, that constructor must be explicitly called, as in the following example:

```
super(100, true);
```

If there is no explicit call to the parent class constructor, the compiler inserts a call to the zero-parameter constructor in the parent class.

Java does not support the private derivations of C++. One can surmise that the Java designers believed that subclasses should be subtypes, which they

are not when private derivations are supported. Thus, they did not include them. So, Java's subclasses can be subtypes.

Early versions of Java included a collection, `Vector`, which included a long list of methods for manipulating data in a collection construct. These versions of Java also included a subclass of `Vector`, `Stack`, which added methods for push and pop operations. Unfortunately, because Java does not have private derivation, all of the methods of `Vector` were also visible in the `Stack` class, which made `Stack` objects liable to a variety of operations that could invalidate those objects.

Java directly supports only single inheritance. However, it includes a kind of abstract class, called an **interface**, which provides partial support for multiple inheritance. An interface definition is similar to a class definition, except that it can contain only named constants and method declarations (not definitions). It cannot contain constructors, nonabstract methods, or variable declarations. So, an interface is no more than what its name indicates—it defines only the specification of a class. (Recall that a C++ abstract class can have instance variables and all but one of the methods can be completely defined.) A class does not inherit an interface; it implements it. In fact, a class can implement any number of interfaces. To implement an interface, the class must implement all of the methods whose specifications (but not bodies) appear in the interface definition.

An interface can be used to simulate multiple inheritance. A class can be derived from a class and implement an interface, with the interface taking the place of a second parent class. This is sometimes called **mixin inheritance,** because the constants and methods of the interface are mixed in with the methods and data inherited from the superclass, as well as any new data and/or methods defined in the subclass.

One more interesting capability of interfaces is that they provide another kind of polymorphism. This is because interfaces can be treated as types. For example, a method can specify a formal parameter that is an interface. Such a formal parameter can accept an actual parameter of any class that implements the interface, making the method polymorphic.

A nonparameter variable also can be declared to be of the type of an

interface. Such a variable can reference any object of any class that implements the interface.

One of the problems with multiple inheritance occurs when a class is derived from two parent classes and both define a public method with the same name and protocol. This problem is avoided with interfaces. Although a class that implements an interface must provide definitions for all of the methods specified in the interface, if the class and the interface both include methods with the same name and protocol, the class need not reimplement that method. So, the method name conflicts that can occur with multiple inheritance cannot occur with single inheritance and interfaces. Furthermore, variable name conflicts are completely avoided because interfaces cannot define variables.

An interface is not a replacement for multiple inheritance, because in multiple inheritance there is code reuse, while interfaces provide no code reuse. This is an important difference, because code reuse is one of the primary benefits of inheritance. Java provides one way to partially avoid this deficiency. One of the implemented interfaces could be replaced by an abstract class, which could include code that could be inherited, thereby providing some code reuse.

One problem with interfaces being a replacement for multiple inheritance is the following: If a class attempts to implement two interfaces and both define methods that have the same name and protocol, there is no way to implement both in the class.

As an example of an interface, consider the `sort` method of the standard Java class, `Array`. Any class that uses this method must provide an implementation of a method to compare the elements to be sorted. The generic `Comparable` interface provides the protocol for this comparing method, which is named `compareTo`. The code for the `Comparable` interface is as follows:

```
public interface Comparable <T> {
    public int compareTo(T b);
}
```

The `compareTo` method must return a negative integer if the object through which it is called belongs before the parameter object, zero if they are equal,

and a positive integer if the parameter belongs before the object through which `compareTo` was called. A class that implements the `Comparable` interface can sort the contents of any array of objects of the generic type, as long as the implemented `compareTo` method for the generic type is implemented and provides the appropriate value. Interfaces have become a common substitute for multiple inheritance. Some form of interfaces are now part of C#, Swift, Ruby, and Ada.

In addition to interfaces, Java also supports abstract classes, similar to those of C++. The abstract methods of a Java abstract class are represented as just the method's header, which includes the **`abstract`** reserved word. The abstract class is also marked **`abstract`**. Of course, abstract classes cannot be instantiated.

Chapter 14 illustrates the use of interfaces in Java event handling.

12.4.3.3 Dynamic Binding

In C++, a method must be defined as virtual to allow dynamic binding. In Java, all method calls are dynamically bound unless the called method has been defined as **`final`**, in which case it cannot be overridden and all bindings are static. Static binding is also used if the method is **`static`** or **`private`**, both of which disallow overriding.

12.4.3.4 Nested Classes

Java has several varieties of nested classes, all of which have the advantage of being hidden from all classes in their package, except for the nesting class. Nonstatic classes that are nested directly in another class are called **inner classes**. Each instance of an inner class must have an implicit pointer to the instance of its nesting class to which it belongs. This gives the methods of the nested class access to all of the members of the nesting class, including the private members. Static nested classes do not have this pointer, so they cannot access members of the nesting class. Therefore, static nested classes in

Java are like the nested classes of C++.

Though it seems odd in a static-scoped language, the members of the inner class, even the private members, are accessible in the outer class. Such references must include the variable that references the inner class object. For example, suppose the outer class creates an instance of the inner class with the following statement:

```
myInner = this.new Inner();
```

Then, if the inner class defines a variable named `sum`, it can be referenced in the outer class as `myInner.sum`.

An instance of a nested class can only exist within an instance of its nesting class. Nested classes can also be anonymous. Anonymous nested classes have complex syntax but are really only an abbreviated way to define a class that is used from just one location. An example of an anonymous nested class appears in Chapter 14.

A **local nested class** is defined in a method of its nesting class. Local nested classes are never defined with an access specifier (**`private`** or **`public`**). Their scope is always limited to their nesting class. A method in a local nested class can access the variables defined in its nesting class and the **`final`** variables defined in the method in which the local nested class is defined. The members of a local nested class are visible only in the method in which the local nested class is defined.

12.4.3.5 Evaluation

Java's design for supporting object-oriented programming is similar to that of C++, but it employs more consistent adherence to object-oriented principles. Java does not allow parentless classes and uses dynamic binding as the "normal" way to bind method calls to method definitions. This, of course, increases execution time slightly over languages in which many method bindings are static. At the time this design decision was made, however, most Java programs were interpreted, so interpretation time made the extra binding time insignificant. Access controls for the contents of a class definition are

rather simple when compared with the jungle of access controls of C++, ranging from derivation controls to friend functions. Finally, Java uses interfaces to provide a form of support for multiple inheritance, which does not have all of the drawbacks of actual multiple inheritance.

12.4.4 C#

C#'s support for object-oriented programming is similar to that of Java.

12.4.4.1 General Characteristics

C# includes both classes and structs, with the classes being very similar to Java's classes and the structs being somewhat less powerful constructs. One important difference is that structs are value types; that is, they are stack dynamic. This could cause the problem of object slicing, but this is prevented by the restriction that structs cannot be subclassed. More details of how C# structs differ from its classes appeared in Chapter 11.

12.4.4.2 Inheritance

C# uses the syntax of C++ for defining classes. For example,

```
public class  NewClass : ParentClass { . . . }
```

A method inherited from the parent class can be replaced in the derived class by marking its definition in the subclass with **new**. The **new** method hides the method of the same name in the parent class to normal access. However, the parent class version can still be called by prefixing the call with **base**. For example,

```
base.Draw();
```

As with Java, subclasses can be subtypes. C#'s support for interfaces is the same as that of Java. It does not support multiple inheritance.

12.4.4.3 Dynamic Binding

To allow dynamic binding of method calls to methods in C#, both the base method and its corresponding methods in derived classes must be specially marked. The base class method must be marked with **`virtual`**, as in C++. To make clear the intent of a method in a subclass that has the same name and protocol as a virtual method in an ancestor class, C# requires that such methods be marked **`override`** if they are to override the parent class virtual method.13 For example, the C# version of the C++ `Shape` class that appears in Section 12.4.2.3 is as follows:

13. Recall that this can be specified in Java with the annotation `@Override`.

```
public class Shape {
  public virtual void Draw() { . . . }
  . . .
}
public class Circle : Shape {
  public override void Draw() { . . . }
  . . .
}
public class Rectangle : Shape {
  public override void Draw() { . . . }
  . . .
}
public class Square : Rectangle {
  public override void Draw() { . . . }
  . . .
}
```

C# includes abstract methods similar to those of C++, except that they are specified with different syntax. For example, the following is a C# abstract method:

```
abstract public void Draw();
```

A class that includes at least one abstract method is an abstract class, and every abstract class must be marked **`abstract`**. Abstract classes cannot be instantiated. It follows that any subclass of an abstract class that will be instantiated must implement all abstract methods that it inherits.

As with Java, all C# classes are ultimately derived from a single root class, `Object`. The `Object` class defines a collection of methods, including `ToString`, `Finalize`, and `Equals`, which are inherited by all C# types.

12.4.4.4 Nested Classes

A C# class that is directly nested in a class behaves like a Java static nested class (which is like a nested class in C++). Like C++, C# does not support nested classes that behave like the nonstatic nested classes of Java.

12.4.4.5 Evaluation

Because C# is a recently designed C-based object-oriented language, one should expect that its designers learned from their predecessors and duplicated the successes of the past and remedied some of the problems. One result of this, coupled with the few problems with Java, is that the differences between C#'s support for object-oriented programming and that of Java are relatively minor. The availability of structs in C#, which Java does not have, can be considered an improvement. Like that of Java, C#'s support for object-oriented programming is simpler than that of C++, which many consider an improvement.

12.4.5 Ruby

As stated previously, Ruby is a pure object-oriented programming language in the sense of Smalltalk. Virtually everything in the language is an object and all computation is accomplished through message passing. Although programs have expressions that use infix operators and therefore have the same appearance as expressions in languages like Java, those expressions actually are evaluated through message passing. As is the case with Smalltalk, when one writes `a + b`, it is evaluated by sending the message `+` to the object referenced by `a`, passing a reference to the object `b` as a parameter. In other words, `a + b` is implemented as `a.+ b`.

12.4.5.1 General Characteristics

Ruby class definitions differ from those of languages such as C++ and Java in that they are executable. Because of this, they are allowed to remain open during execution. A program can add members to a class any number of times, simply by providing secondary definitions of the class that include the new members. During execution, the current definition of a class is the union of all definitions of the class that have been executed. Method definitions are also executable, which allows a program to choose between two versions of a method definition during execution, simply by putting the two definitions in the then and else clause of a selection construct.

Ruby objects are created with **`new`**, which implicitly calls a constructor. The usual constructor in a Ruby class is named `initialize`. A constructor in a subclass can initialize the data members of the parent class that have setters defined. This is done by calling **`super`** with the initial values as actual parameters. **`super`** calls the method in the parent class that has the same name as the method in which the call to **`super`** appears.

Ruby classes can be nested, but the nested class has no special access to the variables or methods of the nesting class.

All variables in Ruby are references to objects, and all are typeless. Recall that the names of all instance variables in Ruby begin with an at sign (`@`).

In a clear departure from the other common programming languages, access control in Ruby is different for data than it is for methods. All instance data has private access by default, and that cannot be changed. Therefore, no subclass in Ruby is a subtype. If external access to an instance variable is required, accessor methods must be defined. For example, consider the following skeletal class definition:

```
class MyClass
# A constructor
  def initialize
    @one = 1
    @two = 2
  end
```

```
# A getter for @one
  def one
    @one
  end
# A setter for @one
  def one=(my_one)
    @one = my_one
  end
end  # of class MyClass
```

The equal sign (=) attached to the name of the setter method means that its variable is assignable. So, all setter methods have equal signs attached to their names. The body of the `one` getter method illustrates the Ruby design of methods returning the value of the last expression evaluated when there is no return statement. In this case, the value of `@one` is returned.

Because getter and setter methods are so frequently needed, Ruby provides shortcuts for creating them. If one wants a class to have getter methods for the two instance variables, `@one` and `@two`, those getters can be specified with the single statement in the class:

```
attr_reader :one, :two
```

`attr_reader` is actually a function call, using `:one` and `:two` as the actual parameters. Preceding a variable with a colon (`:`) causes the variable name to be used, rather than dereferencing it to the object to which it refers. Instead of passing a value or an address, the text of the variable's name is passed. This is exactly how macro parameters are passed.

The function that similarly creates setters is called `attr_writer`. This function has the same parameter profile as `attr_reader`.

The functions for creating getter and setter methods are so named because they provide the protocol for objects of the class, which then are called **attributes**. So, the attributes of a class define the data interface (the data made public through accessor methods) to objects of the class.

Class variables, which are specified by preceding their names with two at signs (@@), are private to the class and its instances. That privacy cannot be changed. Also, unlike global and instance variables, class variables must be

initialized before they are used.

12.4.5.2 Inheritance

Subclasses are defined in Ruby using the less-than symbol (<), rather than the colon of C++. For example,

```
class MySubClass < BaseClass
```

One distinct thing about the method access controls of Ruby is that they can be changed in a subclass, simply by calling the access control functions. This means that two subclasses of a base class can be defined so that objects of one of the subclasses can access a method defined in the base class, but objects of the other subclass cannot. Also, this allows one to change the access of a publicly accessible method in the base class to a privately accessible method in the subclass.

12.4.5.3 Dynamic Binding

Support for dynamic binding in Ruby is the same as it is in Smalltalk. - Variables are not typed; rather, they are all references to objects of any class. So, all variables are polymorphic and all bindings of method calls to methods are dynamic.

12.4.5.4 Evaluation

Because Ruby is an object-oriented programming language in the purest sense, its support for object-oriented programming is obviously adequate. However, access control to class members is weaker than that of C++. Ruby does not support abstract classes or interfaces, although its mixins are closely related to interfaces. Finally, in large part because Ruby is interpreted, its execution efficiency is far worse than that of the compiled languages.

Table 12.1 summarizes how the designers of the languages in this section chose to deal with the design issues described in Section 12.3.

Table 12.1 Designs

DESIGN ISSUE/ LANGUAGE	SMALLTALK	C++	JAVA	C#	RUBY
Exclusivity of objects	All data are objects	Primitive types plus objects	Primitive types plus objects	Primitive types plus objects	All data are objects
Are subclasses subtypes?	They can be and usually are	They can be and usually are if the derivation is public	They can be and usually are	They can be and usually are	No subclasses are subtypes
Single and multiple inheritance	Single only	Both	Single only, but some effects with interfaces	Single only, but some effects with interfaces	Single only, but some effects with modules
Allocation and deallocation of objects	All objects are heap allocated; allocation is explicit and deallocation is implicit	Objects can be static, stack dynamic, or heap dynamic; allocation and deallocation are explicit	All objects are heap dynamic; allocation is explicit and deallocation is implicit	All objects are heap dynamic; allocation is explicit and deallocation is implicit	All objects are heap dynamic; allocation is explicit and deallocation is implicit
Dynamic and static binding	All method bindings are dynamic	Method binding can be either	Method binding can be either	Method binding can be either	All method bindings are dynamic
Nested classes?	No	Yes	Yes	Yes	Yes
Initialization	Constructors must be explicitly called	Constructors are implicitly called	Constructors are implicitly called	Constructors are implicitly called	Constructors are implicitly called

12.5 Implementation of Object-Oriented Constructs

There are at least two parts of language support for object-oriented programming that pose interesting questions for language implementers: storage structures for instance variables and the dynamic bindings of messages to methods. In this section, we provide a brief look at these.

12.5.1 Instance Data Storage

In C++, classes are defined as extensions of C's record structures—structs. This similarity suggests a storage structure for the instance variables of class instances—that of a record. This form of this structure is called a **class instance record (CIR)**. The structure of a CIR is static, so it is built at compile time and used as a template for the creation of the data of class instances. Every class has its own CIR. When a derivation takes place, the CIR for the subclass is a copy of that of the parent class, with entries for the new instance variables added at the end.

Because the structure of the CIR is static, access to all instance variables can be done as it is in records, using constant offsets from the beginning of the CIR instance. This makes these accesses as efficient as those for the fields of records.

12.5.2 Dynamic Binding of Method Calls to Methods

Methods in a class that are statically bound need not be involved in the CIR for the class. However, methods that will be dynamically bound must have entries in this structure. Such entries could simply have a pointer to the code

of the method, which must be set at object creation time. Calls to a method could then be connected to the corresponding code through this pointer in the CIR. The drawback to this technique is that every instance would need to store pointers to all dynamically bound methods that could be called from the instance.

Notice that the list of dynamically bound methods that can be called from an instance of a class is the same for all instances of that class. Therefore, the list of such methods must be stored only once. So the CIR for an instance needs only a single pointer to that list to enable it to find called methods. The storage structure for the list is often called a **virtual method table (vtable)**. Method calls can be represented as offsets from the beginning of the vtable. Polymorphic variables of an ancestor class always reference the CIR of the correct type object, so getting to the correct version of a dynamically bound method is assured. Consider the following Java example, in which all methods are dynamically bound:

```
public class A {
  public int a, b;
  public void draw() { . . . }
  public int area() { . . . }
}
public class B extends A {
  public int c, d;
  public void draw() { . . . }
  public void sift() { . . . }
}
```

The CIRs for the `A` and `B` classes, along with their vtables, are shown in Figure 12.7. Notice that the method pointer for the `area` method in B's vtable points to the code for A's `area` method. The reason is that B does not override A's `area` method, so if a client of B calls `area`, it is the `area` method inherited from A. On the other hand, the pointers for `draw` and `sift` in B's vtable point to B's `draw` and `sift`. The `draw` method is overridden in B and `sift` is defined as an addition in B.

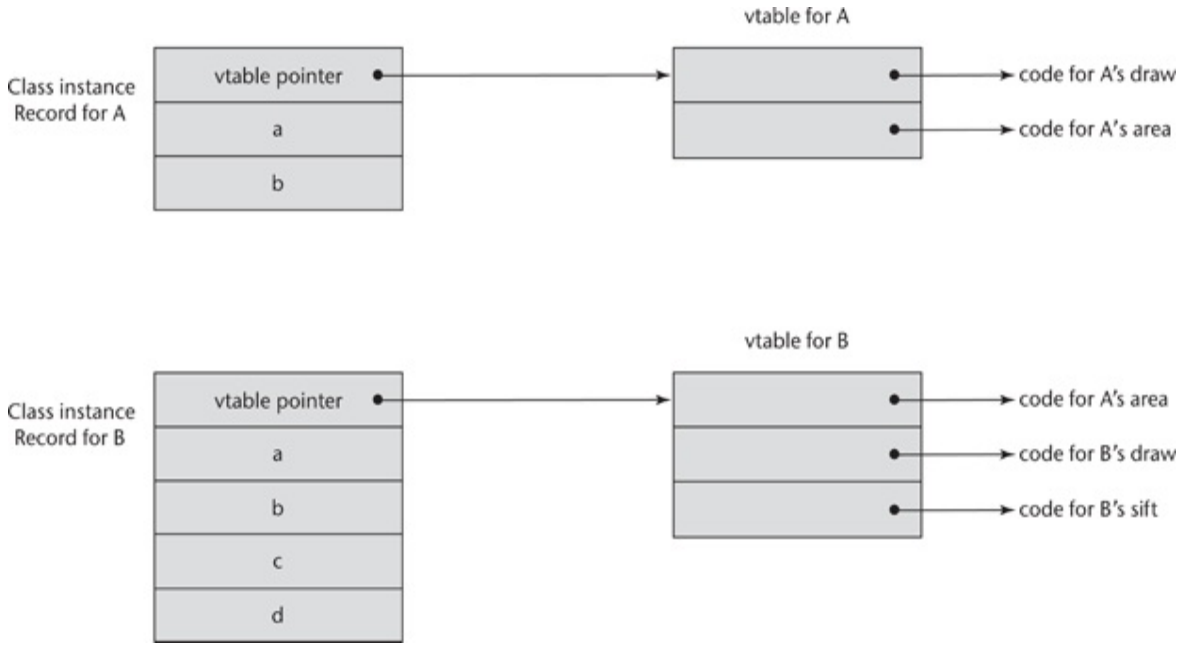


Figure 12.7 An example of the CIRs with single inheritance

Figure 12.7 Full Alternative Text

Multiple inheritance complicates the implementation of dynamic binding. Consider the following three C++ class definitions:

```
class A {
  public:
     int a;
     virtual void fun() { . . . }
     virtual void init() { . . . }
};
 class B {
   public:
     int b;
     virtual void  sum() { . . . }
};
class C : public A, public B {
  public:
    int  c;
    virtual void fun() { . . . }
```

```
    virtual void dud() { . . . }
};
```

The C class inherits the variable a and the init method from the A class. It redefines the fun method, although both its fun and that of the parent class A are potentially visible through a polymorphic variable (of type A). From B, C inherits the variable b and the sum method. C defines its own variable, c, and defines an uninherited method, dud. A CIR for C must include A's data, B's data, and C's data, as well as some means of accessing all visible methods. Under single inheritance, the CIR would include a pointer to a vtable that has the addresses of the code of all visible methods. With multiple inheritance, however, it is not that simple. There must be at least two different views available in the CIR—one for each of the parent classes, one of which includes the view for the subclass, C. This inclusion of the view of the subclass in the parent class's view is just as in the implementation of single inheritance.

There must also be two vtables: one for the A and C view and one for the B view. The first part of the CIR for C in this case can be the C and A view, which begins with a vtable pointer for the methods of C and those inherited from A, and includes the data inherited from A. Following this in C's CIR is the B view part, which begins with a vtable pointer for the virtual methods of B, which is followed by the data inherited from B and the data defined in C. The CIR for C is shown in Figure 12.8.

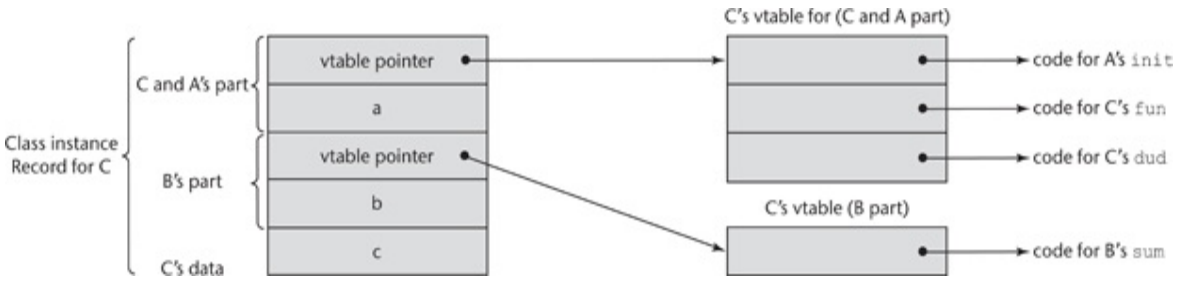


Figure 12.8 An example of a subclass CIR with multiple parents

Figure 12.8 Full Alternative Text

12.6 Reflection

A discussion of reflection is not a perfect fit into a chapter on object orientation, but it is even a worse fit into any other chapter of this book. So, this is where we put it.

12.6.1 Introduction

In general, the later bindings take place in a programming language, the more flexible the language is. For example, the late binding of data types in scripting languages and functional languages allows their programs to be more generic than those in the static-typed languages. Likewise, the dynamic binding of method calls to methods that is part of the object-oriented languages allows their programs to be easier to maintain and extend. Among other things, reflection provides the possibility of late binding of calls to methods that are outside the inheritance hierarchy of the calling code.

12.6.2 What Is Reflection?

A programming language that supports reflection allows its programs to have run-time access to their types and structure and to be able to dynamically modify their behavior. To allow a program to examine its types and structure, that information must be gathered by the compiler or interpreter and made available to the program. Just as information about the structure of a database is called metadata, the types and structure of a program are called **metadata**. The process of a program examining its metadata is called **introspection**. A program can modify its behavior dynamically in several different ways: it could change its metadata directly, it could use the metadata, or it could intercede in the execution of the program. The first of these is complicated; the second is less complex and is common among languages; the third often is called **intercession**.

Some of the primary uses of reflection are in the construction of software tools. A class browser needs to enumerate the classes of a program. Visual Integrated Development Environments can use type information to assist a developer in building type-correct code. Debuggers must be able to examine private fields and methods of classes. Test systems need to be able to discover all of the methods of a class to be sure that test data drives all of them.

To illustrate a relatively simple and common use of reflection, we pose the following problem. A zoo has a large area devoted to birds. The flight cage for each species includes a plaque that provides general information about the inhabitant's species. Included on the plaque is a small screen onto which a visitor with special interest in the species can enter his or her entrance ticket number. At the exit to the bird exhibit, the visitor can again enter his or her ticket number on a small screen, which causes a computer to print pictures of the birds for which the visitor previously indicated particular interest. The computer system that supports these activities has an object that includes a method that draws a picture of its bird for each of the birds on display. This seems simple enough. When a visitor selects a bird at its flight cage, the system places a reference to an object associated with that bird in a list. At the exit of the exhibit, the system calls the draw method of each object in the visitor's list. The process is complicated by the fact that the zoo supplies only some of the bird objects. Some of them are purchased from third-party vendors and some are donated by zoo benefactors. Because of the multiple sources of the bird objects, they do not have a common base class (other than Object) and do not implement a common interface, so what type references can be saved? One obvious solution is to make each bird object the subclass of a base class. References of the base class type could be stored in the list and dynamic binding could be used to invoke the draw methods. The drawback of this approach is that every bird class would need to be modified to make the new classes subclasses of the common base class. It would be better if the new bird classes could simply be added to a code file without modification. Another possible solution would be to use instance of and casting to determine the concrete types of the references. This would add much code to the system, increasing its complexity and cost of maintenance. A better solution is to use the dynamic binding that is possible with reflection.

12.6.3 Reflection in Java

Java provides limited support for reflection. The primary class of the metadata is defined in the namespace, `java.lang.Class`.[14] This class has the unfortunately confusing name, `Class`. The Java run-time system instantiates an instance of `Class` for each object in the program. The `Class` class provides a collection of methods to examine the type information and members of the program objects. `Class` is the access point for all of the reflection API.

14. It is so named because its instantiations are themselves classes.

If the program has a reference to an object (not a primitive), the `Class` object of that object can be obtained by calling its `getClass` method. All classes inherit `getClass` from `Object`, from which all objects descend. Consider the following examples:

```
float[] totals = new float[100];
Class fltlist = totals.getClass();
Class stg = "hello".getClass();
```

The value of the variable `fltlist` will be the `Class` object of the `totals` array object. The value of `stg` will be the `Class` object of `String` (because `"hello"` is an instance of `String`).

If there is no object of a class, its `Class` object can be obtained through the class' name by attaching `.class` to the name. For example, we could have the following:

```
Class stg = String.class;
```

If the class has no name, its `Class` object can still be obtained by attaching `.class` to the class definition. For example, consider the following:

```
Class intmat = int[][].class;
```

The `.class` modifier can also be attached to primitive types. Although `float.getClass()` is illegal, `float.class` is not.

There are four methods to get the `Class` of a method. The `getMethod` method searches a class to find a specific public method defined in the class or inherited by the class. The `getMethods` method returns an array of all of the public methods defined in a class or inherited by the class. The `getDeclaredMethod` method searches for a specific method declared in a class, including private methods. The `getDeclaredMethods` method returns all of the methods defined in a class.

If the `Class` object of an object is known and a particular method defined by the class of the object is found, that method can be called through the `Method` object of the method with the `invoke` method. For example, if the `Method` object named `method` is found with `getMethod`, it can be called with the following:

```
method.invoke(...);
```

We can now develop a solution in Java for the problem posed in Section 12.6.2. The heart of this application is a class that defines a method that is passed an `Object` reference. The method determines the class of the passed reference, finds a `draw` method of that class, and calls that method. The solution class is tested with a second class, `ReflectTest`, which creates an array of three `Object` references to classes that represent three different birds. Each of these defines a `draw` method that, when called, displays a message indicating that it was called. Then the test calls the class method, passing the elements of the array of references.

The caller method can raise three different exceptions, each of which is handled in the method.

```
// A project to illustrate dynamic method calling
// using reflection in Java
package reflect;
import java.lang.reflect.*;
// A class to test the Reflect class
//  Creates three objects that represent different birds
//  and calls a method that dynamically calls the draw
//  methods of the three bird classes
public class ReflectTest {
   public static void main(String[] args) {
        Object[] birdList = new Object[3];
        birdList[0] = new Bird1();
```

```
        birdList[1] = new Bird2();
        birdList[2] = new Bird3();
        Reflect.callDraw(birdList[2]);
        Reflect.callDraw(birdList[0]);
        Reflect.callDraw(birdList[1]);
       }
}
// A class to define the method that dynamically calls the
//  methods of a passed class object
class Reflect {
    public static void callDraw(Object birdObj) {
        Class cls = birdObj.getClass();
        try {
            // Find the draw method of the given class
            Method method = cls.getMethod("draw");
            // Dynamically call the method
            method.invoke(birdObj);
        }
        // In case the given class does not support draw
        catch (NoSuchMethodException e) {
            throw new IllegalArgumentException (
                cls.getName() + "does not support draw");
        }
        // In case the callDraw cannot call draw
        catch (IllegalAccessException e) {
            throw new IllegalArgumentException (
                "Insufficient access permissions to call" +
                "draw in class " + cls.getName());
        }
        // In case draw throws an exception
        catch (InvocationTargetException e) {
            throw new RuntimeException(e);
        }
      }
}
class Bird1 {
  public void draw() {
      System.out.println("This is draw from Bird1");
  }
}
class Bird2 {
    public void draw() {
        System.out.println("This is draw from Bird2");
    }
}
class Bird3 {
    public void draw() {
        System.out.println("This is draw from Bird3");
```

```
    }
]
```

The output of this program is as follows:

```
This is the draw from Bird3
This is the draw from Bird1
This is the draw from Bird2
```

12.6.4 Reflection in C#

Support for reflection in C# is similar to that of Java, with a few important differences. In C#, as in all .NET languages, the compiler places the intermediate code, written in Common Intermediate Language (CIL), in an assembly, which could include several files. An assembly also contains an assembly version number and the metadata for all classes defined in the assembly, as well as for all external classes it uses.

Instead of the `java.lang.Class` namespace, `System.Type` is used in .NET; instead of `java.lang.reflect`, `System.Reflection` is used. Rather than the `getClass` method, `getType` is used to get the class of an instance. Also, the .NET languages use the `typeof` operator in place of the `.class` field used in Java. Following is a C# version of the Java project shown above:

```
using System;
using System.Reflection;
namespace TestReflect
{
// A project to illustrate dynamic method calling
// using reflection in C#
// A class to test the Reflect class
//  Creates three objects that represent different birds
//  and calls a method that dynamically calls the draw
//  methods of the three bird classes
public class ReflectTest {
   public static void Main(String[] args) {
        Object[] birdList = new Object[3];
        birdList[0] = new Bird1();
        birdList[1] = new Bird2();
        birdList[2] = new Bird3();
        Reflect.callDraw(birdList[2]);
```

```
        Reflect.callDraw(birdList[0]);
        Reflect.callDraw(birdList[1]);
        }
}
// A class to define the method that dynamically calls the
//  methods of a passed class object
class Reflect {
    public static void callDraw(Object birdObj) {
        Type typ = birdObj.GetType();
            // Find the draw method of the given class
            MethodInfo method = typ.GetMethod("draw");
            // Dynamically call the method
            method.Invoke(birdObj, null);
    }
}
class Bird1 {
    public void draw() {
        Console.WriteLine("This is draw from Bird1");
    }
}
class Bird2 {
    public void  draw() {
        Console.WriteLine("This is draw from Bird2");
    }
}
class Bird3 {
    public void draw() {
        Console.WriteLine("This is draw from Bird3");
    }
}
}
```

Our simple example of dynamic method binding shows just one of the many uses of reflection.

In addition to the methods and fields of a class, the following program elements can be accessed with reflection in both Java and C#: class modifiers, such as public, static, and final, constructors, method parameter types, and implemented interfaces. Also, a description of the inheritance path of a class can be introspected. In C#, but not Java, the names of the formal parameters of methods can be discovered.

One significant difference between Java's reflection and that of C# is the `System.Reflection.Emit` namespace, which is part of .NET. This

namespace provides the ability to create CIL code and an assembly to house that code. Java provides no such capability, although it can be done with tools from other suppliers.

Although reflection adds a variety of capabilities to the static-typed languages Java and C#, the user of reflection must be aware of its downsides:

- Performance nearly always suffers with the use of reflection. Resolving types, methods, and fields at run time are not part of the cost of running nonreflective code. Also, when types are dynamically resolved, some optimizations cannot be done on the code.

- Reflection exposes private fields and methods, which violate the rules of abstraction and information hiding, and also may result in unexpected side effects and adversely affect portability.

- Although the advantage of early type checking is widely accepted, the late binding that is possible with reflection obviously negates that advantage.

- Some reflective operations may not work when the code is run under a security manager, also making it non-portable. One such security environment is that of running applets. In most cases, if a problem can be solved without reflection, reflection should not be used.

Reflection is an integral part of most dynamically typed languages. In LISP, reflection is routinely used and the dynamic construction and execution of code is not uncommon. In other interpreted languages, such as JavaScript, Perl, and Python, the symbol table is kept during interpretation, providing all useful type information.

In Python, for example, the `type` method returns the type of a given value. For example, `type([7, 14, 21])` is `list`. The `isinstance` method returns a Boolean value if its first parameter has the type named in its second parameter. For example, `isinstance(17, int)` returns `True`. The `callable` function is used to determine whether an expression returns a function object. The `dir` function returns the list of attributes, both data and methods, of its parameter object.

SUMMARY

Object-oriented programming is based on three fundamental concepts: abstract data types, inheritance, and dynamic bindinwg. Object-oriented programming languages support the paradigm with classes, methods, objects, and message passing.

The discussion of object-oriented programming languages in this chapter revolves around seven design issues: exclusivity of objects, subclasses and subtypes, type checking and polymorphism, single and multiple inheritance, dynamic binding, explicit or implicit deallocation of objects, and nested classes.

Smalltalk is a pure object-oriented language—everything is an object and all computation is accomplished through message passing. All type checking and binding of messages to methods is dynamic, and all inheritance is single. Smalltalk has no explicit deallocation operation.

C++ provides support for data abstraction, inheritance, and optional dynamic binding of messages to methods, along with all of the conventional features of C. This means that it has two distinct type systems. C++ provides multiple inheritance and explicit object deallocation. It includes a variety of access controls for the entities in classes, some of which prevent subclasses from being subtypes. Both constructor and destructor methods can be included in classes; both are usually implicitly called.

While Smalltalk’s dynamic type binding provides somewhat more programming flexibility than the hybrid language C++, it is far less efficient.

Unlike C++, Java is not a hybrid language; it is meant to support only object-oriented programming. Java has both primitive scalar types and classes. All objects are allocated from the heap and are accessed through reference variables. There is no explicit object deallocation operation—garbage collection is used. The only subprograms are methods, and they can be called only through objects or classes. Only single inheritance is directly supported,

although a kind of multiple inheritance is possible using interfaces. All binding of messages to methods is dynamic, except in the case of methods that cannot be overridden. In addition to classes, Java includes packages as a second encapsulation construct.

C#, which is based on C++ and Java, supports object-oriented programming. Objects can be instantiated from either classes or structs. The struct objects are stack dynamic and do not support inheritance. Methods in a derived class can call the hidden methods of the parent class by including **`base`** on the method name. Methods that can be overridden must be marked **`virtual`**, and the overriding methods must be marked with **`override`**. All classes (and all primitives) are derived from `Object`.

Ruby is an object-oriented scripting language in which all data are objects. As with Smalltalk, all objects are heap allocated and all variables are typeless references to objects. All constructors are named `initialize`. All instance data are private, but getter and setter methods can be easily included. The collection of all instance variables for which access methods have been provided forms the public interface to the class. Such instance data are called attributes. Ruby classes are dynamic in the sense that they are executable and can be changed at any time. Ruby supports only single inheritance.

The instance variables of a class are stored in a CIR, the structure of which is static. Subclasses have their own CIRs, as well as the CIR of their parent class. Dynamic binding is supported with a virtual method table, which stores pointers to specific methods. Multiple inheritance greatly complicates the implementation of CIRs and virtual method tables.

Reflection is a process by which a program can access its classes and types and possibly dynamically change them to affect program behavior. One of the primary uses of reflection is in the construction of software tools, such as visual program construction tools, debuggers, and test systems. The class and type information, called metadata, is collected by the compiler or interpreter for the language. In Java, the class information, such as the methods of the class, are available in the `Class` object of a class. Support for reflection in C#, which is similar to that of Java, is available in the `System.Reflection` namespace.

REVIEW QUESTIONS

1. Describe the three characteristic features of object-oriented languages.
2. What is the difference between a class variable and an instance variable?
3. What is multiple inheritance?
4. What is a polymorphic variable?
5. What is an overriding method?
6. Describe a situation where dynamic binding is a great advantage over static binding.
7. What is a virtual method?
8. What is an abstract method? What is an abstract class?
9. Describe briefly the seven design issues used in this chapter for object-oriented languages.
10. What is a nesting class?
11. What is the message protocol of an object?
12. From where are Smalltalk objects allocated?
13. Explain how Smalltalk messages are bound to methods. When does this take place?
14. What type checking is done in Smalltalk? When does it take place?
15. What kind of inheritance, single or multiple, does Smalltalk support?
16. What are the two most important effects that Smalltalk has had on computing?

17. In essence, all Smalltalk variables are of a single type. What is that type?

18. From where can C++ objects be allocated?

19. How are C++ heap-allocated objects deallocated?

20. Are all C++ subclasses subtypes? If so, explain. If not, why not?

21. Under what circumstances is a C++ method call statically bound to a method?

22. What drawback is there to allowing designers to specify which methods can be statically bound?

23. What are the differences between private and public derivations in C++?

24. What is a `friend` function in C++?

25. What is a pure virtual function in C++?

26. How are parameters sent to a superclass's constructor in C++?

27. What is the single most important practical difference between Smalltalk and C++?

28. How is the type system of Java different from that of C++?

29. From where can Java objects be allocated?

30. What is boxing?

31. How are Java objects deallocated?

32. Are all Java subclasses subtypes?

33. How are superclass constructors called in Java?

34. Under what circumstances is a Java method call statically bound to a method?

35. In what way do overriding methods in C# syntactically differ from their counterparts in C++?

36. How can the parent version of an inherited method that is overridden in a subclass be called in that subclass in C#?

37. How does Ruby implement primitive types, such as those for integer and floating-point data?

38. How are getter methods defined in a Ruby class?

39. What access controls does Ruby support for instance variables?

40. What access controls does Ruby support for methods?

41. Are all Ruby subclasses subtypes?

42. Does Ruby support multiple inheritance?

43. What does reflection allow a program to do?

44. In the context of reflection, what is metadata?

45. What is introspection?

46. What is intercession?

47. What class in Java stores information about classes in a program?

48. For what is the Java name extension `.class` used?

49. What does the Java `getMethods` method do?

50. For what is the C# namespace `System.Reflection.Emit` used?

PROBLEM SET

1. What important part of support for object-oriented programming is missing in SIMULA 67?

2. Explain the principle of substitution.

3. Explain the ways subclasses can be created that are not subtypes.

4. Compare the dynamic binding of C++ and Java.

5. Compare the class entity access controls of C++ and Java.

6. Compare the multiple inheritance of C++ with that provided by interfaces in Java.

7. What is one programming situation where multiple inheritance has a significant advantage over interfaces?

8. Explain the two problems with abstract data types that are ameliorated by inheritance.

9. Describe the categories of changes that a subclass can make to its parent class.

10. Explain one disadvantage of inheritance.

11. Explain the advantages and disadvantages of having all values in a language be objects.

12. What exactly does it mean for a subclass to have an is-a relationship with its parent class?

13. Describe the issue of how closely the parameters of an overriding method must match those of the method it overrides.

14. Explain type checking in Smalltalk.

15. The designers of Java obviously thought it was not worth the additional efficiency of allowing any method to be statically bound, as is the case with C++. What are the arguments for and against the Java design?

16. What is the primary reason why all Java objects have a common ancestor?

17. What is the purpose of the **`finalize`** clause in Java?

18. What would be gained if Java allowed stack-dynamic objects as well as heap-dynamic objects? What would be the disadvantage of having both?

19. What are the differences between a C++ abstract class and a Java interface?

20. Explain why allowing a class to implement multiple interfaces in Java and C# does not create the same problems that multiple inheritance in C++ creates.

21. Study and explain the issue of why C# does not include Java's nonstatic nested classes.

22. Can you define a reference variable for an abstract class? What use would such a variable have?

23. Compare the access controls for instance variables in Java and Ruby.

24. Compare the type error detection for instance variables in Java and Ruby.

25. Explain the downsides of reflection.

PROGRAMMING EXERCISES

1. Rewrite the `single_linked_list`, `stack_2`, and `queue_2` classes in Section 12.5.2 in Java and compare the result with the C++ version in terms of readability and ease of programming.

2. Repeat Programming Exercise 1 using Ruby.

3. Design and implement a C++ program that defines a base class A, which has a subclass B, which itself has a subclass C. The A class must implement a method, which is overridden in both B and C. You must also write a test class that instantiates A, B, and C and includes three calls to the method. One of the calls must be statically bound to A's method. One call must be dynamically bound to B's method, and one must be dynamically bound to C's method. All of the method calls must be through a pointer to class A.

4. Write a program in C++ that calls both a dynamically bound method and a statically bound method a large number of times, timing the calls to both of the two. Compare the timing results and compute the difference of the time required by the two. Explain the results.

5. Repeat Programming Exercise 1 using Java, forcing static binding with **`final`**.

13 Concurrency

This chapter begins with introductions to the various kinds of concurrency at the subprogram, or unit level, and at the statement level. Included is a brief description of the most common kinds of multiprocessor computer architectures. Next, a lengthy discussion on unit-level concurrency is presented. This begins with a description of the fundamental concepts that must be understood before discussing the problems and challenges of language support for unit-level concurrency, specifically competition and cooperation synchronization. Next, the design issues for providing language support for concurrency are described. Following this is a detailed discussion of three major approaches to language support for concurrency: semaphores, monitors, and message passing. A pseudocode example program is used to - demonstrate how semaphores can be used. Ada and Java are used to illustrate monitors; for message passing, Ada is used. The Ada features that support concurrency are described in some detail. Although tasks are the focus,

protected objects (which are effectively monitors) are also discussed. Support for unit-level concurrency using threads in Java and C# is then discussed, including approaches to synchronization. This is followed by brief overviews of support for concurrency in several functional programming languages. The last section of the chapter is a brief discussion of statement-level concurrency, including an introduction to part of the language support provided for it in High-Performance Fortran.

13.1 Introduction

Concurrency in software execution can occur at four different levels: instruction level (executing two or more machine instructions simultaneously), statement level (executing two or more high-level language statements simultaneously), unit level (executing two or more subprogram units simultaneously), and program level (executing two or more programs simultaneously). Because no language design issues are involved with them, instruction-level and program-level concurrency are not discussed in this chapter. Concurrency at both the subprogram and the statement levels is discussed, with most of the focus on the subprogram level.

At first glance, concurrency may appear to be a simple concept, but it presents significant challenges to the programmer, the programming language designer, and the operating system designer (because much of the support for concurrency is provided by the operating system).

Concurrent control mechanisms increase programming flexibility. They were originally invented to be used for particular problems faced in operating systems, but they are required for a variety of other programming applications. One of the most commonly used programs is Web browsers, whose design is based heavily on concurrency. Browsers must perform many different functions at the same time, among them sending and receiving data from Web servers, rendering text and images on the screen, and reacting to user actions with the mouse and the keyboard. Most contemporary browsers use the extra core processors that are part of many contemporary personal computers to perform some of their processing, for example the interpretation of client-side scripting code. Another example is the software systems that are designed to simulate actual physical systems that consist of multiple concurrent subsystems. For all of these kinds of applications, the programming language (or a library or at least the operating system) must support unit-level concurrency.

Statement-level concurrency is quite different from concurrency at the unit level. From a language designer’s point of view, statement-level concurrency

is largely a matter of specifying how data should be distributed over multiple memories and which statements can be executed concurrently.

The goal of developing concurrent software is to produce scalable and portable concurrent algorithms. A concurrent algorithm is **scalable** if the speed of its execution increases when more processors are available. This is important because the number of processors sometimes increases with the new generations of machines. The algorithms must be portable because the lifetime of hardware is relatively short. Therefore, software systems should not depend on a particular architecture—that is, they should run efficiently on machines with different architectures.

The intention of this chapter is to discuss the aspects of concurrency that are most relevant to language design issues, rather than to present a definitive study of all of the issues of concurrency, including the development of concurrent programs. That would clearly be inappropriate for a book on programming languages.

13.1.1 Multiprocessor Architectures

A large number of different computer architectures have more than one processor and can support some form of concurrent execution. Before beginning to discuss concurrent execution of programs and statements, we briefly describe some of these architectures.

The first computers that had multiple processors had one general-purpose processor and one or more other processors, often called peripheral processors, that were used only for input and output operations. This architecture allowed those computers, which appeared in the late 1950s, to execute one program while concurrently performing input or output for that program or other programs.

By the early 1960s, there were machines that had multiple complete processors. These processors were used by the job scheduler of the operating

system, which distributed separate jobs from a batch-job queue to the separate processors. Systems with this structure supported program-level concurrency.

In the mid-1960s, some machines appeared that had several identical partial processors. These were fed instructions from a single instruction stream. For example, some machines had two or more floating-point multipliers, while others had two or more complete floating-point arithmetic units. The compilers for these machines were required to determine which instructions could be executed concurrently and to schedule these instructions accordingly. Systems with this structure supported instruction-level concurrency.

In 1966, Michael J. Flynn suggested a categorization of computer architectures defined by whether the instruction and data streams were single or multiple. The names of these were widely used from the 1970s to the early 2000s. The two categories that used multiple data streams are defined as follows: Computers that have multiple processors that execute the same instruction simultaneously, each on different data, are called Single Instruction, Multiple Data (SIMD) architecture computers. In an SIMD computer, each processor has its own local memory. One processor controls the operation of the other processors. Because all of the processors, except the controller, execute the same instruction at the same time, no synchronization is required in the software. Perhaps the most widely used SIMD machines are a category of machines called **vector processors**. They have groups of registers that store the operands of a vector operation in which the same instruction is executed on the whole group of operands simultaneously. Originally, the kinds of programs that could most benefit from this architecture were in scientific computation, an area of computing that is often the target of multiprocessor machines. However, SIMD processors are now used for a variety of application areas, among them graphics and video processing. Until recently, most supercomputers were vector processors.

Computers that have multiple processors that operate independently but whose operations can be synchronized are called Multiple Instruction, Multiple Data (MIMD) computers. Each processor in an MIMD computer

executes its own instruction stream. MIMD computers can appear in two distinct configurations: distributed and shared memory systems. The distributed MIMD machines, in which each processor has its own memory, can be either built in a single chassis or distributed, perhaps over a large area. The shared-memory MIMD machines obviously must provide some means of synchronization to prevent memory access clashes. Even distributed MIMD machines require synchronization to operate together on single programs. MIMD computers, which are more general than SIMD computers, support unit-level concurrency. The primary focus of this chapter is on language design for shared memory MIMD computers, which are often called **multiprocessors**.

With the advent of powerful but low-cost single-chip computers, it became possible to have large numbers of these microprocessors connected into physically small networks within a single chassis. These kinds of computers, which often use off-the-shelf microprocessors, are available from a number of different manufacturers.

One important reason why software has not evolved faster to make use of concurrent machines is that the power of processors has continually increased. One of the strongest motivations to use concurrent machines is to increase the speed of computation. However, two hardware factors have combined to provide faster computation, without requiring any change in the architecture of software systems. First, processor clock rates have become faster with each new generation of processors (the generations have appeared roughly every 18 months). Second, several different kinds of concurrency have been built into the processor architectures. Among these are the pipelining of instructions and data from the memory to the processor (instructions are fetched and decoded for future execution while the current instruction is being executed), the use of separate lines for instructions and data, prefetching of instructions and data, and parallelism in the execution of arithmetic operations. All of these are collectively called **hidden concurrency**. The result of the increases in execution speed is that there have been great productivity gains without requiring software developers to produce concurrent software systems.

However, the situation has changed. The end of the sequence of significant

increases in the speed of individual processors is now near. Significant increases in computing power now result from increases in the number of processors, for example large server systems like those run by Google and Amazon and scientific research applications. Many other large computing tasks are now run on machines with large numbers of relatively small processors.

Another recent advance in computing hardware was the development of multiple processors on a single chip, such as with the Intel Core Duo and Core Quad chips, which is putting more pressure on software developers to make more use of the available multiple processor machines. If they do not, the concurrent hardware will be wasted and significant productivity gains will not be realized.

13.1.2 Categories of Concurrency

There are two distinct categories of concurrent unit control. The most natural category of concurrency is that in which, assuming that more than one processor is available, several program units from the same program literally execute simultaneously. This is **physical concurrency**. A slight relaxation of this concept of concurrency allows the programmer and the application software to assume that there are multiple processors providing actual concurrency, when in fact the actual execution of programs is taking place in interleaved fashion on a single processor. This is **logical concurrency**. From the programmer's and language designer's points of view, logical concurrency is the same as physical concurrency. It is the language implementor's task, using the capabilities of the underlying operating system, to map the logical concurrency to the host hardware. Both logical and physical concurrencies allow the concept of concurrency to be used as a program design methodology. For the remainder of this chapter, the discussion will apply to both physical and logical concurrencies.

One useful technique for visualizing the flow of execution through a program is to imagine a thread laid on the statements of the source text of the program. Every statement reached on a particular execution is covered by the thread representing that execution. Visually following the thread through the source

program traces the execution flow through the executable version of the program. Of course, in all but the simplest of programs, the thread follows a highly complex path that would be impossible to follow visually. Formally, a **thread of control** in a program is the sequence of program points reached as control flows through the program.

Programs that have coroutines (see Chapter 9) but no concurrent subprograms, though they are sometimes called **quasi-concurrent**, have a single thread of control. Programs executed with physical concurrency can have multiple threads of control. Each processor can execute one of the threads. Although logically concurrent program execution may actually have only a single thread of control, such programs can be designed and analyzed only by imagining them as having multiple threads of control. A program designed to have more than one thread of control is said to be **multithreaded**. When a multithreaded program executes on a single-processor machine, its threads are mapped onto a single thread. It becomes, in this scenario, a virtually multithreaded program.

Statement-level concurrency is a relatively simple concept. In a common use of statement-level concurrency, loops that include statements that operate on array elements are unwound so that the processing can be distributed over multiple processors. For example, a loop that executes 500 repetitions and includes a statement that operates on one of 500 array elements may be unwound so that each of 10 different processors can simultaneously process 50 of the array elements.

13.1.3 Motivations for the Use of Concurrency

There are at least four different reasons to design concurrent software systems. The first reason is the speed of execution of programs on machines with multiple processors. These machines provide an effective way of increasing the execution speed of programs, provided that the programs are designed to make use of the concurrent hardware. There are now a large number of installed multiple-processor computers, including many of the

personal computers sold in the last few years. It is wasteful not to use this hardware capability.

The second reason is that even when a machine has just one processor, a program written to use concurrent execution can be faster than the same program written for sequential (nonconcurrent) execution. The requirement for this to happen is that the program is not compute bound (the sequential version does not fully utilize the processor).

The third reason is that concurrency provides a different method of conceptualizing program solutions to problems. Many problem domains lend themselves naturally to concurrency in much the same way that recursion is a natural way to design solutions to some problems. Also, many programs are written to simulate physical entities and activities. In many cases, the system being simulated includes more than one entity, and the entities do whatever they do simultaneously—for example, aircraft flying in a controlled airspace, relay stations in a communications network, and the various machines in a factory. Software that uses concurrency must be used to simulate such systems accurately.

The fourth reason for using concurrency is to program applications that are distributed over several machines, either locally or through the Internet. Many machines, for example cars, have more than one built-in computer, each of which is dedicated to some specific task. In many cases, these collections of computers must synchronize their program executions. Internet games are another example of software that is distributed over multiple processors.

Concurrency is now used in numerous everyday computing tasks. Web servers process document requests concurrently. Web browsers now use secondary core processors to run graphic processing and to interpret programming code embedded in documents. In every operating system there are many concurrent processes being executed at all times, managing resources, getting input from keyboards, displaying output from programs, and reading and writing external memory devices. In short, concurrency has become a ubiquitous part of computing.

13.2 Introduction to Subprogram-Level Concurrency

Before language support for concurrency can be considered, one must understand the underlying concepts of concurrency and the requirements for it to be useful. These topics are covered in this section.

13.2.1 Fundamental Concepts

A **task** is a unit of a program, similar to a subprogram, that can be in concurrent execution with other units of the same program. Each task in a program can support one thread of control. Tasks are sometimes called **processes**. In some languages, for example Java and C#, certain methods serve as tasks. Such methods are executed in objects called **threads**.

Three characteristics of tasks distinguish them from subprograms. First, a task may be implicitly started, whereas a subprogram must be explicitly called. Second, when a program unit invokes a task, in some cases it need not wait for the task to complete its execution before continuing its own. Third, when the execution of a task is completed, control may or may not return to the unit that started that execution.

Tasks fall into two general categories: heavyweight and lightweight. Simply stated, a **heavyweight task** executes in its own address space. **Lightweight tasks** all run in the same address space. It is easier to implement lightweight tasks than heavyweight tasks. Furthermore, lightweight tasks can be more efficient than heavyweight tasks, because less effort is required to manage their execution.

A task can communicate with other tasks through shared nonlocal variables, through message passing, or through parameters. If a task does not communicate with or affect the execution of any other task in the program in any way, it is said to be **disjoint**. Because tasks often work together to create

simulations or solve problems and therefore are not disjoint, they must use some form of communication to either synchronize their executions or share data or both.

Synchronization is a mechanism that controls the order in which tasks execute. Two kinds of synchronization are required when tasks share data: cooperation and competition. **Cooperation synchronization** is required between task `A` and task `B` when task `A` must wait for task `B` to complete some specific activity before task `A` can begin or continue its execution. **Competition synchronization** is required between two tasks when both require the use of some resource that cannot be simultaneously used. Specifically, if task `A` needs to access shared data location `x` while task `B` is accessing `x`, task `A` must wait for task `B` to complete its processing of `x`. So, for cooperation synchronization, tasks may need to wait for the completion of specific processing on which their correct operation depends, whereas for competition synchronization, tasks may need to wait for the completion of any other processing by any task currently occurring on specific shared data.

A simple form of cooperation synchronization can be illustrated by a common problem called the **producer-consumer problem**. This problem originated in the development of operating systems, in which one program unit produces some data value or resource and another uses it. Produced data are usually placed in a storage buffer by the producing unit and removed from that buffer by the consuming unit. The sequence of stores to and removals from the buffer must be synchronized. The consumer unit must not be allowed to take data from the buffer if the buffer is empty. Likewise, the producer unit cannot be allowed to place new data in the buffer if the buffer is full. This is a problem of cooperation synchronization because the users of the shared data structure must cooperate if the buffer is to be used correctly.

Competition synchronization prevents two tasks from accessing a shared data structure at exactly the same time—a situation that could destroy the integrity of that shared data. To provide competition synchronization, mutually exclusive access to the shared data must be guaranteed.

To clarify the competition problem, consider the following scenario: Suppose task `A` has the statement `TOTAL += 1`, where `TOTAL` is a shared integer variable. Furthermore, suppose task `B` has the statement `TOTAL *= 2`. Task `A`

and task B could try to change TOTAL at the same time.

At the machine language level, each task may accomplish its operation on TOTAL with the following three-step process:

1. Fetch the value of TOTAL.
2. Perform the arithmetic operation.
3. Put the new value back in TOTAL.

Without competition synchronization, given the previously described operations performed by tasks A and B on TOTAL, four different values could result, depending on the order of the steps of the operations. Assume TOTAL has the value 3 before either A or B attempts to modify it. If task A completes its operation before task B begins, the value will be 8, which is assumed here to be correct. But if both A and B fetch the value of TOTAL before either task puts its new value back, the result will be incorrect. If A puts its value back first, the value of TOTAL will be 6. This case is shown in Figure 13.1. If B puts its value back first, the value of TOTAL will be 4. Finally, if B completes its operation before task A begins, the value will be 7. A situation that leads to these problems is sometimes called a **race condition**, because two or more tasks are racing to use the shared resource and the behavior of the program depends on which task arrives first (and wins the race). The importance of competition synchronization should now be clear.

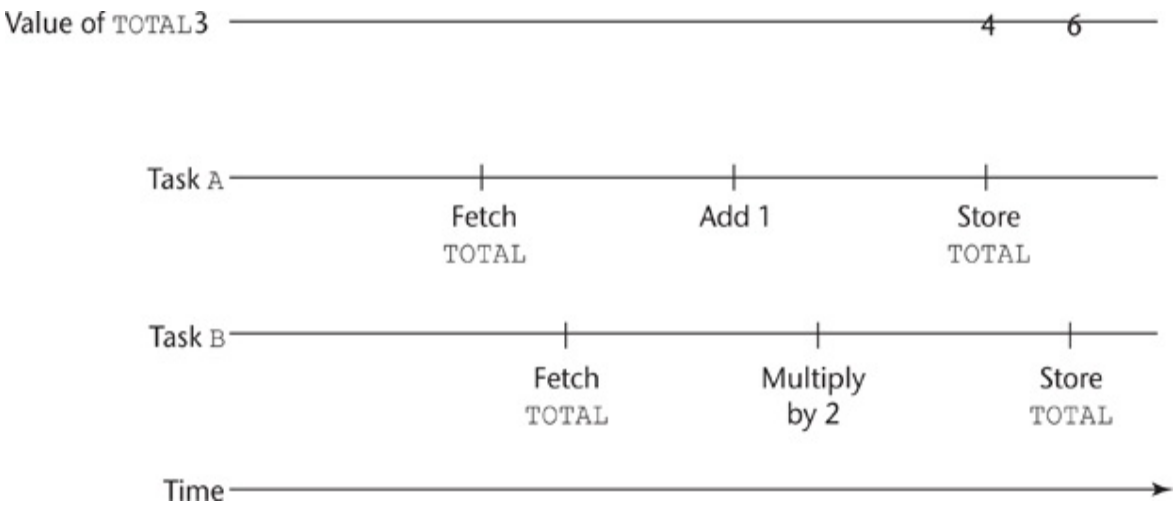


Figure 13.1 The need for - competition synchronization

Figure 13.1 Full Alternative Text

One general method for providing mutually exclusive access (to support competition synchronization) to a shared resource is to consider the resource to be something that a task can possess and allow only a single task to possess it at a time. To gain possession of a shared resource, a task must request it. Possession will be granted only when no other task has possession. While a task possesses a resource, all other tasks are prevented from having access to that resource. When a task is finished with a shared resource that it possesses, it must relinquish that resource so it can be made available to other tasks.

Three methods of providing for mutually exclusive access to a shared resource are semaphores, which are discussed in Section 13.3; monitors, which are discussed in Section 13.4; and message passing, which is discussed in Section 13.5.

Mechanisms for synchronization must be able to delay task execution. Synchronization imposes an order of execution on tasks that is enforced with these delays. To understand what happens to tasks through their lifetimes, we must consider how task execution is controlled. Regardless of whether a machine has a single processor or more than one, there is always the possibility of there being more tasks than there are processors. A run-time system program called a **scheduler** manages the sharing of processors among the tasks. If there were never any interruptions and tasks all had the same priority, the scheduler could simply give each task a time slice, such as 0.1 second, and when a task's turn came, the scheduler could let it execute on a processor for that amount of time. Of course, there are several events that complicate this, for example, task delays for synchronization and for input or output operations. Because input and output operations are very slow relative to the processor's speed, a task is not allowed to keep a processor while it waits for completion of such an operation.

Tasks can be in several different states:

1. New: A task is in the new state when it has been created but has not yet begun its execution.

2. Ready: A ready task is ready to run but is not currently running. Either it has not been given processor time by the scheduler, or it had run previously but was blocked in one of the ways described in Paragraph 4 of this subsection. Tasks that are ready to run are stored in a queue that is often called the **task-ready queue**.

3. Running: A running task is one that is currently executing; that is, it has a processor and its code is being executed.

4. Blocked: A task that is blocked has been running, but that execution was interrupted by one of several different events, the most common of which is an input or output operation. In addition to input and output, some languages provide operations for the user program to specify that a task be blocked.

5. Dead: A dead task is no longer active in any sense. A task dies when its execution is completed or it is explicitly killed by the program.

A flow diagram of the states of a task is shown in Figure 13.2.

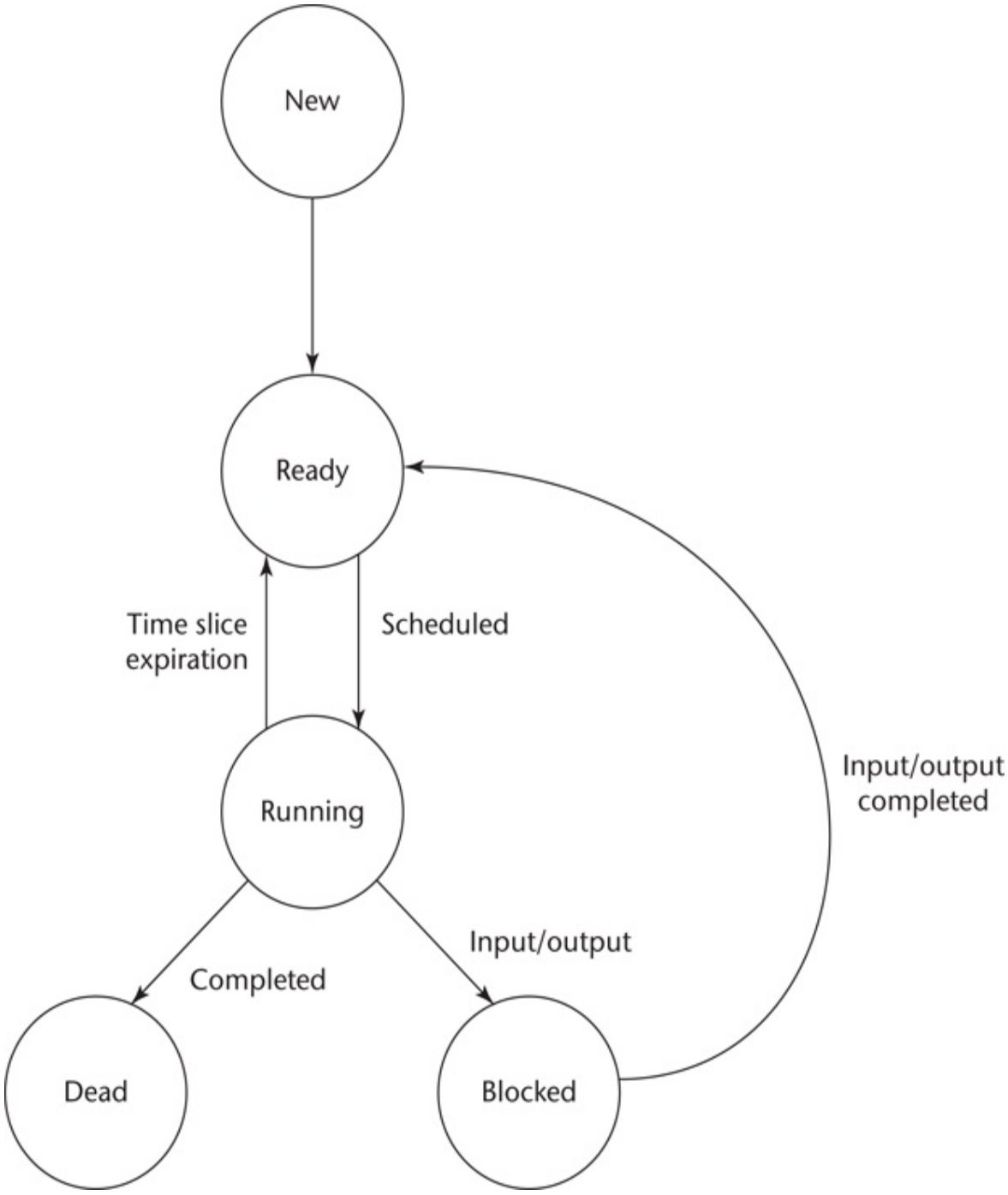


Figure 13.2 Flow diagram of task states

Figure 13.2 Full Alternative Text

One important issue in task execution is the following: How is a ready task chosen to move to the running state when the task currently running has become blocked or whose time slice has expired? Several different algorithms have been used for making this choice, some based on specifiable priority levels. The algorithm that does the choosing is implemented in the scheduler.

Associated with the concurrent execution of tasks and the use of shared resources is the concept of liveness. In the environment of sequential programs, a program has the **liveness** characteristic if it continues to execute, eventually leading to completion. In more general terms, liveness means that if some event—say, program completion—is supposed to occur, it will occur, eventually. That is, progress is continually made. In a concurrent environment and with shared resources, the liveness of a task can cease to exist, meaning that the program cannot continue and thus will never terminate.

For example, suppose task A and task B both need the shared resources X and Y to complete their work. Furthermore, suppose that task A gains possession of X and task B gains possession of Y. After some execution, task A needs resource Y to continue, so it requests Y but must wait until B releases it. Likewise, task B requests X but must wait until A releases it. Neither relinquishes the resource it possesses, and as a result, both lose their liveness, guaranteeing that execution of the program will never complete normally. This particular kind of loss of liveness is called **deadlock**. Deadlock is a serious threat to the reliability of a program, and therefore its avoidance demands serious consideration in both language and program design.

We are now ready to discuss some of the linguistic mechanisms for providing concurrent unit control.

13.2.2 Language Design for Concurrency

In some cases, concurrency is implemented through libraries. Among these is OpenMP, an applications programming interface to support shared memory multiprocessor programming in C, C++, and Fortran on a variety of platforms. Our interest in this book, of course, is language support for concurrency. A number of languages have been designed to support concurrency, beginning with PL/I in the mid-1960s and including the contemporary languages Ada 95, Java, C#, F#, Python, and Ruby.[1]

1. In the cases of Python and Ruby, programs are interpreted, so there only can be logical concurrency. Even if the machine has multiple processors, these programs cannot make use of more than one.

13.2.3 Design Issues

The most important design issues for language support for concurrency have already been discussed at length: competition and cooperation synchronization. In addition to these, there are several design issues of secondary importance. Prominent among them is how an application can influence task scheduling. Also, there are the issues of how and when tasks start and end their executions, and how and when they are created.

Keep in mind that our discussion of concurrency is intentionally incomplete, and only the most important of the language design issues related to support for concurrency are discussed.

The following sections discuss three alternative approaches to the design issues for concurrency: semaphores, monitors, and message passing.

13.3 Semaphores

A semaphore is a simple mechanism that can be used to provide synchronization of tasks. Although semaphores are an early approach to providing synchronization, they are still used, both in contemporary languages and in library-based concurrency support systems. In the following paragraphs, we describe semaphores and discuss how they can be used for this purpose.

13.3.1 Introduction

In an effort to provide competition synchronization through mutually exclusive access to shared data structures, Edsger Dijkstra devised semaphores in 1965 (Dijkstra, 1968b). Semaphores can also be used to provide cooperation synchronization.

To provide limited access to a data structure, guards can be placed around the code that accesses the structure. A **guard** is a linguistic device that allows the guarded code to be executed only when a specified condition is true. So, a guard can be used to allow only one task to access a particular shared data structure at a time. A semaphore is an implementation of a guard. Specifically, a **semaphore** is a data structure that consists of an integer and a queue that stores task descriptors. A **task descriptor** is a data structure that stores all of the relevant information about the execution state of a task.

An integral part of a guard mechanism is a procedure for ensuring that all attempted executions of the guarded code eventually take place. The typical approach is to have requests for access that occur when access cannot be granted or stored in the task descriptor queue, from which they are later allowed to leave and execute the guarded code. This is the reason a semaphore must have both a counter and a task descriptor queue.

The only two operations provided for semaphores were originally named P and V by Dijkstra, after the two Dutch words *passeren* (to pass) and *vrygeren*

(to release) (Andrews and Schneider, 1983). We will refer to these as *wait* and *release*, respectively, in the remainder of this section.

13.3.2 Cooperation Synchronization

Through much of this chapter, we use the example of a shared buffer used by producers and consumers to illustrate the different approaches to providing cooperation and competition synchronization. For cooperation synchronization, such a buffer must have some way of recording both the number of empty positions and the number of filled positions in the buffer (to prevent buffer underflow and overflow). The counter component of a semaphore can be used for this purpose. One semaphore variable—for example, `emptyspots`—can use its counter to maintain the number of empty locations in a shared buffer used by producers and consumers, and another—say, `fullspots`—can use its counter to maintain the number of filled locations in the buffer. The queues of these semaphores can store the descriptors of tasks that have been forced to wait for access to the buffer. The queue of `emptyspots` can store producer tasks that are waiting for available positions in the buffer; the queue of `fullspots` can store consumer tasks waiting for values to be placed in the buffer.

Our example buffer is designed as an abstract data type in which all data enters the buffer through the subprogram `DEPOSIT`, and all data leaves the buffer through the subprogram `FETCH`. The `DEPOSIT` subprogram needs only to check with the `emptyspots` semaphore to see whether there are any empty positions. If there is at least one, it can proceed with the `DEPOSIT`, which must have the side effect of decrementing the counter of `emptyspots`. If the buffer is full, the caller to `DEPOSIT` must be made to wait in the `emptyspots` queue for an empty spot to become available. When the `DEPOSIT` is complete, the `DEPOSIT` subprogram increments the counter of the `fullspots` semaphore to indicate that there is one more filled location in the buffer.

The `FETCH` subprogram has the opposite sequence of `DEPOSIT`. It checks the `fullspots` semaphore to see whether the buffer contains at least one item. If

it does, an item is removed and the `emptyspots` semaphore has its counter incremented by 1. If the buffer is empty, the calling task is put in the `fullspots` queue to wait until an item appears. When `FETCH` is finished, it must increment the counter of `emptyspots`.

The operations on semaphore types often are not direct—they are done through `wait` and `release` subprograms. Therefore, the `DEPOSIT` operation just described is actually accomplished in part by calls to `wait` and `release`. Note that `wait` and `release` must be able to access the task-ready queue.

The `wait` semaphore subprogram is used to test the counter of a given semaphore variable. If the value is greater than zero, the caller can carry out its operation. In this case, the counter value of the semaphore variable is decremented to indicate that there is now one fewer of whatever it counts. If the value of the counter is zero, the caller must be placed on the waiting queue of the semaphore variable, and the processor must be given to some other ready task.

The `release` semaphore subprogram is used by a task to allow some other task to have one of whatever the counter of the specified semaphore variable counts. If the queue of the specified semaphore variable is empty, which means no task is waiting, `release` increments its counter (to indicate there is one more of whatever is being controlled that is now available). If one or more tasks are waiting, `release` moves one of them from the semaphore queue to the ready queue.

The following are concise pseudocode descriptions of `wait` and `release`:

```
wait(aSemaphore)

if aSemaphore's counter > 0 then

    decrement aSemaphore's counter

else

    put the caller in aSemaphore's queue

    attempt to transfer control to some ready task
```

(if the task-ready queue is empty, deadlock occurs)

end if

`release(aSemaphore)`

if `aSemaphore`'s queue is empty `(no task is waiting)` **then**

increment `aSemaphore`'s counter

else

put the calling task in the task-ready queue

transfer control to a task from `aSemaphore`'s queue

end

We can now present an example program that implements cooperation synchronization for a shared buffer. In this case, the shared buffer stores integer values and is a logically circular structure. It is designed for use by possibly multiple producer and consumer tasks.

The following pseudocode shows the definition of the producer and consumer tasks. Two semaphores are used to ensure against buffer underflow or overflow, thus providing cooperation synchronization. Assume that the buffer has length `BUFLEN`, and the routines that actually manipulate it already exist as `FETCH` and `DEPOSIT`. Accesses to the counter of a semaphore are specified by dot notation. For example, if `fullspots` is a semaphore, its counter is referenced by `fullspots.count`.

```
semaphore fullspots, emptyspots;
fullspots.count = 0;
emptyspots.count = BUFLEN;
task producer;
  loop
  -- produce VALUE --
  wait(emptyspots);     { wait for a space }
  DEPOSIT(VALUE);
  release(fullspots);   { increase filled spaces }
  end loop;
end producer;
```

```
task consumer;
   loop
   wait(fullspots);     { make sure it is not empty }
   FETCH(VALUE);
   release(emptyspots); { increase empty spaces }
   -- consume VALUE --
   end loop
end consumer;
```

The semaphore `fullspots` causes the `consumer` task to be queued to wait for a buffer entry if it is currently empty. The semaphore `emptyspots` causes the `producer` task to be queued to wait for an empty space in the buffer if it is currently full.

13.3.3 Competition Synchronization

Our buffer example does not provide competition synchronization. Access to the structure can be controlled with an additional semaphore. This semaphore need not count anything but can simply indicate with its counter whether the buffer is currently being used. The `wait` statement allows the access only if the semaphore's counter has the value `1`, which indicates that the shared buffer is not currently being accessed. If the semaphore's counter has a value of `0`, there is a current access taking place, and the task is placed in the queue of the semaphore. Notice that the semaphore's counter must be initialized to `1`. The queues of semaphores must always be initialized to empty before use of the queue can begin.

A semaphore that requires only a binary-valued counter, like the one used to provide competition synchronization in the following example, is called a **binary semaphore**.

The example pseudocode that follows illustrates the use of semaphores to provide both competition and cooperation synchronization for a concurrently accessed shared buffer. The `access` semaphore is used to ensure mutually exclusive access to the buffer. Remember that there may be more than one producer and more than one consumer.

```
semaphore access, fullspots, emptyspots;
access.count = 1;
fullspots.count = 0;
emptyspots.count = BUFLEN;
task producer;
  loop
  -- produce VALUE --
  wait(emptyspots);      { wait for a space }
  wait(access);          { wait for access }
  DEPOSIT(VALUE);
  release(access);       { relinquish access }
  release(fullspots);    { increase filled spaces }
  end loop;
end producer;
task consumer;
  loop
  wait(fullspots);      { make sure it is not empty }
  wait(access);         { wait for access }
  FETCH(VALUE);
  release(access);      { relinquish access }
  release(emptyspots);  { increase empty spaces }
  -- consume VALUE --
  end loop
end consumer;
```

A brief look at this example may lead one to believe that there is a problem. Specifically, suppose that while a task is waiting at the `wait(access)` call in `consumer`, another task takes the last value from the shared buffer. Fortunately, this cannot happen, because the `wait(fullspots)` reserves a value in the buffer for the task that calls it by decrementing the `fullspots` counter.

There is one crucial aspect of semaphores that thus far has not been discussed. Recall the earlier description of the problem of competition synchronization: Operations on shared data must not overlap. If a second operation begins while an earlier operation is still in progress, the shared data can become corrupted. A semaphore is itself a shared data object, so the operations on semaphores are also susceptible to the same problem. It is therefore essential that semaphore operations be uninterruptible. Many computers have uninterruptible instructions that were designed specifically for semaphore operations. If such instructions are not available, then using semaphores to provide competition synchronization is a serious problem with no simple solution.

13.3.4 Evaluation

Using semaphores to provide cooperation synchronization creates an unsafe programming environment. There is no way to check statically for the correctness of their use, which depends on the semantics of the program in which they appear. In the buffer example, leaving out the `wait(emptyspots)` statement of the `producer` task would result in buffer overflow. Leaving out the `wait(fullspots)` statement of the `consumer` task would result in buffer underflow. Leaving out either of the releases would result in deadlock. These are cooperation synchronization failures.

The reliability problems that semaphores cause in providing cooperation synchronization also arise when using them for competition synchronization. Leaving out the `wait(access)` statement in either task can cause insecure access to the buffer. Leaving out the `release(access)` statement in either task results in deadlock. These are competition synchronization failures. Noting the danger in using semaphores, Per Brinch Hansen (1973) wrote, “The semaphore is an elegant synchronization tool for an ideal programmer who never makes mistakes.” Unfortunately, ideal programmers are rare.

13.4 Monitors

One solution to some of the problems of semaphores in a concurrent environment is to encapsulate shared data structures with their operations and hide their representations—that is, to make shared data structures abstract data types with some special restrictions. This solution can provide competition synchronization without semaphores by transferring responsibility for synchronization to the run-time system.

13.4.1 Introduction

When the concepts of data abstraction were being formulated, the people involved in that effort applied the same concepts to shared data in concurrent programming environments to produce monitors. According to Per Brinch Hansen (Brinch Hansen, 1977, p. xvi), Edsger Dijkstra suggested in 1971 that all synchronization operations on shared data be gathered into a single program unit. Brinch Hansen (1973) formalized this concept in the environment of operating systems. The following year, Hoare (1974) named these structures *monitors*.

The first programming language to incorporate monitors was Concurrent Pascal (Brinch Hansen, 1975). Modula (Wirth, 1977), CSP/k (Holt et al., 1978), and Mesa (Mitchell et al., 1979) also provide monitors. Among contemporary languages, monitors are supported by Ada, Java, and C#, all of which are discussed later in this chapter.

13.4.2 Competition Synchronization

One of the most important features of monitors is that shared data is resident in the monitor rather than in any of the client units. The programmer does not

synchronize mutually exclusive access to shared data through the use of semaphores or other mechanisms. Because the access mechanisms are part of the monitor, implementation of a monitor can be made to guarantee synchronized access by allowing only one access at a time. Calls to monitor procedures are implicitly blocked and stored in a queue if the monitor is busy at the time of the call.

13.4.3 Cooperation Synchronization

Although mutually exclusive access to shared data is intrinsic with a monitor, cooperation between processes is still the task of the programmer. In particular, the programmer must guarantee that a shared buffer does not experience underflow or overflow. Different languages provide different ways of programming cooperation synchronization, all of which are related to semaphores.

A figure depicting a program containing four tasks and a monitor that provides synchronized access to a concurrently shared buffer is shown in Figure 13.3. In this figure, the interface to the monitor is shown as the two boxes labeled `insert` and `remove` (for the insertion and removal of data). The monitor appears exactly like an abstract data type—a data structure with limited access—which is what a monitor is.

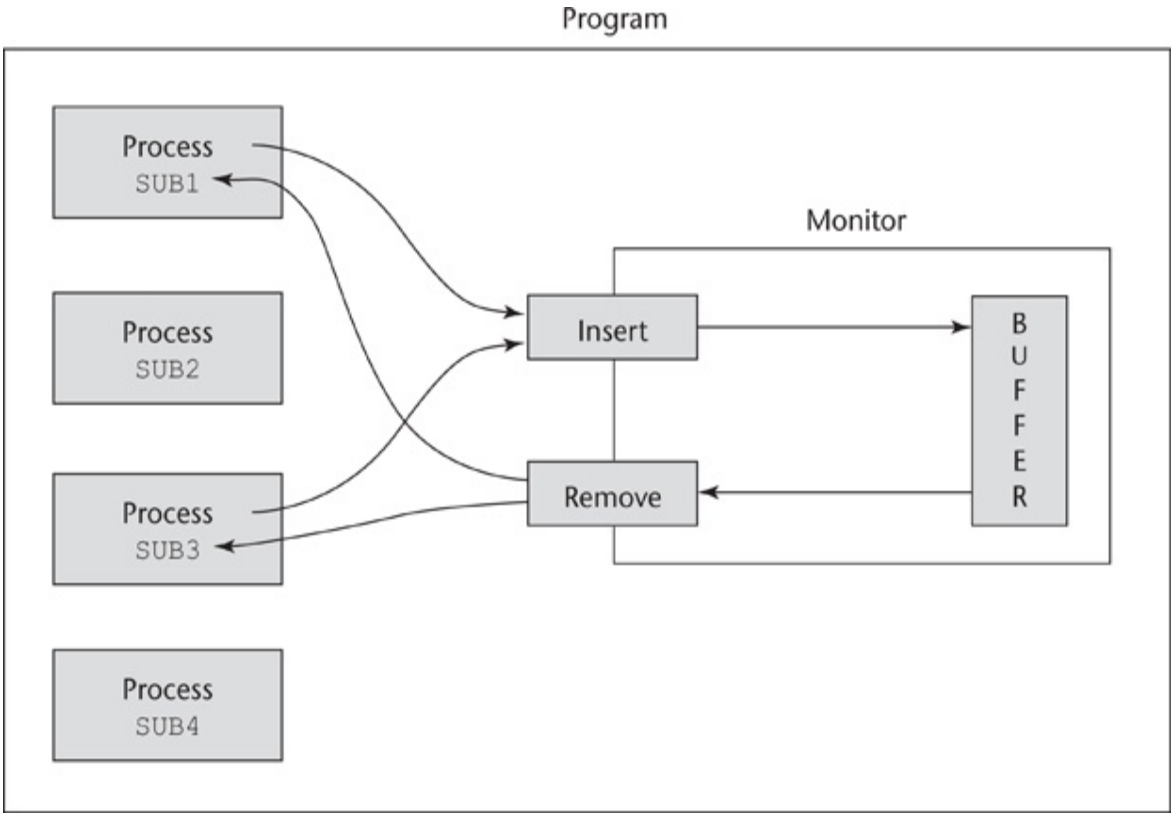


Figure 13.3 A program using a monitor to control access to a shared buffer

Figure 13.3 Full Alternative Text

13.4.4 Evaluation

Monitors are a better way to provide competition synchronization than are semaphores, primarily because of the problems of semaphores, as discussed in Section 13.3. The cooperation synchronization is still a problem with monitors, as will be clear when Ada and Java implementations of monitors are discussed in the following sections.

Semaphores and monitors are equally powerful at expressing concurrency control—semaphores can be used to implement monitors and monitors can be used to implement semaphores.

Ada provides two ways to implement monitors. Ada 83 includes a general tasking model that can be used to support monitors. Ada 95 added a cleaner and more efficient way of constructing monitors, called *protected objects*. Both of these approaches use message passing as a basic model for supporting concurrency. The message-passing model allows concurrent units to be distributed, which monitors do not allow. Message passing is described in Section 13.5; Ada support for message passing is discussed in Section 13.6.

In Java, a monitor can be implemented in a class designed as an abstract data type, with the shared data being the type. Accesses to objects of the class are controlled by adding the **`synchronized`** modifier to the access methods. An example of a monitor for the shared buffer written in Java is given in Section 13.7.4.

C# has a predefined class, `Monitor`, which is designed for implementing monitors.

13.5 Message Passing

This section introduces the fundamental concept of message passing in concurrency. Note that this concept of message passing is unrelated to the message passing used in object-oriented programming to enact methods.

13.5.1 Introduction

The first efforts to design languages that provide the capability for message passing among concurrent tasks were those of Brinch Hansen (1978) and Hoare (1978). These pioneer developers of message passing also developed a technique for handling the problem of what to do when multiple simultaneous requests were made by other tasks to communicate with a given task. It was decided that some form of nondeterminism was required to provide fairness in choosing which among those requests would be taken first. This fairness can be defined in various ways, but in general, it means that all requesters are provided an equal chance of communicating with a given task (assuming that every requester has the same priority). Nondeterministic constructs for statement-level control, called *guarded commands*, were introduced by Dijkstra (1975). Guarded commands are discussed in Chapter 8. Guarded commands are the basis of the construct designed for controlling message passing.

13.5.2 The Concept of Synchronous Message Passing

Message passing can be either synchronous or asynchronous. Here, we describe synchronous message passing. The basic concept of synchronous message passing is that tasks are often busy, and when busy, they cannot be interrupted by other units. Suppose task A and task B are both in execution, and A wishes to send a message to B. Clearly, if B is busy, it is not desirable to

allow another task to interrupt it. That would disrupt B's current processing. Furthermore, messages usually cause associated processing in the receiver, which might not be sensible if other processing is incomplete. The alternative is to provide a linguistic mechanism that allows a task to specify to other tasks when it is ready to receive messages. This approach is somewhat like an executive who instructs his or her secretary to hold all incoming calls until another activity, perhaps an important conversation, is completed. Later, when the current conversation is complete, the executive tells the secretary that he or she is now willing to talk to one of the callers who has been placed on hold.

A task can be designed so that it can suspend its execution at some point, either because it is idle or because it needs information from another unit before it can continue. This is like a person who is waiting for an important call. In some cases, there is nothing else to do but sit and wait. However, if task A is waiting for a message at the time task B sends that message, the message can be transmitted. This actual transmission of the message is called a **rendezvous**. Note that a rendezvous can occur only if both the sender and receiver want it to happen. During a rendezvous, the information of the message can be transmitted in either or both directions.

Both cooperation and competition synchronization of tasks can be conveniently handled with the message-passing model, as described in the following section.

13.6 Ada Support for Concurrency

This section describes the support for concurrency provided by Ada. Ada 83 supports only synchronous message passing.

13.6.1 Fundamentals

The Ada design for tasks is partially based on the work of Brinch Hansen and Hoare in that message passing is the design basis and nondeterminism is used to choose among the tasks that have sent messages.

The full Ada tasking model is complex, and the following discussion of it is limited. The focus here will be on the Ada version of the synchronous message-passing mechanism.

Ada tasks can be more active than monitors. Monitors are passive entities that provide management services for the shared data they store. They provide their services, though only when those services are requested. When used to manage shared data, Ada tasks can be thought of as managers that can reside with the resource they manage. They have several mechanisms, some deterministic and some nondeterministic, that allow them to choose among competing requests for access to their resources.

There are two syntactic parts to an Ada task—a specification part and a body part—both with the same name. The interface of a task is its entry points or locations where it can accept messages from other tasks. Because these entry points are part of its interface, it is natural that they be listed in the specification part of a task. Because a rendezvous can involve an exchange of information, messages can have parameters; therefore, task entry points must also allow parameters, which must also be described in the specification part. In appearance, a task specification is similar to the package specification for an abstract data type.

As an example of an Ada task specification, consider the following code,

which includes a single entry point named `Entry_1`, which has an in-mode parameter:

```
task Task_Example is
  entry Entry_1(Item : in Integer);
end Task_Example;
```

A task body must include some syntactic form of the entry points that correspond to the **`entry`** clauses in that task's specification part. In Ada, these task body entry points are specified by clauses that are introduced by the **`accept`** reserved word. An **`accept` clause** is defined as the range of statements beginning with the **`accept`** reserved word and ending with the matching **`end`** reserved word. **`accept`** clauses are themselves relatively simple, but other constructs in which they can be embedded can make their semantics complex. A simple **`accept`** clause has the following form:

```
accept entry_name (formal parameters) do
  ...
end entry_name;
```

The **`accept`** entry name matches the name in an **`entry`** clause in the associated task specification part. The optional parameters provide the means of communicating data between the caller and the called task. The statements between the **`do`** and the **`end`** define the operations that take place during the rendezvous. These statements are together called the **`accept` clause body**. During the actual rendezvous, the sender task is suspended.

Whenever an **`accept`** clause receives a message that it is not willing to accept, for whatever reason, the sender task must be suspended until the **`accept`** clause in the receiver task is ready to accept the message. Of course, the **`accept`** clause must also remember the sender tasks that have sent messages that were not accepted. For this purpose, each **`accept`** clause in a task has a queue associated with it that stores a list of other tasks that have unsuccessfully attempted to communicate with it.

The following is the skeletal body of the task whose specification was given previously:

```
task body Task_Example is
  begin
```

```
  loop
    accept Entry_1(Item : in Integer) do
      ...
    end Entry_1;
  end loop;
  end Task_Example;
```

The **accept** clause of this task body is the implementation of the **entry** named `Entry_1` in the task specification. If the execution of `Task_Example` begins and reaches the `Entry_1` **accept** clause before any other task sends a message to `Entry_1`, `Task_Example` is suspended. If another task sends a message to `Entry_1` while `Task_Example` is suspended at its **accept**, a rendezvous occurs and the **accept** clause body is executed. Then, because of the loop, execution proceeds back to the **accept**. If no other task has sent a message to `Entry_1`, execution is again suspended to wait for the next message.

A rendezvous can occur in two basic ways in this simple example. First, the receiver task, `Task_Example`, can be waiting for another task to send a message to the `Entry_1` entry. When the message is sent, the rendezvous occurs. This is the situation described earlier. Second, the receiver task can be busy with one rendezvous, or with some other processing not associated with a rendezvous, when another task attempts to send a message to the same entry. In that case, the sender is suspended until the receiver is free to accept that message in a rendezvous. If several messages arrive while the receiver is busy, the senders are queued to wait their turn for a rendezvous.

The two rendezvous just described are illustrated with the timeline diagrams in Figure 13.4.

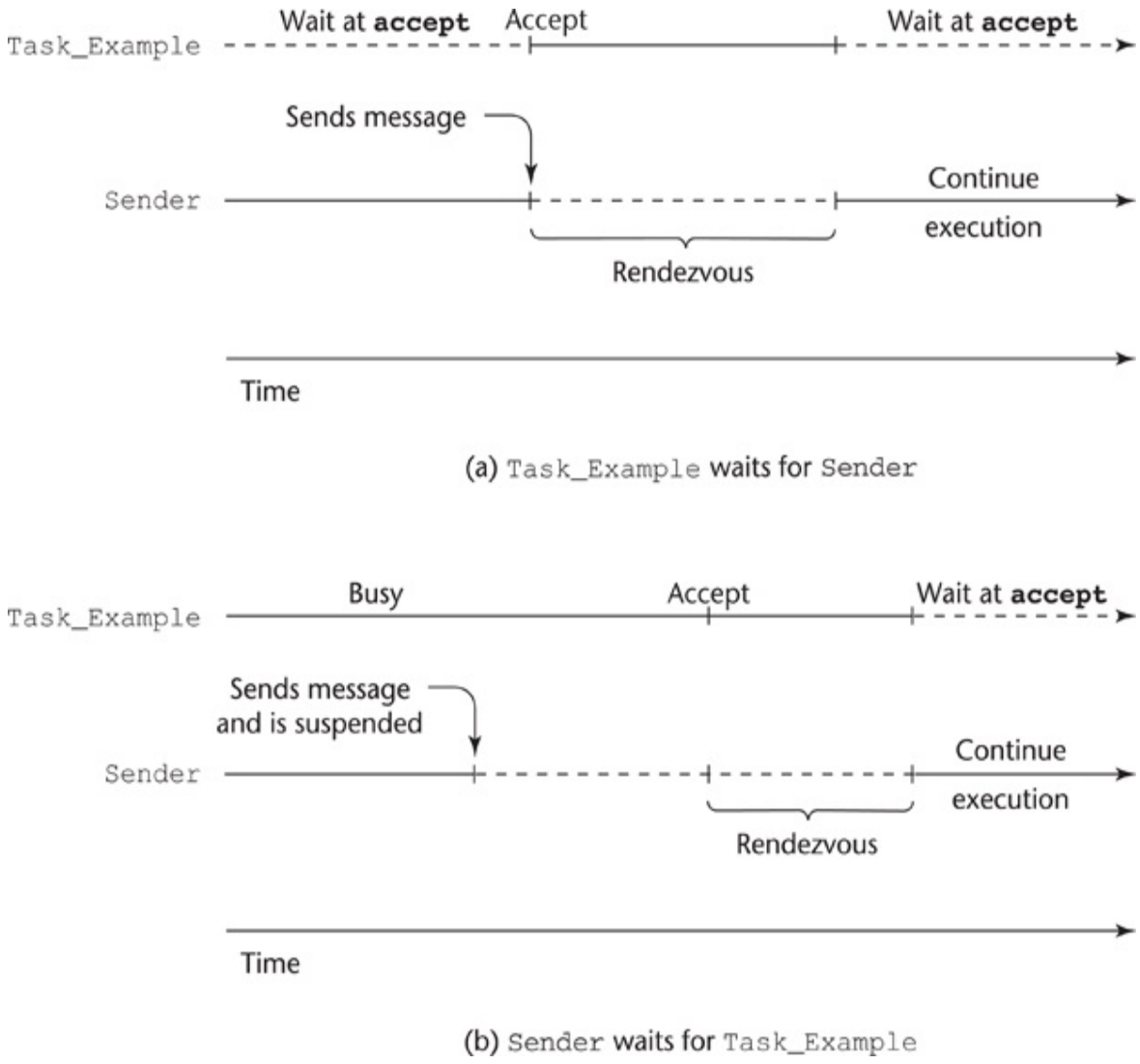


Figure 13.4 Two ways a rendezvous with `Task_Example` can occur

Figure 13.4 Full Alternative Text

Tasks need not have entry points. Such tasks are called **actor tasks** because they do not wait for a rendezvous in order to do their work. Actor tasks can rendezvous with other tasks by sending them messages. In contrast to actor

tasks, a task can have **`accept`** clauses but not have any code outside those **`accept`** clauses, so it can only react to other tasks. Such a task is called a **server task**.

An Ada task that sends a message to another task must know the entry name in that task. However, the opposite is not true: A task entry need not know the name of the task from which it will accept messages. This asymmetry is in contrast to the design of the language known as CSP, or Communicating Sequential Processes (Hoare, 1978). In CSP, which also uses the message--passing model of concurrency, tasks accept messages only from explicitly named tasks. The disadvantage of this is that libraries of tasks cannot be built for general use.

The usual graphical method of describing a rendezvous in which task A sends a message to task B is shown in Figure 13.5.

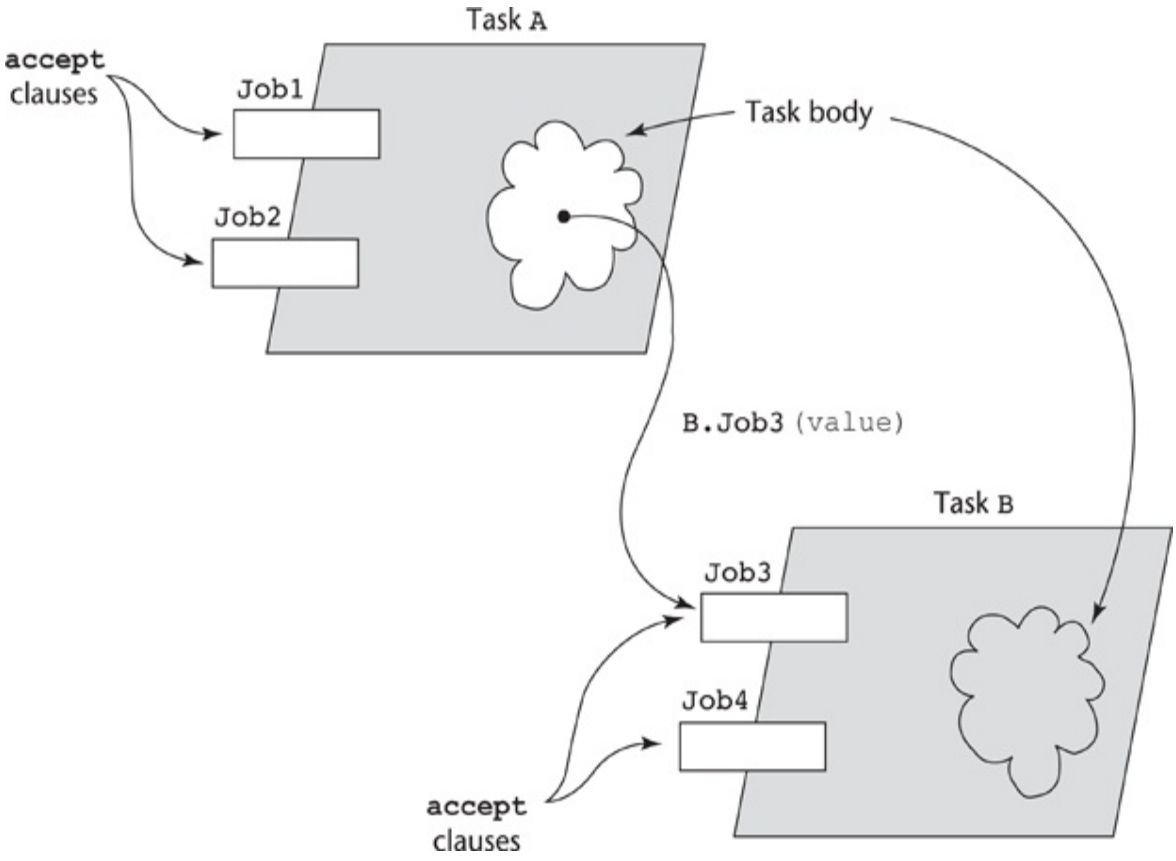


Figure 13.5 Graphical -

representation of a rendezvous caused by a message sent from task `A` to task `B`

Figure 13.5 Full Alternative Text

Tasks are declared in the declaration part of a package, subprogram, or block. Statically created tasks2 begin executing at the same time as the statements in the code to which that declarative part is attached. For example, a task declared in a main program begins execution at the same time as the first statement in the code body of the main program. Task termination, which is a complex issue, is discussed later in this section.

2. Tasks can also be dynamically created, but such tasks are not covered here.

Tasks may have any number of entries. The order in which the associated **`accept`** clauses appear in the task dictates the order in which messages can be accepted. If a task has more than one entry point and requires them to be able to receive messages in any order, the task uses a **`select`** statement to enclose the entries. For example, suppose a task models the activities of a bank teller, who must serve customers at a walk-up station inside the bank and also serve customers at a drive-up window. The following skeletal teller task illustrates a **`select`** construct:

```
task body Teller is
begin
  loop
    select
      accept Drive_Up(formal parameters) do
        ...
      end Drive_Up;
      ...
    or
      accept Walk_Up(formal parameters) do
        ...
      end Walk_Up;
      ...
```

```
    end select;
  end loop;
end Teller;
```

In this task, there are two **accept** clauses, Walk_Up and Drive_Up, each of which has an associated queue. The action of the **select**, when it is executed, is to examine the queues associated with the two **accept** clauses. If one of the queues is empty, but the other contains at least one waiting message (customer), the **accept** clause associated with the waiting message or messages has a rendezvous with the task that sent the first message that was received. If both **accept** clauses have empty queues, the **select** waits until one of the entries is called. If both **accept** clauses have nonempty queues, one of the **accept** clauses is nondeterministically chosen to have a rendezvous with one of its callers. The loop forces the **select** statement to be executed repeatedly, forever.

The **end** of the **accept** clause marks the end of the code that assigns or references the formal parameters of the **accept** clause. The code, if there is any, between an **accept** clause and the next **or** (or the **end select**, if the **accept** clause is the last one in the **select**) is called the **extended accept clause**. The extended **accept** clause is executed only after the associated (immediately preceding) **accept** clause is executed. This execution of the extended **accept** clause is not part of the rendezvous and can take place concurrently with the execution of the calling task. The sender is suspended during the rendezvous, but it is put back in the ready queue when the end of the **accept** clause is reached. If an **accept** clause has no formal parameters, the **do-end** is not required, and the **accept** clause can consist entirely of an extended **accept** clause. Such an **accept** clause would be used exclusively for synchronization. Extended **accept** clauses are illustrated in the Buf_Task task in Section 13.6.3.

13.6.2 Cooperation Synchronization

Each **accept** clause can have a guard attached, in the form of a **when** clause,

that can delay rendezvous. For example,

```
when not Full(Buffer) =>
  accept Deposit(New_Value) do
    ...
  end
```

An **accept** clause with a **when** clause is either open or closed. If the Boolean expression of the **when** clause is currently true, that **accept** clause is called **open**; if the Boolean expression is false, the **accept** clause is called **closed**. An **accept** clause that does not have a guard is always open. An open **accept** clause is available for rendezvous; a closed **accept** clause cannot rendezvous.

Suppose there are several guarded **accept** clauses in a **select** clause. Such a **select** clause is usually placed in an infinite loop. The loop causes the **select** clause to be executed repeatedly, with each **when** clause evaluated on each repetition. Each repetition causes a list of open **accept** clauses to be constructed. If exactly one of the open clauses has a nonempty queue, a message from that queue is taken and a rendezvous takes place. If more than one of the open **accept** clauses has nonempty queues, one queue is chosen nondeterministically, a message is taken from that queue, and a rendezvous takes place. If the queues of all open clauses are empty, the task waits for a message to arrive at one of those **accept** clauses, at which time a rendezvous will occur. If a **select** is executed and every **accept** clause is closed, a run-time exception or error results. This possibility can be avoided either by making sure one of the **when** clauses is always true or by adding an **else** clause in the **select**. An **else** clause can include any sequence of statements, except an **accept** clause.

A **select** clause may have a special statement, **terminate**, that is selected only when it is open and no other **accept** clause is open. A **terminate** clause, when selected, means that the task is finished with its job but is not yet terminated. Task termination is discussed later in this section.

13.6.3 Competition Synchronization

The features described so far provide for cooperation synchronization and communication among tasks. Next, we discuss how mutually exclusive access to shared data structures can be enforced in Ada.

If access to a data structure is to be controlled by a task, then mutually exclusive access can be achieved by declaring the data structure within a task. The semantics of task execution usually guarantees mutually exclusive access to the structure, because only one **accept** clause in the task can be active at a given time. The only exceptions to this occur when tasks are nested in procedures or other tasks. For example, if a task that defines a shared data structure has a nested task, that nested task can also access the shared structure, which could destroy the integrity of the data. Thus, tasks that are meant to control access to a shared data structure should not define tasks.

The following is an example of an Ada task that implements a monitor for a buffer. The buffer behaves very much like the buffer in Section 13.3, in which synchronization is controlled with semaphores.

```
task Buf_Task is
   entry Deposit(Item : in Integer);
   entry Fetch(Item : out Integer);
end Buf_Task;
task body Buf_Task is
  Bufsize : constant Integer := 100;
  Buf   :  array (1..Bufsize) of Integer;
  Filled : Integer range 0..Bufsize := 0;
  Next_In,
  Next_Out : Integer range 1..Bufsize := 1;
begin
  loop
    select
      when Filled < Bufsize =>
         accept Deposit(Item : in Integer) do
           Buf(Next_In) := Item;
         end Deposit;
        Next_In := (Next_In mod Bufsize) + 1;
        Filled := Filled + 1;
    or
      when Filled > 0 =>
        accept Fetch(Item : out Integer) do
          Item := Buf(Next_Out);
        end Fetch;
        Next_Out := (Next_Out mod Bufsize) + 1;
```

```
        Filled := Filled - 1;
    end select;
   end loop;
end Buf_Task;
```

In this example, both **accept** clauses are extended. These extended clauses can be executed concurrently with the tasks that called the associated **accept** clauses.

The tasks for a producer and a consumer that could use `Buf_Task` have the following form:

```
task Producer;
task Consumer;
task body Producer is
  New_Value : Integer;
begin
  loop
    -- produce New_Value --
    Buf_Task.Deposit(New_Value);
  end loop;
end Producer;
task body Consumer is
  Stored_Value : Integer;
begin
  loop
    Buf_Task.Fetch(Stored_Value);
    -- consume Stored_Value --
  end loop;
end Consumer;
```

13.6.4 Protected Objects

As we have seen, access to shared data can be controlled by enclosing the data in a task and allowing access only through task entries, which implicitly provide competition synchronization. One problem with this method is that it is difficult to implement the rendezvous mechanism efficiently. Ada 95 protected objects provide an alternative method of providing competition synchronization that need not involve the rendezvous mechanism.

A protected object is not a task; it is more like a monitor, as described in

Section 13.4. Protected objects can be accessed either by protected subprograms or by entries that are syntactically similar to the **accept** clauses in tasks.3 The protected subprograms can be either protected procedures, which provide mutually exclusive read-write access to the data of the protected object, or protected functions, which provide concurrent read-only access to that data. Entries differ from protected subprograms in that they can have guards.

3. Entries in protected object bodies use the reserved word `entry`, rather than the `accept` used in task bodies.

Within the body of a protected procedure, the current instance of the enclosing protected unit is defined to be a variable; within the body of a protected function, the current instance of the enclosing protected unit is defined to be a constant, which allows concurrent read-only access.

Entry calls to a protected object provide synchronous communication with one or more tasks using the same protected object. These entry calls provide access similar to that provided to the data enclosed in a task.

The buffer problem that is solved with a task in the previous subsection can be more simply solved with a protected object. Note that this example does not include protected subprograms.

```
protected Buffer is
  entry Deposit(Item : in Integer);
  entry Fetch(Item : out Integer);
private
  Bufsize : constant Integer := 100;
  Buf   : array  (1..Bufsize) of Integer;
  Filled : Integer range 0..Bufsize := 0;
  Next_In,
  Next_Out : Integer range 1..Bufsize := 1;
  end Buffer;
protected body Buffer is
  entry Deposit(Item : in Integer)
     when Filled < Bufsize is
    begin
    Buf(Next_In) := Item;
    Next_In := (Next_In mod Bufsize) + 1;
    Filled := Filled + 1;
    end Deposit;
```

```
  entry Fetch(Item : out Integer) when Filled > 0 is
    begin Item := Buf(Next_Out);
    Next_Out := (Next_Out mod Bufsize) + 1;
    Filled := Filled - 1;
    end Fetch;
end Buffer;
```

13.6.5 Evaluation

Using the general message-passing model of concurrency to construct monitors is like using Ada packages to support abstract data types—both are tools that are more general than is necessary. Protected objects are a better way to provide synchronized access to shared data.

In the absence of distributed processors with independent memories, the choice between monitors and tasks with message passing as a means of implementing synchronized access to shared data in a concurrent environment is somewhat a matter of taste. However, in the case of Ada, protected objects are clearly better than tasks for supporting concurrent access to shared data. Not only is the code simpler; it is also much more efficient.

For distributed systems, message passing is a better model for concurrency, because it naturally supports the concept of separate processes executing in parallel on separate processors.

13.7 Java Threads

The concurrent units in Java are methods named `run`, whose code can be in concurrent execution with other such methods (of other objects) and with the `main` method. The process in which the `run` methods execute is called a **thread**. Java's threads are lightweight tasks, which means that they all run in the same address space. This is different from Ada tasks, which are heavyweight threads (they run in their own address spaces).4 One important result of this difference is that threads require far less overhead than Ada's tasks.

4. Actually, although Ada tasks behave as if they were heavyweight tasks, in some cases, they are now implemented as threads. This is sometimes done using libraries, such as the IBM Rational Apex Native POSIX Threading Library.

There are two ways to define a class with a `run` method. One of these is to define a subclass of the predefined class `Thread` and override its `run` method. However, if the new subclass has a necessary natural parent, then defining it as a subclass of `Thread` obviously will not work. In these situations, we define a subclass that inherits from its natural parent and implements the `Runnable` interface. `Runnable` provides the `run` method protocol, so any class that implements `Runnable` must define `run`. An object of the class that implements `Runnable` is passed to the `Thread` constructor. So, this approach still requires a `Thread` object, as will be seen in the example in Section 13.7.5.

In Ada, tasks can be either actors or servers and tasks communicate with each other through **`accept`** clauses. Java `run` methods are all actors and there is no mechanism for them to communicate with each other, except for the `join` method (see Section 13.7.1) and through shared data.

Java threads is a complex topic—this section only provides an introduction to its simplest but most useful parts.

13.7.1 The Thread Class

The `Thread` class is not the natural parent of any other classes. It provides some services for its subclasses, but it is not related in any natural way to their computational purposes. `Thread` is the only class available for creating concurrent Java programs. As previously stated, Section 13.7.5 will briefly discuss the use of the `Runnable` interface.

The `Thread` class includes five constructors and a collection of methods and constants. The `run` method, which describes the actions of the thread, is always overridden by subclasses of `Thread`. The `start` method of `Thread` starts its thread as a concurrent unit by calling its `run` method.5 The call to `start` is unusual in that control returns immediately to the caller, which then continues its execution, in parallel with the newly started `run` method.

5. Calling the `run` method directly does not always work, because initialization that is sometimes required is included in the `start` method.

Following is a skeletal subclass of `Thread` and a code fragment that creates an object of the subclass and starts the `run` method's execution in the new thread:

```
class MyThread extends Thread {
  public void run() { . . . }
}
. . .
Thread myTh = new MyThread();
myTh.start();
```

When a Java application program begins execution, a new thread is created (in which the **`main`** method will run) and **`main`** is called. Therefore, all Java application programs run in threads.

When a program has multiple threads, a scheduler must determine which thread or threads will run at any given time. In many cases, there is only a single processor available, so only one thread actually runs at a time. It is difficult to give a precise description of how the Java scheduler works, because the different implementations (Solaris, Windows, and so on) do not

necessarily schedule threads in exactly the same way. Typically, however, the scheduler gives equal-size time slices to each ready thread in round-robin fashion, assuming all of these threads have the same priority. Section 13.7.2 describes how different priorities can be given to different threads.

The `Thread` class provides several methods for controlling the execution of threads. The `yield` method, which takes no parameters, is a request from the running thread to surrender the processor voluntarily.6 The thread is put immediately in the task-ready queue, making it ready to run. The scheduler then chooses the highest-priority thread from the task-ready queue. If there are no other ready threads with priority higher than the one that just yielded the processor, it may also be the next thread to get the processor.

6. The `yield` method is actually defined to be a "suggestion" to the scheduler, which it may or may not follow (though it usually does).

The `sleep` method has a single parameter, which is the integer number of milliseconds that the caller of `sleep` wants the thread to be blocked. After the specified number of milliseconds has passed, the thread will be put in the task-ready queue. Because there is no way to know how long a thread will be in the task-ready queue before it runs, the parameter to `sleep` is the minimum amount of time the thread will *not* be in execution. The `sleep` method can throw an `InterruptedException`, which must be handled in the method that calls `sleep`. Exceptions are described in detail in Chapter 14.

The `join` method is used to force a method to delay its execution until the `run` method of another thread has completed its execution. `join` is used when the processing of a method cannot continue until the work of the other thread is complete. For example, we might have the following `run` method:

```
public void run() {
  . . .
  Thread myTh = new Thread();
  myTh.start();
  // do part of the computation of this thread
  myTh.join();  // Wait for myTh to complete
  // do the rest of the computation of this thread
}
```

The `join` method puts the thread that calls it in the blocked state, which can

be ended only by the completion of the thread on which `join` was called. If that thread happens to be blocked, there is the possibility of deadlock. To prevent this, `join` can be called with a parameter, which is the time limit in milliseconds of how long the calling thread will wait for the called thread to complete. For example, the following call to `join` will cause the calling thread to wait two seconds for `myTh` to complete. If it has not completed its execution after two seconds have passed, the calling thread is put back in the ready queue, which means that it will continue its execution as soon as it is scheduled.

```
myTh.join(2000);
```

Early versions of Java included three more `Thread` methods: `stop`, `suspend`, and `resume`. All three of these have been deprecated because of safety problems. The `stop` method is sometimes overridden with a simple method that destroys the thread by setting its reference variable to **`null`**.

The normal way a `run` method ends its execution is by reaching the end of its code. However, in many cases, threads run until told to terminate. Regarding this, there is the question of how a thread can determine whether it should continue or end. The `interrupt` method is one way to communicate to a thread that it should stop. This method does not stop the thread; rather, it sends the thread a message that actually just sets a bit in the thread object, which can be checked by the thread. The bit is checked with the predicate method, `isInterrupted`. This is not a complete solution, because the thread one is attempting to interrupt may be sleeping or waiting at the time the `interrupt` method is called, which means that it will not be checking to see if it has been interrupted. For these situations, the `interrupt` method also throws an exception, `InterruptedException`, which also causes the thread to awaken (from sleeping or waiting). So, a thread can periodically check to see whether it has been interrupted and if so, whether it can terminate. The thread cannot miss the interrupt, because if it was asleep or waiting when the interrupt occurred, it will be awakened by the interrupt. Actually, there are more details to the actions and uses of `interrupt`, but they are not covered here (Arnold et al., 2006).

13.7.2 Priorities

The priorities of threads need not all be the same. A thread's default priority initially is the same as the thread that created it. If `main` creates a thread, its default priority is the constant `NORM_PRIORITY`, which is usually 5. `Thread` defines two other priority constants, `MAX_PRIORITY` and `MIN_PRIORITY`, whose values are usually 10 and 1, respectively.7 The priority of a thread can be changed with the method `setPriority`. The new priority can be any of the predefined constants or any other number between `MIN_PRIORITY` and `MAX_PRIORITY`. The `getPriority` method returns the current priority of a thread. The priority constants are defined in `Thread`.

7. The number of priorities is implementation dependent, so there may be fewer or more than 10 levels in some implementations.

When there are threads with different priorities, the scheduler's behavior is controlled by those priorities. When the executing thread is blocked or killed or the time slice for it expires, the scheduler chooses the thread from the task-ready queue that has the highest priority. A thread with lower priority will run only if one of higher priority is not in the task-ready queue when the opportunity arises.

13.7.3 Semaphores

The `java.util.concurrent.Semaphore` package defines the `Semaphore` class. Objects of this class implement counting semaphores. A counting semaphore has a counter, but no queue for storing thread descriptors. The `Semaphore` class defines two methods, `acquire` and `release`, which correspond to the `wait` and `release` operations described in Section 13.3.

The basic constructor for `Semaphore` takes one integer parameter, which initializes the semaphore's counter. For example, the following could be used to initialize the `fullspots` and `emptyspots` semaphores for the buffer example of Section 13.3.2:

```
fullspots = new Semaphore(0);
emptyspots = new Semaphore(BUFLEN);
```

The deposit operation of the producer method would appear as follows:

```
emptyspots.acquire();
deposit(value);
fullspots.release();
```

Likewise, the fetch operation of the consumer method would appear as follows:

```
fullspots.acquire();
fetch(value);
emptyspots.release();
```

The `deposit` and `fetch` methods could use the approach used in Section 13.7.4 to provide the competition synchronization required for the accesses to the buffer.

13.7.4 Competition Synchronization

Java methods (but not constructors) can be specified to be **`synchronized`**. A synchronized method called through a specific object must complete its execution before any other synchronized method can run on that object. Competition synchronization on an object is implemented by specifying that the methods that access shared data are synchronized. The synchronized mechanism is implemented as follows: Every Java object has a lock. Synchronized methods must acquire the lock of the object before they are allowed to execute, which prevents other synchronized methods from executing on the object during that time. A synchronized method releases the lock on the object on which it runs when it completes its execution, even if that completion is due to an exception. Consider the following skeletal class definition:

```
class ManageBuf {
  private int [100] buf;
```

```
  . . .
  public synchronized void deposit(int item) { . . . }
  public synchronized int fetch() { . . . }
  . . .
}
```

The two methods defined in `ManageBuf` are both defined to be **`synchronized`**, which prevents them from interfering with each other while executing on the same object, when they are called by separate threads.

An object whose methods are all synchronized is effectively a monitor. Note that an object may have one or more synchronized methods, as well as one or more unsynchronized methods. An unsynchronized method can run on an object at anytime, even during the execution of a synchronized method.

In some cases, the number of statements that deal with the shared data structure is significantly less than the number of other statements in the method in which it resides. In these cases, it is better to synchronize the code segment that changes the shared data structure rather than the whole method. This can be done with a so-called *synchronized statement*, whose general form is as follows:

```
synchronized (expression){
 statements
}
```

The expression in this code must evaluate to an object and the statement can be a single statement or a compound statement. The object is locked during execution of the statement or compound statement, so the statement or compound statement is executed exactly as if it were the body of a synchronized method.

An object that has synchronized methods defined for it must have a queue associated with it that stores the synchronized methods that have attempted to execute on it while it was being operated upon by another synchronized method. Actually, every object has a queue called the **intrinsic condition queue**. These queues are implicitly supplied. When a synchronized method completes its execution on an object, a method that is waiting in the object's intrinsic condition queue, if there is such a method, is put in the task-ready queue.

13.7.5 Cooperation Synchronization

Cooperation synchronization in Java is implemented with the `wait`, `notify`, and `notifyAll` methods, all of which are defined in `Object`, the root class of all Java classes. All classes except `Object` inherit these methods. Every object has a wait list of all of the threads that have called `wait` on the object. The `notify` method is called to tell one waiting thread that an event that it may have been waiting for has occurred. The specific thread that is awakened by `notify` cannot be determined, because the Java Virtual Machine (JVM) chooses one from the wait list of the thread object at random. Because of this, along with the fact that the waiting threads may be waiting for different conditions, the `notifyAll` method is often used, rather than `notify`. The `notifyAll` method awakens all of the threads on the object's wait list by putting them in the task-ready queue.

The methods `wait`, `notify`, and `notifyAll` can be called only from within a synchronized method, because they use the lock placed on an object by such a method. The call to `wait` is always put in a **`while`** loop that is controlled by the condition for which the method is waiting. The **`while`** loop is necessary because the `notify` or `notifyAll` that awakened the thread may have been called because of a change in a condition other than the one for which the thread was waiting. If it was a call to `notifyAll`, there is even a smaller chance that the condition being waited for is now true. Because of the use of `notifyAll`, some other thread may have changed the condition to false since it was last tested.

The `wait` method can throw `InterruptedException`, which is a descendant of `Exception`. Java's exception handling is discussed in Chapter 14. Therefore, any code that calls `wait` must also catch `InterruptedException`. Assuming the condition being waited for is called `theCondition`, the conventional way to use `wait` is as follows:

```
try  {
  while (!theCondition)
    wait();
```

```
  -- Do whatever is needed after theCondition comes true
}
catch(InterruptedException myProblem) { . . . }
```

The following program implements a circular queue for storing **int** values. It illustrates both cooperation and competition synchronization.

```
// Queue
// This class implements a circular queue for storing int
// values. It includes a constructor for allocating and
// initializing the queue to a specified size. It has
// synchronized methods for inserting values into and
// removing values from the queue.
class Queue {
  private int [] que;
  private int nextIn,
              nextOut,
              filled,
              queSize;
  public Queue(int size) {
    que = new int [size];
    filled = 0;
    nextIn = 1;
    nextOut = 1;
    queSize = size;
  }  //** end of Queue constructor
  public synchronized void deposit (int item)
         throws InterruptedException {
    try {
      while (filled == queSize)
        wait();
      que [nextIn] = item;
      nextIn = (nextIn % queSize) + 1;
      filled++;
      notifyAll();
    }  //** end of try clause
    catch(InterruptedException e) {}
  }  //** end of deposit method
  public synchronized int fetch()
      throws InterruptedException {
    int item = 0;
    try {
      while (filled == 0)
        wait();
      item = que [nextOut];
      nextOut = (nextOut % queSize) + 1;
      filled--;
```

```
      notifyAll();
    }  //** end of try clause
    catch(InterruptedException e) {}
    return item;
  }  //** end of fetch method
}  //** end of Queue class
```

Notice that the exception handler (**`catch`**) does nothing here.

Classes to define producer and consumer objects that could use the `Queue` class can be defined as follows:

```
class Producer extends Thread {
  private Queue buffer;
  public Producer(Queue que) {
    buffer = que;
  }
  public void run() {
     int new_item;
     while  (true) {
      //-- Create a new_item
      buffer.deposit(new_item);
    }
  }
}

class Consumer extends Thread {
  private Queue buffer;
  public Consumer(Queue que) {
    buffer = que;
  }
  public void run() {
    int stored_item;
    while (true) {
      stored_item = buffer.fetch();
      //-- Consume the stored_item
    }
  }
}
```

The following code creates a `Queue` object, and a `Producer` and a `Consumer` object, both attached to the `Queue` object, and starts their execution:

```
Queue buff1 = new Queue(100);
Producer producer1 = new Producer(buff1);
Consumer consumer1 = new Consumer(buff1);
```

```
producer1.start();
consumer1.start();
```

We could define one or both of the `Producer` and the `Consumer` as implementations of the `Runnable` interface rather than as subclasses of `Thread`. The only difference is in the first line, which would now appear as follows:

```
class Producer implements Runnable { . . . }
```

To create and run an object of such a class, it is still necessary to create a `Thread` object that is connected to the object. This is illustrated in the following code:

```
Producer producer1 = new Producer(buff1);
Thread producerThread = new Thread(producer1);
producerThread.start();
```

Note that the buffer object is passed to the `Producer` constructor and the `Producer` object is passed to the `Thread` constructor.

13.7.6 Nonblocking Synchronization

Java includes some classes for controlling accesses to certain variables that do not include blocking or waiting. The `java.util.concurrent.atomic` package defines classes that allow certain nonblocking synchronized access to **`int`**, **`long`**, and **`boolean`** primitive type variables, as well as references and arrays. For example, the `AtomicInteger` class defines getter and setter methods, as well as methods for add, increment, and decrement operations. These operations are all atomic; that is, they cannot be interrupted, so locks are not required to guarantee the integrity of the values of the affected variables in a multithreaded program. This is fine-grained synchronization—just a single variable. Most machines now have atomic instructions for these operations on **`int`** and **`long`** types, so they are often easy to implement (implicit locks are not required).

The advantage of nonblocking synchronization is efficiency. A nonblocking access that does not occur during contention will be no slower, and usually faster than one that uses **`synchronized`**. A nonblocking access that occurs during contention definitely will be faster than one that uses **`synchronized`**, because the latter will require suspension and rescheduling of threads.

13.7.7 Explicit Locks

Java 5.0 introduced explicit locks as an alternative to **`synchronized`** method and blocks, which provide implicit locks. The `Lock` interface declares the `lock`, `unlock`, and `tryLock` methods. The predefined `ReentrantLock` class implements the `Lock` interface. To lock a block of code, the following idiom can be used:

```
Lock lock = new ReentrantLock();
. . .
Lock.lock();
try {
   // The code that accesses the shared data
} finally  {
  Lock.unlock();
}
```

This skeletal code creates a `Lock` object and calls the `lock` method on the `Lock` object. Then, it uses a **`try`** block to enclose the critical code. The call to `unlock` is in a **`finally`** clause to guarantee the lock is released, regardless of what happens in the **`try`** block.

There are at least two situations in which explicit locks are used rather than implicit locks: First, if the application needs to try to acquire a lock but cannot wait forever for it, the `Lock` interface includes a method, `tryLock`, that takes a time limit parameter. If the lock is not acquired within the time limit, execution continues at the statement following the call to `tryLock`. Second, explicit locks are used when it is not convenient to have the lock-unlock pairs block structured. Implicit locks are always unlocked at the end of the compound statement in which they are locked. Explicit locks can be unlocked anywhere in the code, regardless of the structure of the program.

One danger of using explicit locks (and is not the case with using implicit locks) is that of omitting the unlock. Implicit locks are implicitly unlocked at the end of the locked block. However, explicit locks stay locked until explicitly unlocked, which can potentially be never.

As stated previously, each object has an intrinsic condition queue, which stores threads waiting for a condition on the object. The `wait`, `notify`, and `notifyAll` methods are the API for an intrinsic condition queue. Because each object can have just one condition queue, a queue may have threads in it waiting for different conditions. For example, the queue for our buffer example `Queue` can have threads waiting for either of two conditions (`filled == queSize` or `filled == 0`). That is the reason why the buffer uses `notifyAll`. (If it used `notify`, only one thread would be awakened, and it might be one that was waiting for a different condition than the one that actually became true.) However, `notifyAll` is expensive to use, because it awakens all threads waiting on an object and all must check their condition to determine which one runs. Furthermore, to check their condition, they must first acquire the lock on the object.

An alternative to using the intrinsic condition queue is the `Condition` interface, which uses a condition queue associated with a `Lock` object. It also declares alternatives to `wait`, `notify`, and `notifyAll` named `await`, `signal`, and `signalAll`. There can be any number of `Condition` objects with one `Lock` object. With `Condition`, `signal`, rather than `signalAll`, can be used, which is both easier to understand and more efficient, in part because it results in fewer context switches.

13.7.8 Evaluation

Java’s support for concurrency is relatively simple but effective. All Java `run` methods are actor tasks and there is no mechanism for communication, except through shared data, as there is among Ada tasks. Because they are heavyweight threads, Ada’s tasks easily can be distributed to different processors; in particular, different processors with different memories, which could be on different computers in different places. These kinds of systems are not possible with Java’s threads.

13.8 C# Threads

Although C#'s threads are loosely based on those of Java, there are significant differences. Following is a brief overview of C#'s threads.

13.8.1 Basic Thread Operations

Rather than just methods named `run`, as in Java, any C# method can run in its own thread. When C# threads are created, they are associated with an instance of a predefined delegate, `ThreadStart`. When execution of a thread is started, its delegate has the address of the method it is supposed to run. So, execution of a thread is controlled through its associated delegate.

A C# thread is created by creating a `Thread` object. The `Thread` constructor must be sent an instantiation of `ThreadStart`, to which must be sent the name of the method that is to run in the thread. For example, we might have

```
public void MyRun1() { . . . }
. . .
Thread myThread = new Thread(new ThreadStart(MyRun1));
```

In this example, we create a thread named `myThread`, whose delegate points to the method `MyRun1`. So, when the thread begins execution it calls the method whose address is in its delegate. In this example, `myThread` is the delegate and `MyRun1` is the method.

Unlike Java, in which all threads are actors, C# has two categories of threads: actors and servers. Actor threads are not called specifically; rather, they are started. Also, the methods that they execute do not take parameters or return values. As with Java, creating a thread does not start its concurrent execution. For actor threads, execution must be requested through a method of the `Thread` class, in this case named `Start`, as in

```
myThread.Start();
```

As in Java, a thread can be made to wait for another thread to finish its execution before continuing, using the similarly named method `Join`. For example, suppose thread `A` has the following call:

```
B.Join();
```

Thread `A` will be blocked until thread `B` exits.

The `Join` method can take an **`int`** parameter, which specifies a time limit in milliseconds that the caller will wait for the thread to finish.

A thread can be suspended for a specified amount of time with `Sleep`, which is a public static method of `Thread`. The parameter to `Sleep` is an integer number of milliseconds. Unlike its Java relative, C#'s `Sleep` does not raise any exceptions, so it need not be called in a **`try`** block.

A thread can be terminated with the `Abort` method, although it does not literally kill the thread. Instead, it throws `ThreadAbortException`, which the thread can catch. When the thread catches this exception, it usually deallocates any resources it allocated, and then ends (by getting to the end of its code).

A server thread runs only when called through its delegate. These threads are called servers because they provide some service when it is requested. Server threads are more interesting than actor threads because they usually interact with other threads and often must have their execution synchronized with other threads.

Recall from Chapter 9, that any C# method can be called indirectly through a delegate. Such calls can be made by treating the delegate object as if it were the name of the method. This was actually an abbreviation for a call to a delegate method named `Invoke`. So, if a delegate object's name is `chgfun1` and the method it references takes one **`int`** parameter, we could call that method with either of the following statements:

```
chgfun1(7);
chgfun1.Invoke(7);
```

These calls are synchronous; that is, when the method is called, the caller is

blocked until the method completes its execution. C# also supports asynchronous calls to methods that execute in threads. When a thread is called asynchronously, the called thread and the caller thread execute concurrently, because the caller is not blocked during the execution of the called thread.

A thread is called asynchronously through the delegate instance method `BeginInvoke`, to which are sent the parameters for the method of the delegate, along with two additional parameters, one of type `AsyncCallback` and the other of type **`object`**. `BeginInvoke` returns an object that implements the `IAsyncResult` interface. The delegate class also defines the `EndInvoke` instance method, which takes one parameter of type `IAsyncResult` and returns the same type that is returned by the method encapsulated in the delegate object. To call a thread asynchronously, we call it with `BeginInvoke`. For now, we will use **`null`** for the last two parameters. Suppose we have the following method declaration and thread definition:

```
public float MyMethod1(int  x);
. . .
Thread myThread = new Thread(new ThreadStart(MyMethod1));
```

The following statement calls `MyMethod` asynchronously:

```
IAsyncResult result = myThread.BeginInvoke(10, null, null);
```

The return value of the called thread is fetched with `EndInvoke` method, which takes as its parameter the object (of type `IAsyncResult`) returned by `BeginInvoke`. `EndInvoke` returns the return value of the called thread. For example, to get the **`float`** result of the call to `MyMethod`, we would use the following statement:

```
float  returnValue = EndInvoke(result);
```

If the caller must continue some work while the called thread executes, it must have a way to determine when the called thread is finished. For this, the `IAsyncResult` interface defines the `IsCompleted` property. While the called thread is executing, the caller can include code it can execute in a **`while`** loop that depends on `IsCompleted`. For example, we could have the following:

```
IAsyncResult result = myThread.BeginInvoke(10, null, null);
while(!result.IsCompleted) {
  // Do some computation
}
```

This is an effective way to accomplish something in the calling thread while waiting for the called thread to complete its work. However, if the amount of computation in the **while** loop is relatively small, this is an inefficient way to use that time (because of the time required to test `IsCompleted`). An alternative is to give the called thread a delegate with the address of a callback method and have it call that method when it is finished. The delegate is sent as the second last parameter to `BeginInvoke`. For example, consider the following call to `BeginInvoke`:

```
IAsyncResult result = myThread.BeginInvoke(10,
                    new AsyncCallback(MyMethodComplete), null);
```

The callback method is defined in the caller. Such methods often simply set a Boolean variable, for example named `isDone`, to **true**. No matter how long the called thread takes, the callback method is called only once.

13.8.2 Synchronizing Threads

There are three different ways that C# threads can be synchronized: the `Interlocked` class, the `Monitor` class from the `System.Threading` namespace, and the `lock` statement. Each of these mechanisms is designed for a specific need. The `Interlocked` class is used when the only operations that need to be synchronized are the incrementing and decrementing of an integer. These operations are done atomically with the two methods of `Interlocked`, `Increment` and `Decrement`, both of which take a reference to an integer as the parameter. For example, to increment a shared integer named `counter` in a thread, we could use

```
Interlocked.Increment(ref  counter);
```

The `lock` statement is used to mark a critical section of code in a thread. The syntax of this is as follows:

```
lock(token) {
   // The critical section
}
```

If the code to be synchronized is in a private instance method, the token is the current object, so **`this`** is used as the token for `lock`. If the code to be synchronized is in a public instance method, a new instance of **`object`** is created (in the class of the method with the code to be synchronized) and a reference to it is used as the token for `lock`.

The `Monitor` class defines five methods, `Enter`, `Wait`, `Pulse`, `PulseAll`, and `Exit`, which can be used to provide more control of the synchronization of threads. The `Enter` method, which takes an object reference as its parameter, marks the beginning of synchronization of the thread on that object. The `Wait` method suspends execution of the thread and instructs the Common Language Runtime (CLR) of .NET that this thread wants to resume its execution the next time there is an opportunity. The `Pulse` method, which also takes an object reference as its parameter, notifies one waiting thread that it now has a chance to run again. `PulseAll` is similar to Java's `notifyAll`. Threads that have been waiting are run in the order in which they called the `Wait` method. The `Exit` method ends the critical section of the thread.

The `lock` statement is compiled into a monitor, so `lock` is shorthand for a monitor. A monitor is used when the additional control (for example, with `Wait` and `PulseAll`) is needed.

.NET 4.0 added a collection of generic concurrent data structures, including structures for queues, stacks, and bags.[8] These new classes are thread safe, meaning that they can be used in a multithreaded program without requiring the programmer to worry about competition synchronization. The `System.Collections.Concurrent` namespace defines these classes, whose names are `ConcurrentQueue<T>`, `ConcurrentStack<T>`, and `ConcurrentBag<T>`. So, our producer-consumer queue program could be written in C# using a `ConcurrentQueue<T>` for the data structure and there would be no need to program the competition synchronization for it. Because these concurrent collections are defined in .NET, they are also available in all of the other .NET languages.

8. Bags are unordered collections of objects.

13.8.3 Evaluation

C#'s threads are a slight improvement over those of its predecessor, Java. For one thing, any method can be run in its own thread. Recall that in Java, only methods named `run` can run in their own threads. Java supports actor threads only, but C# supports both actor and server threads. Thread termination is also cleaner with C# (calling a method (`Abort`) is more elegant than setting the thread's pointer to **`null`**). Synchronization of thread execution is more sophisticated in C#, because C# has several different mechanisms, each for a specific application. Java's `Lock` variables are similar to the locks of C#, except that in Java, a lock must be explicitly unlocked with a call to `unlock`. This provides one more way to create erroneous code. C# threads, like those of Java, are lightweight, so although they are more efficient, they cannot be as versatile as Ada's tasks. The availability of the concurrent collection classes is another advantage C# has over the other nonfunctional languages discussed in this chapter.

13.9 Concurrency in Functional Languages

This section provides a brief overview of support for concurrency in several functional programming languages.

13.9.1 Multi-LISP

Multi-LISP (Halstead, 1985) is an extension to Scheme that allows the programmer to specify program parts that can be executed concurrently. These forms of concurrency are implicit; the programmer is simply telling the compiler (or interpreter) some parts of the program that can be run concurrently.

One of the ways a programmer can tell the system about possible concurrency is the `pcall` construct. If a function call is embedded in a `pcall` construct, the parameters to the function can be evaluated concurrently. For example, consider the following `pcall` construct:

```
(pcall f a b c d)
```

The function is `f`, with parameters `a`, `b`, `c`, and `d`. The effect of `pcall` is that the parameters of the function can be evaluated concurrently (any or all of the parameters could be complicated expressions). Unfortunately, whether this process can be safely used, that is, without affecting the semantics of the function evaluation, is the responsibility of the programmer. This is actually a simple matter if the language does not allow side effects or if the programmer designed the function not to have side effects or at least to have limited ones. However, Multi-LISP does allow some side effects. If the function was not written to avoid side effects, it may be difficult for the programmer to determine whether `pcall` can be safely used.

The `future` construct of Multi-LISP is a more interesting and potentially

more productive source of concurrency. As with `pcall`, a function call is wrapped in a `future` construct. Such a function is evaluated in a separate thread, with the parent thread continuing its execution. The parent thread continues until it needs to use the return value of the function. If the function has not completed its execution when its result is needed, the parent thread waits until it has before it continues.

If a function has two or more parameters, they can also be wrapped in `future` constructs, in which case their evaluations can be done concurrently in separate threads.

These are the only additions to Scheme in Multi-LISP.

13.9.2 Concurrent ML

Concurrent ML (CML) is an extension to ML that includes a form of threads and a form of synchronous message passing to support concurrency. The language is completely described in Reppy (1999).

A thread is created in CML with the `spawn` primitive, which takes the function as its parameter. In many cases, the function is specified as an anonymous function. As soon as the thread is created, the function begins its execution in the new thread. The return value of the function is discarded. The effects of the function are either output produced or through communications with other threads. Either the parent thread (the one that spawned the new thread) or the child thread (the new one) could terminate first and it would not affect the execution of the other.

Channels provide the means of communicating between threads. A channel is created with the `channel` constructor. For example, the following statement creates a channel of arbitrary type named `mychannel`:

```
let val mychannel = channel()
```

The two primary operations (functions) on channels are for sending (`send`) and receiving (`recv`) messages. The type of the message is inferred from the send operation. For example, the following function call sends the integer

value `7`, and therefore the type of the channel is then inferred to be integer:

```
send(mychannel, 7)
```

The `recv` function names the channel as its parameter. Its return value is the value it received.

Because CML communications are synchronous, a message is both sent and received only if both the sender and the receiver are ready. If a thread sends a message on a channel and no other thread is ready to receive on that channel, the sender is blocked and waits for another thread to execute a `recv` on the channel. Likewise, if a `recv` is executed on a channel by a thread but no other thread has sent a message on that channel, the thread that ran the `recv` is blocked and waits for a message on that channel.

Because channels are types, functions can take them as parameters.

As was the case with Ada's synchronous message passing, an issue with CML synchronous message passing is deciding which message to choose when more than one channel has received one. And the same solution is used: the guarded command **`do-od`** construct that chooses randomly among messages to different channels.

The synchronization mechanism of CML is the *event*. An explanation of this complicated mechanism is beyond the scope of this chapter (and this book).

13.9.3 F#

Part of the F# support for concurrency is based on the same .NET classes that are used by C#, specifically `System.Threading.Thread`. For example, suppose we want to run the function `myConMethod` in its own thread. The following function, when called, will create the thread and start the execution of the function in the new thread:

```
let createThread() =
    let newThread = new Thread(myConMethod)
    newThread.Start()
```

Recall that in C#, it is necessary to create an instance of a predefined delegate, `ThreadStart`, send its constructor the name of the subprogram, and send the new delegate instance as a parameter to the `Thread` constructor. In F#, if a function expects a delegate as its parameter, a lambda expression or a function can be sent and the compiler will behave as if you sent the delegate. So, in the above code, the function `myConMethod` is sent as the parameter to the `Thread` constructor, but what is actually sent is a new instance of `ThreadStart` (to which was sent `myConMethod`).

The `Thread` class defines the `Sleep` method, which puts the thread from which it is called to sleep for the number of milliseconds that is sent to it as a parameter.

Shared immutable data does not require synchronization among the threads that access it. However, if the shared data is mutable, which is possible in F#, locking will be required to prevent corruption of the shared data by multiple threads attempting to change it. A mutable variable can be locked while a function operates on it to provide synchronized access to the object with the `lock` function. This function takes two parameters, the first of which is the variable to be changed. The second parameter is a lambda expression that changes the variable.

A mutable heap-allocated variable is of type `ref`. For example, the following declaration creates such a variable named `sum` with the initial value of `0`:

```
let sum = ref 0
```

A `ref` type variable can be changed in a lambda expression that uses the ALGOL/Pascal/Ada assignment operator, `:=`. The `ref` variable must be prefixed with an exclamation point (`!`) to get its value. In the following, the mutable variable `sum` is locked while the lambda expression adds the value of `x` to it:

```
lock(sum) (fun () -> sum := !sum + x)
```

Threads can be called asynchronously, just as with C#, using the same subprograms, `BeginInvoke` and `EndInvoke`, as well as the `IAsyncResult` interface to facilitate the determination of the completion of the execution of

the asynchronously called thread.

As stated previously, F# has the concurrent generic collections of .NET available to its programs. This can save a great deal of programming effort when building multithreaded programs that need a shared data structure in the form of a queue, stack, or bag.

13.10 Statement-Level Concurrency

In this section, we take a brief look at language design for statement-level concurrency. From the language design point of view, the objective of such designs is to provide a mechanism that the programmer can use to inform the compiler of ways it can map the program onto a multiprocessor architecture.9

9. Although ALGOL 68 included a semaphore type that was meant to deal with statement-level concurrency, we do not discuss that application of semaphores here.

In this section, only one collection of linguistic constructs from one language for statement-level concurrency is discussed: High-Performance Fortran.

13.10.1 High-Performance Fortran

High-Performance Fortran (HPF; ACM, 1993b) is a collection of extensions to Fortran 90 that are meant to allow programmers to specify information to the compiler to help it optimize the execution of programs on multiprocessor computers. HPF includes both new specification statements and intrinsic, or built-in, subprograms. This section discusses only some of the HPF statements.

The primary specification statements of HPF are for specifying the number of processors, the distribution of data over the memories of those processors, and the alignment of data with other data in terms of memory placement. The HPF specification statements appear as special comments in a Fortran program. Each of them is introduced by the prefix `!HPF$`, where `!` is the character used to begin lines of comments in Fortran 90. This prefix makes them invisible to Fortran 90 compilers but easy for HPF compilers to recognize.

The PROCESSORS specification has the following form:

```
!HPF$ PROCESSORS procs (n)
```

This statement is used to specify to the compiler the number of processors that can be used by the code generated for this program. This information is used in conjunction with other specifications to tell the compiler how data are to be distributed to the memories associated with the processors.

The DISTRIBUTE and ALIGN specifications are used to provide information to the compiler on machines that do not share memory—that is, each processor has its own memory. The assumption is that an access by a processor to its own memory is faster than an access to the memory of another processor.

The DISTRIBUTE statement specifies what data are to be distributed and the kind of distribution that is to be used. Its form is as follows:

```
!HPF$ DISTRIBUTE (kind) ONTO procs :: identifier_list
```

In this statement, kind can be either BLOCK or CYCLIC. The identifier list is the names of the array variables that are to be distributed. A variable that is specified to be BLOCK distributed is divided into *n* equal groups, where each group consists of contiguous collections of array elements evenly distributed over the memories of all the processors. For example, if an array with 500 elements named LIST is BLOCK distributed over 5 processors, the first 100 elements of LIST will be stored in the memory of the first processor, the second 100 in the memory of the second processor, and so forth. A CYCLIC distribution specifies that individual elements of the array are cyclically stored in the memories of the processors. For example, if LIST is CYCLIC distributed, again over five processors, the first element of LIST will be stored in the memory of the first processor, the second element in the memory of the second processor, and so forth.

The form of the ALIGN statement is

```
ALIGN array1_element WITH array2_element
```

ALIGN is used to relate the distribution of one array with that of another. For example,

```
ALIGN list1(index) WITH list2(index+1)
```

specifies that the `index` element of `list1` is to be stored in the memory of the same processor as the `index+1` element of `list2`, for all values of `index`. The two array references in an `ALIGN` appear together in some statement of the program. Putting them in the same memory (which means the same processor) ensures that the references to them will be as close as possible.

Consider the following example code segment:

```
     REAL list_1 (1000), list_2 (1000)
     INTEGER list_3 (500), list_4 (501)
!HPF$ PROCESSORS proc (10)
!HPF$ DISTRIBUTE (BLOCK) ONTO procs :: list_1, list_2
!HPF$ ALIGN list_3 (index) WITH list_4 (index+1)
     . . .
     list_1 (index) = list_2 (index)
     list_3 (index) = list_4 (index+1)
```

In each execution of these assignment statements, the two referenced array elements will be stored in the memory of the same processor.

The HPF specification statements provide information for the compiler that it may or may not use to optimize the code it produces. What the compiler actually does depends on its level of sophistication and the particular architecture of the target machine.

The `FORALL` statement specifies a sequence of assignment statements that may be executed concurrently. For example,

```
FORALL (index = 1:1000)
  list_1(index) = list_2(index)
END FORALL
```

specifies the assignment of the elements of `list_2` to the corresponding elements of `list_1`. However, the assignments are restricted to the following order: the right side of all 1,000 assignments must be evaluated first, before any assignments take place. This permits concurrent execution of all of the assignment statements. In addition to assignment statements, `FORALL` statements can appear in the body of a `FORALL` construct. The `FORALL` statement is a good match with vector machines, in which the same

instruction is applied to many data values, usually in one or more arrays. The HPF `FORALL` statement is included in Fortran 95 and subsequent versions of Fortran.

We have briefly discussed only a small part of the capabilities of HPF. However, it should be enough to provide the reader with an idea of the kinds of language extensions that are useful for programming computers with possibly large numbers of processors.

C# 4.0 (and the other .NET languages) include two methods that behave somewhat like `FORALL`. They are loop control statements in which the iterations can be unrolled and the bodies executed concurrently. These are `Parallel.For` and `Parallel.ForEach`.

SUMMARY

Concurrent execution can be at the instruction, statement, or subprogram level. We use the phrase *physical concurrency* when multiple processors are actually used to execute concurrent units. If concurrent units are executed on a single processor, we use the term *logical concurrency*. The underlying conceptual model of all concurrency can be referred to as *logical concurrency*.

Most multiprocessor computers fall into one of two broad categories—SIMD or MIMD. MIMD computers can be distributed.

Languages that support subprogram-level concurrency must provide two fundamental capabilities: mutually exclusive access to shared data structures (competition synchronization) and cooperation among tasks (cooperation synchronization).

Tasks can be in any one of five different states: new, ready, running, blocked, or dead.

Rather than designing language constructs for supporting concurrency, sometimes libraries, such as OpenMP, are used.

The design issues for language support for concurrency are how competition and cooperation synchronization are provided, how an application can influence task scheduling, how and when tasks start and end their executions, and how and when they are created.

A semaphore is a data structure consisting of an integer and a task description queue. Semaphores can be used to provide both competition and cooperation synchronization among concurrent tasks. It is easy to use semaphores incorrectly, resulting in errors that cannot be detected by the compiler, linker, or run-time system.

Monitors are data abstractions that provide a natural way of providing mutually exclusive access to data shared among tasks. They are supported by

several programming languages, among them Ada, Java, and C#. Cooperation synchronization in languages with monitors must be provided with some form of semaphores.

The underlying concept of the message-passing model of concurrency is that tasks send each other messages to synchronize their execution.

Ada provides complex but effective constructs, based on the message-passing model, for concurrency. Ada's tasks are heavyweight tasks. Tasks communicate with each other through the rendezvous mechanism, which is synchronous message passing. A rendezvous is the action of a task accepting a message sent by another task. Ada includes both simple and complicated methods of controlling the occurrences of rendezvous among tasks.

Ada 95+ includes additional capabilities for the support of concurrency, primarily protected objects. Ada 95+ supports monitors in two ways, with tasks and with protected objects.

Java supports lightweight concurrent units in a relatively simple but effective way. Any class that either inherits from `Thread` or implements `Runnable` can override a method named `run` and have that method's code executed concurrently with other such methods and with the main program. Competition synchronization is specified by defining methods that access shared data to be implicitly synchronized. Small sections of code can also be implicitly synchronized. A class whose methods are all synchronized is a monitor. Cooperation synchronization is implemented with the methods `wait`, `notify`, and `notifyAll`. The `Thread` class also provides the `sleep`, `yield`, `join`, and `interrupt` methods.

Java has direct support for counting semaphores through its `Semaphore` class and its `acquire` and `release` methods. It also had some classes for providing nonblocking atomic operations, such as addition, increment, and decrement operations for integers. Java also provides explicit locks with the `Lock` interface and `ReentrantLock` class and its `lock` and `unlock` methods. In addition to implicit synchronization using **`synchronized`**, Java provides implicit nonblocking synchronization of **`int`**, **`long`**, and **`boolean`** type variables, as well as references and arrays. In these cases, atomic getters, setters, add, increment, and decrement operations are provided.

C#’s support for concurrency is based on that of Java but is slightly more sophisticated. Any method can be run in a thread. Both actor and server threads are supported. All threads are controlled through associated delegates. Server threads can be synchronously called with `Invoke` or asynchronously called with `BeginInvoke`. A callback method address can be sent to the called thread. Three kinds of thread synchronization are supported with the `Interlocked` class, which provides atomic increment and decrement operations, the `Monitor` class, and the `lock` statement.

All .NET languages have the use of the generic concurrent data structures for stacks, queues, and bags, for which competition synchronization is implicit.

Multi-LISP extends Scheme slightly to allow the programmer to inform the implementation about program parts that can be executed concurrently. Concurrent ML extends ML to support a form of threads and a form of synchronous message passing among those threads. This message passing is designed with channels. F# programs have access to all of the .NET support classes for concurrency. Data shared among threads that is mutable can have access synchronized.

High-Performance Fortran includes statements for specifying how data is to be distributed over the memory units connected to multiple processors. Also included are statements for specifying collections of statements that can be executed concurrently.

BIBLIOGRAPHIC NOTES

- The general subject of concurrency is discussed at great length in Andrews and Schneider (1983), Holt et al. (1978), and Ben-Ari (1982).

- The monitor concept is developed and its implementation in Concurrent Pascal is described by Brinch Hansen (1977).

- The early development of the message-passing model of concurrent unit control is discussed by Hoare (1978) and Brinch Hansen (1978). An in-depth discussion of the development of the Ada tasking model can be found in Ichbiah et al. (1979). Ada 95 is described in detail in ARM (1995). High-Performance Fortran is described in ACM (1993b).

REVIEW QUESTIONS

1. What are the three possible levels of concurrency in programs?
2. Describe the logical architecture of an SIMD computer.
3. Describe the logical architecture of an MIMD computer.
4. What level of program concurrency is best supported by SIMD computers?
5. What level of program concurrency is best supported by MIMD computers?
6. Describe the logical architecture of a vector processor.
7. What is the difference between physical and logical concurrency?
8. What is a thread of control in a program?
9. Why are coroutines called quasi-concurrent?
10. What is a multithreaded program?
11. What are four reasons for studying language support for concurrency?
12. What is a heavyweight task? What is a lightweight task?
13. Define *task*, *synchronization*, *competition* and *cooperation synchronization*, *liveness*, *race condition*, and *deadlock*.
14. What kind of tasks do not require any kind of synchronization?
15. Describe the five different states in which a task can be.
16. What is a task descriptor?

17. In the context of language support for concurrency, what is a guard?

18. What is the purpose of a task-ready queue?

19. What are the two primary design issues for language support for concurrency?

20. Describe the actions of the wait and release operations for semaphores.

21. What is a binary semaphore? What is a counting semaphore?

22. What are the primary problems with using semaphores to provide synchronization?

23. What advantage do monitors have over semaphores?

24. In what three common languages can monitors be implemented?

25. Define *rendezvous*, **`accept`** *clause*, **`entry`** *clause*, *actor task*, *server task*, *extended* **`accept`** *clause*, *open* **`accept`** *clause*, *closed* **`accept`** *clause*, and *completed task*.

26. Which is more general, concurrency through monitors or concurrency through message passing?

27. Are Ada tasks created statically or dynamically?

28. What purpose does an extended **`accept`** clause serve?

29. How is cooperation synchronization provided for Ada tasks?

30. What is the advantage of protected objects in Ada 95 over tasks for providing access to shared data objects?

31. Specifically, what Java program unit can run concurrently with the main method in an application program?

32. Are Java threads lightweight or heavyweight tasks?

33. What does the Java `sleep` method do?

34. What does the Java `yield` method do?

35. What does the Java `join` method do?

36. What does the Java `interrupt` method do?

37. What are the two Java constructs that can be declared to be `synchronized`?

38. How can the priority of a thread be set in Java?

39. Can Java threads be actor threads, server threads, or either?

40. Describe the actions of the three Java methods that are used to support cooperation synchronization.

41. What kind of Java object is a monitor?

42. Explain why Java includes the `Runnable` interface.

43. What are the two methods used with Java `Semaphore` objects?

44. What is the advantage of the nonblocking synchronization in Java?

45. What are the methods of the Java `AtomicInteger` class and what is the purpose of this class?

46. How are explicit locks supported in Java?

47. What kinds of methods can be run in a C# thread?

48. Can C# threads be actor threads, server threads, or either?

49. What are the two ways a C# thread can be called synchronously?

50. How can a C# thread be called asynchronously?

51. How is the returned value from an asynchronously called thread retrieved in C#?

52. What is different about C#'s `Sleep` method, relative to Java's `sleep`?

53. What exactly does C#'s `Abort` method do?

54. What is the purpose of C#'s `Interlocked` class?

55. What does the C# `lock` statement do?

56. On what language is Multi-LISP based?

57. What is the semantics of Multi-LISP's `pcall` construct?

58. How is a thread created in CML?

59. What is the type of an F# heap-allocated mutable variable?

60. Why don't F# immutable variables require synchronized access in a multithreaded program?

61. What is the objective of the specification statements of HPF?

62. What is the purpose of the `FORALL` statement of HPF and Fortran?

PROBLEM SET

1. Explain clearly why competition synchronization is not a problem in a programming environment that supports coroutines but not concurrency.

2. What is the best action a system can take when deadlock is detected?

3. Busy waiting is a method whereby a task waits for a given event by continuously checking for that event to occur. What is the main problem with this approach?

4. In the producer-consumer example of Section 13.3, suppose that we incorrectly replaced the `release(access)` in the consumer process with `wait(access)`. What would be the result of this error on execution of the system?

5. From a book on assembly language programming for a computer that uses an Intel Pentium processor, determine what instructions are provided to support the construction of semaphores.

6. Suppose two tasks, `A` and `B`, must use the shared variable `Buf_Size`. Task `A` adds 2 to `Buf_Size`, and task `B` subtracts 1 from it. Assume that such arithmetic operations are done by the three-step process of fetching the current value, performing the arithmetic, and putting the new value back. In the absence of competition synchronization, what sequences of events are possible and what values result from these operations? Assume that the initial value of `Buf_Size` is 6.

7. Compare the Java competition synchronization mechanism with that of Ada.

8. Compare the Java cooperation synchronization mechanism with that of Ada.

9. What happens if a monitor procedure calls another procedure in the same monitor?

10. Explain the relative safety of cooperation synchronization using semaphores and using Ada's **when** clauses in tasks.

PROGRAMMING EXERCISES

1. Write an Ada task to implement general semaphores.

2. Write an Ada task to manage a shared buffer such as the one in our example, but use the semaphore task from Programming Exercise 1.

3. Define semaphores in Ada and use them to provide both cooperation and competition synchronization in the shared-buffer example.

4. Write Programming Exercise 3 using Java.

5. Write the shared-buffer example of the chapter in C#.

6. The reader-writer problem can be stated as follows: A shared memory location can be concurrently read by any number of tasks, but when a task must write to the shared memory location, it must have exclusive access. Write a Java program for the reader-writer problem.

7. Write Programming Exercise 6 using Ada.

8. Write Programming Exercise 6 using C#.

14 Exception Handling and Event Handling

This chapter discusses programming language support for two related parts of many contemporary programs: exception handling and event handling. Both exceptions and events occur at times that cannot be predetermined, and both are best handled with special language constructs and processes. Some of these constructs and processes—for example, propagation—are similar for exception handling and event handling.

We first describe the fundamental concepts of exception handling, including hardware- and software-detectable exceptions, exception handlers, and the raising of exceptions. Then, the design issues for exception handling are introduced and discussed, including the binding of exceptions to exception handlers, continuation, and default handlers. This section is followed by descriptions and evaluations of the exception-handling facilities of two programming languages: C++ and Java. Brief introductions to exception handling in Python and Ruby are then presented.

The latter part of this chapter is about event handling. We first present an - introduction to the basic concepts of event handling. This is followed by

discussions of the event-handling approaches of Java and C#.

14.1 Introduction to Exception Handling

Most computer hardware systems are capable of detecting certain run-time error conditions, such as floating-point overflow. Early programming - languages were designed and implemented in such a way that the user program could neither detect nor attempt to deal with such errors. In these languages, the occurrence of such an error simply causes the program to be terminated and control to be transferred to the operating system. The typical operating system reaction to a run-time error is to display a diagnostic message, which may be meaningful and therefore useful, or highly cryptic. After displaying the message, the program is terminated.

In the case of input and output operations, however, the situation is somewhat different. For example, a Fortran `Read` statement can intercept input errors and end-of-file conditions, both of which are detected by the input device hardware. In both cases, the `Read` statement can specify the label of some statement in the user program that deals with the condition. In the case of the end-of-file, the condition obviously is not always considered an error. In most cases, it is nothing more than a signal that one kind of processing is completed and another kind must begin. In spite of the obvious difference between end-of-file and events that are always errors, such as a failed input process, Fortran handles both situations with the same mechanism. Consider the following Fortran `Read` statement:

```
Read(Unit=5, Fmt=1000, Err=100, End=999) Weight
```

The `Err` clause specifies that control is to be transferred to the statement labeled `100` if an error occurs in the read operation. The `End` clause specifies that control is to be transferred to the statement labeled `999` if the read operation encounters the end of the file. So, Fortran uses simple branches for both input errors and end-of-file.

There is a category of serious errors that are not detectable by hardware but

can be detected by code generated by the compiler. For example, array subscript range errors are almost never detected by hardware,[1] but they lead to serious errors that often are not noticed until later in the program execution.

1. In the 1970s, there were some computers that *did* detect subscript range errors in hardware.

Detection of subscript range errors is sometimes required by the language design. For example, the Java language specification requires Java compilers to generate code to check the correctness of every subscript expression (they do not generate such code when it can be determined at compile time that a subscript expression cannot have an out-of-range value, for example, if the subscript is a literal). In C, subscript ranges are not checked because the cost of such checking was (and still is) not believed to be worth the benefit of detecting such errors. In some compilers for some languages, subscript range checking can be selected (if not turned on by default) or turned off (if it is on by default) as desired in the program or in the command that executes the compiler.

The designers of most contemporary languages have included mechanisms that allow programs to react in a standard way to certain run-time errors, as well as other program-detected unusual events. Programs may also be notified when certain events are detected by hardware or system software, so that they also can react to these events.

14.1.1 Basic Concepts

We consider both the errors detected by hardware, such as disk read errors, and unusual conditions, such as end-of-file (which is also detected by hardware), to be exceptions. We further extend the concept of an exception to include errors or unusual conditions that are software-detectable (by either a software interpreter or the user code itself). Accordingly, we define **exception** to be any unusual event, erroneous or not, that is detectable by either hardware or software and that may require special processing.

The special processing that may be required when an exception is detected is called **exception handling**. This processing is done by a code unit or segment called an **exception handler**. An exception is **raised** when its associated event occurs. In some C-based languages, exceptions are said to be *thrown*, rather than *raised*.2 Different kinds of exceptions require different exception handlers. Detection of end-of-file nearly always requires some specific program action. But, clearly, that action would not also be appropriate for an array index range error exception. In some cases, the only action is the generation of an error message and an orderly termination of the program.

2. C++ was the first C-based language that included exception handling. The word *throw* was used, rather than *raise*, because the standard C library includes a function named `raise`.

The absence of separate or specific exception-handling facilities in a language does not preclude the handling of user-defined, software-detected exceptions. Such an exception detected within a program unit is often handled by the unit's caller. One possible design is to send an auxiliary parameter, which is used as a status variable. The status variable is assigned a value in the called subprogram according to the correctness and/or normalness of the results of its computations. Immediately upon return from the called unit, the caller tests the status variable. If the value indicates that an exception has occurred, the handler, which may reside in the calling unit, can be enacted. Many of the C standard library functions use a variant of this approach: The return values are used as error indicators.

Another possibility is to pass a label parameter to the subprogram. Of course, this approach is possible only in languages that allow labels to be used as parameters. Passing a label allows the called unit to return to a different point in the caller if an exception has occurred. As in the first alternative, the handler is often a segment of the calling unit's code. This is a common use of label parameters in Fortran.

A third possibility is to have the handler defined as a separate subprogram whose name is passed as a parameter to the called unit. In this case, the handler subprogram is provided by the caller, but the called unit calls the handler when an exception is raised. One problem with this approach is that one is required to send a handler subprogram with *every* call to *every*

subprogram that takes a handler subprogram as a parameter, whether it is needed or not. Furthermore, to deal with several different kinds of exceptions, several different handler routines would need to be passed, complicating the code.

If it is desirable to handle an exception in the unit in which it is detected, the handler is included as a segment of code in that unit.

There are some definite advantages to having exception handling built into a language. First, without exception handling, the code required to detect error conditions can considerably clutter a program. For example, suppose a subprogram includes expressions that contain 10 references to elements of a matrix named `mat`, and any one of them could have an index out-of-range error. Further suppose that the language does not require index range checking. Without built-in index range checking, every one of these operations may need to be preceded by code to detect a possible index range error. For example, consider the following reference to an element of `mat`, which has 10 rows and 20 columns:

```
if (row >= 0 && row < 10 && col >= 0 && col < 20)
  sum += mat[row][col];
else
  System.out.println("Index range error on mat, row = " +
                     row + " col = " + col);
```

The presence of exception handling in the language would permit the -compiler to insert machine code for such checks before every array element access, greatly shortening and simplifying the source program.

Another advantage of language support for exception handling results from exception propagation. Exception propagation allows an exception raised in one program unit to be handled in some other unit in its dynamic or static ancestry. This allows a single exception handler to be used for any number of different program units. This reuse can result in significant savings in development cost, program size, and program complexity.

A language that supports exception handling encourages its users to consider all of the events that could occur during program execution and how they can be handled. This approach is far better than not considering such possibilities

and simply hoping nothing will go wrong.

Finally, there are programs in which dealing with nonerroneous but unusual situations can be simplified with exception handling, and in which program structure can become overly convoluted without it.

14.1.2 Design Issues

We now explore some of the design issues for an exception-handling system when it is part of a programming language. Such a system might allow both predefined and user-defined exceptions and exception handlers. Note that predefined exceptions are implicitly raised, whereas user-defined exceptions must be explicitly raised by user code. Consider the following skeletal subprogram that includes an exception-handling mechanism for an implicitly raised exception:

```
void example() {
  . . .
  average = sum / total;
  . . .
  return;
/* Exception handlers */
  when zero_divide {
    average = 0;
    printf("Error–divisor (total) is zero\n");
  }
  . . .
}
```

The exception of division by zero, which is implicitly raised, causes control to transfer to the appropriate handler, which is then executed.

The first design issue for exception handling is how an exception occurrence is bound to an exception handler. This issue occurs on two different levels. On the unit level, there is the question of how the same exception being raised at different points in a unit can be bound to different handlers within the unit. For example, in the example subprogram, there is a handler for a -division-by-zero exception that appears to be written to deal with an occurrence of division by zero in a particular statement (the one shown). But

suppose the function includes several other expressions with division operators. For those operators, this handler would probably not be appropriate. So, it should be possible to bind the exceptions that can be raised by particular statements to particular handlers, even though the same exception can be raised by many different statements.

At a higher level, the binding question arises when there is no exception handler local to the unit in which the exception is raised. In this case, the language designer must decide whether to propagate the exception to some other unit and, if so, where. How this propagation takes place and how far it goes have an important impact on the writability of exception handlers. For example, if handlers must be local, then many handlers must be written, which complicates both the writing and reading of the program. On the other hand, if exceptions are propagated, a single handler might handle the same exception raised in several program units, which may require the handler to be more general than one would prefer.

An issue that is related to the binding of an exception to an exception handler is whether information about the exception is made available to the handler.

history note

PL/I (ANSI, 1976) pioneered the concept of allowing user programs to be directly involved in exception handling. The language allowed the user to write exception handlers for a long list of language-defined exceptions. Furthermore, PL/I introduced the concept of user-defined exceptions, which allow programs to create software-detected exceptions. These exceptions use the same mechanisms that are used for the built-in exceptions.

Since PL/I was designed, a substantial amount of work has been done to design alternative methods of exception handling, and exception-handling mechanisms have been included in a long list of subsequent programming languages.

After an exception handler executes, either control can transfer to somewhere in the program outside of the handler code or program execution can simply

terminate. We term this the question of control continuation after handler execution, or simply **continuation**. Termination is obviously the simplest choice, and in many error exception conditions, the best. However, in other situations, particularly those associated with unusual but not erroneous events, the choice of continuing execution is best. This design is called **resumption**. In these cases, some conventions must be chosen to determine where execution should continue. It might be the statement that raised the exception, the statement after the statement that raised the exception, or possibly some other unit. The choice to return to the statement that raised the exception may seem like a good one, but in the case of an error exception, it is useful only if the handler somehow is able to modify the values or operations that caused the exception to be raised. Otherwise, the exception will simply be reraised. The required modification for an error exception is often very difficult to predict. Even when possible, however, it may not be a sound practice. It allows the program to remove the symptom of a problem without removing the cause.

The two issues of binding of exceptions to handlers and continuation are illustrated in Figure 14.1.

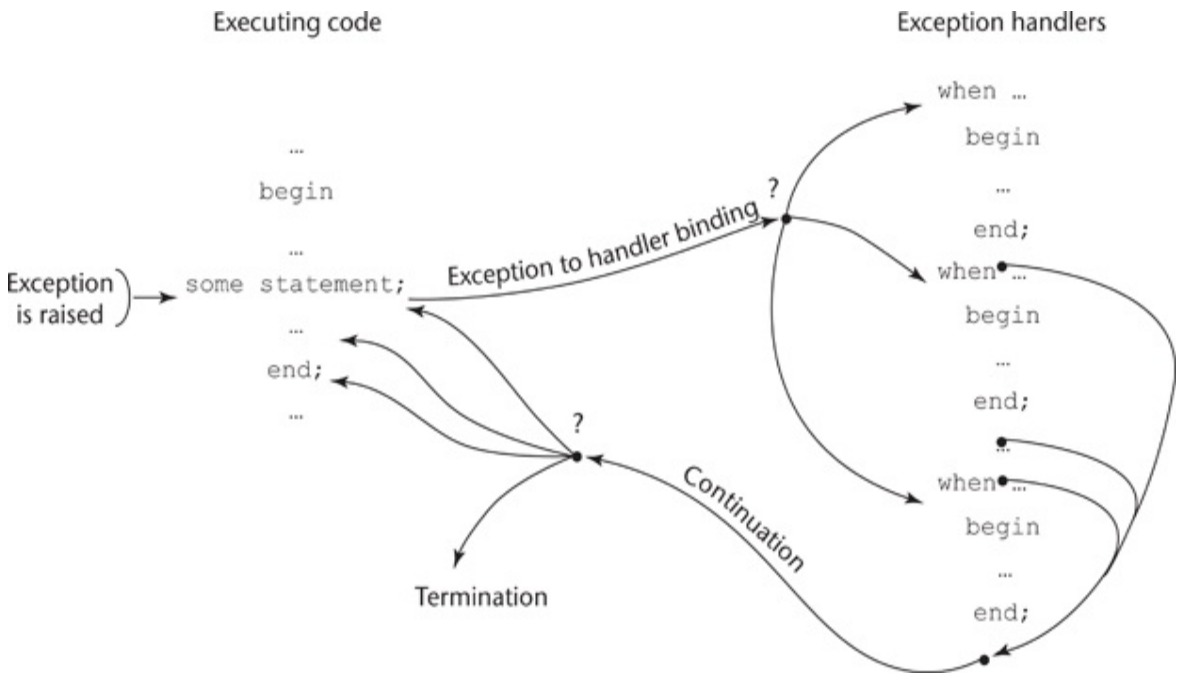


Figure 14.1 Exception-handling

control flow

Figure 14.1 Full Alternative Text

When exception handling is included, a subprogram's execution can terminate in two ways: when its execution is complete or when it encounters an exception.3 In some situations, it is necessary to complete some computation regardless of how subprogram execution terminates. The ability to specify such a computation is called *finalization*. The choice of whether to support finalization is obviously a design issue for exception handling.

3. Of course, even if the language does not support exception handling, a subprogram could terminate due to a system-detected error.

Another design issue is the following: If users are allowed to define exceptions, how are these exceptions specified? The usual answer is to require that they be declared in the specification parts of the program units in which they can be raised. The scope of a declared exception is usually the scope of the program unit that contains the declaration.

In the case where a language provides predefined exceptions, several other design issues follow. For example, should the language run-time system provide default handlers for the built-in exceptions, or should the user be required to write handlers for all exceptions? Another question is whether predefined exceptions can be raised explicitly by the user program. This usage can be convenient if there are software-detectable situations in which the user would like to use a predefined handler.

Another issue is whether hardware-detectable errors can be handled by user programs. If not, all exceptions obviously are software detectable. A related question is whether there should be any predefined exceptions. Predefined exceptions are implicitly raised by either hardware or system software.

The exception-handling design issues can be summarized as follows:

- How and where are exception handlers specified, and what is their scope?

- How is an exception occurrence bound to an exception handler?
- Can information about an exception be passed to the handler?
- Where does execution continue, if at all, after an exception handler completes its execution? (This is the question of continuation or resumption.)
- Is some form of finalization provided?
- How are user-defined exceptions specified?
- If there are predefined exceptions, should there be default exception handlers for programs that do not provide their own?
- Can predefined exceptions be explicitly raised?
- Are hardware-detectable errors treated as exceptions that may be handled?
- Are there any predefined exceptions?

We are now in a position to examine the exception-handling facilities of several contemporary programming languages.

14.2 Exception Handling in C++

The exception handling of C++ was accepted by the ANSI C++ standardization committee in 1990 and subsequently found its way into C++ implementations. The design is based in part on the exception handling of CLU, Ada, and ML.

14.2.1 Exception Handlers

C++ uses a special construct that is introduced with the reserved word **`try`** to specify the scope for exception handlers. A **`try`** construct includes a compound statement called the **try clause** and a list of exception handlers. The compound statement defines the scope of the following handlers. The general form of this construct is as follows:

```
try {
//** Code that might raise an exception
} catch (formal parameter) {
//** A handler body
}
. . .
catch(formal parameter) {
//** A handler body
}
```

Each **`catch`** function is an exception handler. A **`catch`** function can have only a single formal parameter, which is similar to a formal parameter in a function definition in C++, including the possibility of it being an ellipsis (. . .). A handler with an ellipsis formal parameter is the catch-all handler; it is enacted for any raised exception if no appropriate handler was found. The formal parameter also can be a naked type specifier, such as **`float`**, as in a function prototype. In such a case, the only purpose of the formal parameter is to make the handler uniquely identifiable. When information about the exception is to be passed to the handler, the formal parameter includes a variable name that is used for that purpose. Because the class of the

parameter can be any user-defined class, the parameter can include as many data members as are necessary. Binding exceptions to handlers are discussed in Section 14.3.2.

In C++, exception handlers can include any C++ code.

14.2.2 Binding Exceptions to Handlers

C++ exceptions are raised only by the explicit statement **`throw`**, whose general form in EBNF is as follows:

```
throw [expression];
```

The brackets here are metasymbols used to specify that the expression is optional. A **`throw`** without an operand can appear only in a handler. When it appears there, it reraises the exception, which is then handled elsewhere.

The type of the **`throw`** expression selects the particular handler, which of course must have a "matching" type formal parameter. In this case, *matching* means the following: A handler with a formal parameter of type `T`, **`const`** `T`, `T&` (a reference to an object of type `T`), or **`const`** `T&` matches a **`throw`** with an expression of type `T`. In the case where `T` is a class, a handler whose parameter is type `T` or any class that is an ancestor of `T` matches. There are more complicated situations in which a **`throw`** expression matches a formal parameter, but they are not described here.

An exception raised in a **`try`** clause causes an immediate end to the execution of the code in that **`try`** clause. The search for a matching handler begins with the handlers that immediately follow the **`try`** clause. The matching process is done sequentially on the handlers until a match is found. This means that if any other match precedes an exactly matching handler, the exactly matching handler will not be used. Therefore, handlers for specific exceptions are placed at the top of the list, followed by more generic handlers. The last handler is often one with an ellipsis (. . .) formal parameter, which matches

any exception. This would guarantee that all exceptions are caught.

If an exception is raised in a **try** clause and there is no matching handler associated with that **try** clause, the exception is propagated. If the **try** clause is nested inside another **try** clause, the exception is propagated to the handlers associated with the outer **try** clause. If none of the enclosing **try** clauses yields a matching handler, the exception is propagated to the caller of the function in which it was raised. If the call to the function was not in a **try** clause, the exception is propagated to that function's caller. If no matching handler is found in the program through this propagation process, the default handler is called. This handler is further discussed in Section 14.2.4.

14.2.3 Continuation

After a handler has completed its execution, control flows to the first statement following the **try** construct (the statement immediately after the last handler in the sequence of handlers of which it is an element). A handler may reraise an exception, using a **throw** without an expression, in which case that exception is propagated.

14.2.4 Other Design Choices

In terms of the design issues summarized in Section 14.1.2, the exception handling of C++ is simple. There are *only* user-defined exceptions, and they are not specified (though they might be declared as new classes). There is a default exception handler, `unexpected`, whose only action is to terminate the program. This handler catches all exceptions not caught by the program. It can be replaced by a user-defined handler. The replacement handler must be a function that returns **void** and takes no parameters. The replacement function is set by assigning its name to `set_terminate`.

A C++ function can list the types of the exceptions (the types of the **throw** expressions) that it could raise. This is done by attaching the reserved word **throw**, followed by a parenthesized list of these types, to the function header.

For example,

```
int fun() throw (int, char *) { . . . }
```

specifies that the function `fun` could raise exceptions of type **`int`** and **`char *`** but no others. The purpose of the **`throw`** clause is to notify users of the function what exceptions might be raised by the function. The **`throw`** clause is in effect a contract between the function and its callers. It guarantees that no other exception will be raised in the function.

If the types in the **`throw`** clause are classes, then the function can raise any exception that is derived from the listed classes. If a function header has a **`throw`** clause and raises an exception that is not listed in the **`throw`** clause and is not derived from a class listed there, the default handler is called. Note that this error cannot be detected at compile time. The list of types in the list may be empty, meaning that the function will not raise any exceptions. If there is no **`throw`** specification on the header, the function can raise any exception. The list is not part of the function's type.

If a function overrides a function that has a **`throw`** clause, the overriding function cannot have a **`throw`** clause with more exceptions than the overridden function.

Although C++ has no predefined exceptions, the standard libraries define and throw exceptions, such as `out_of_range`, which can be thrown by library container classes, and `overflow_error`, which can be thrown by math library functions.

14.2.5 An Example

The following example program illustrates some simple uses of exception handlers in C++. The program computes and prints a distribution of input grades by using an array of counters. The input is a sequence of grades, terminated by a negative number. The negative number raises a `NegativeInputException` exception because the grades must be nonnegative integers. There are 10 categories of grades (0−9, 10−19, … , 90−100). The grades themselves

are used to compute indexes into an array of counters, one for each grade category. Invalid input grades are detected by trapping indexing errors in the counter array. A grade of 100 is special in the computation of the grade distribution because the categories all have 10 possible grade values, except the highest, which has 11 (90, 91, . . . , 100). (The fact that there are more possible A grades than B's or C's is conclusive evidence of the generosity of teachers.) The grade of 100 is also handled in the same exception handler that is used for invalid input data.

```
// Grade Distribution
//  Input: A list of integer values that represent
//         grades, followed by a negative number
// Output: A distribution of grades, as a percentage for
//         each of the categories 0-9, 10-19, . . .,
//         90-100.
#include <iostream>
int main() { //* Any exception can be raised
  int new_grade,
      index,
      limit_1,
      limit_2,
      freq[10] = {0,0,0,0,0,0,0,0,0,0};
// The exception definition to deal with the end of data
class NegativeInputException {
  public:
   NegativeInputException() {  //* Constructor
    cout << ”End of input data reached" << endl;
  }  //** end of constructor
}  //** end of NegativeInputException class
  try {
     while (true) {
       cout << ”Please input a grade" << endl;
       if ((cin >> new_grade) < 0)  //* End of data
         throw  NegativeInputException();
      index = new_grade / 10;
      {try {
        if (index > 9)
          throw  new_grade;
        freq[index]++;
        }  //* end of inner try compound
      catch(int  grade) {  //* Handler for index errors
        if (grade == 100)
          freq[9]++;
        else
          cout << ”Error -- new grade: " << grade
```

```
                << " is out of range" << endl;
        }  //* end of catch(int grade)
      }  //*  end of the block for the inner try-catch pair
    }  //* end of while (1)
  }  //* end of outer try block
  catch(NegativeInputException& e) {  //**Handler for
                                      //** negative input
    cout << ”Limits   Frequency" << endl;
    for (index = 0; index < 10; index++) {
      limit_1 = 10 * index;
      limit_2 = limit_1 + 9;
      if (index == 9)
        limit_2 = 100;
      cout << limit_1 << limit_2 << freq[index] << endl;
    }  //* end of for (index = 0)
  }  //* end of catch (NegativeInputException& e)
}  //* end of main
```

This program is meant to illustrate the mechanics of C++ exception handling. Note that the index range exception is often handled in C++ by overloading the indexing operation, which could then raise the exception, rather than the direct detection of the indexing operation with the selection construct used in our example.

14.2.6 Evaluation

One deficiency of exception handling in C++ is that there are no predefined hardware-detectable exceptions that can be handled by the user. Exceptions are connected to handlers through a parameter type in which the formal parameter may be omitted. The type of the formal parameter of a handler determines the condition under which it is called but may have nothing whatsoever to do with the nature of the raised exception. Therefore, the use of predefined types for exceptions certainly does not promote readability. It is much better to define classes for exceptions with meaningful names in a meaningful hierarchy that can be used for defining exceptions. The exception parameter provides a way to pass information about an exception to the exception handler.

14.3 Exception Handling in Java

In Chapter 13, the Java example program includes the use of exception - handling with little explanation. This section describes the details of Java's exception-handling capabilities.

Java's exception handling is based on that of C++, but it is designed to be more in line with the object-oriented language paradigm. Furthermore, Java includes a collection of predefined exceptions that are implicitly raised by the Java run-time system.

14.3.1 Classes of Exceptions

All Java exceptions are objects of classes that are descendants of the `Throwable` class. The Java system includes two predefined exception classes that are subclasses of `Throwable`: `Error` and `Exception`. The `Error` class and its descendants are related to errors that are thrown by the Java run-time system, such as running out of heap memory. These exceptions are never thrown by user programs, and they should never be handled there. There are two system-defined direct descendants of `Exception`: `RuntimeException` and `IOException`. As its name indicates, `IOException` is thrown when an error has occurred in an input or output operation, all of which are defined as methods in the various classes defined in the package `java.io`.

There are predefined classes that are descendants of `RuntimeException`. In most cases, `RuntimeException` is thrown when a user program causes an error. For example, `ArrayIndexOutOfBoundsException`, which is defined in `java.util`, is a commonly thrown exception that descends from `RuntimeException`. Another commonly thrown exception that descends from `RuntimeException` is `NullPointerException`.

User programs can define their own exception classes. The convention in Java is that user-defined exceptions are subclasses of `Exception`.

14.3.2 Exception Handlers

The exception handlers of Java have the same form as those of C++, except that every **`catch`** must have a parameter and the class of the parameter must be a descendant of the predefined class `Throwable`.

The syntax of the **`try`** construct in Java is exactly as that of C++, except for the **`finally`** clause described in Section 14.3.6.

14.3.3 Binding Exceptions to Handlers

Throwing an exception is simple. An instance of the exception class is given as the operand of the **`throw`** statement. For example, suppose we define an exception named `MyException` as

```
class MyException extends Exception {
  public MyException() {}
  public MyException(String message) {
    super (message);
  }
}
```

This exception can be thrown with the following statement:

```
throw new MyException();
```

One of the two constructors we have included in our new class has no parameter and the other has a `String` object parameter that it sends to the superclass (`Exception`), which displays it. Therefore, our new exception could be thrown with

```
throw new MyException
       ("a message to specify the location of the error");
```

The binding of exceptions to handlers in Java is similar to that of C++. If an

exception is thrown in the compound statement of a **try** construct, it is bound to the first handler (**catch** function) immediately following the **try** clause whose parameter is the same class as the thrown object, or an ancestor of it. If a matching handler is found, the **throw** is bound to it and it is executed.

Exceptions can be handled and then rethrown by including a **throw** statement without an operand at the end of the handler. The newly thrown exception will not be handled in the same **try** where it was originally thrown, so looping is not a concern. This rethrowing is usually done when some local action is useful, but further handling by an enclosing **try** clause or a **try** clause in the caller is necessary. A **throw** statement in a handler could also throw some exception other than the one that transferred control to this handler.

To ensure that exceptions that can be thrown in a **try** clause are always handled in a method, a special handler can be written that matches all exceptions that are derived from `Exception` simply by defining the handler with an `Exception` type parameter, as in

```
catch (Exception genericObject) {
  . . .
}
```

Because a class name always matches itself or any ancestor class, any class derived from `Exception` matches `Exception`. Of course, such an exception handler should always be placed at the end of the list of handlers, for it will block the use of any handler that follows it in the **try** construct in which it appears. This occurs because the search for a matching handler is sequential, and the search ends when a match is found.

14.3.4 Other Design Choices

As part of its reflection facilities, the Java run-time system stores the class name of every object in the program. The method `getClass` can be used to get an object that stores the class name, which itself can be gotten with the `getName` method. So, we can retrieve the name of the class of the actual parameter from the **throw** statement that caused the handler's execution. For

the handler shown earlier, this is done with

```
genericObject.getClass().getName()
```

In addition, the message associated with the parameter object, which is created by the constructor, can be gotten with

```
genericObject.getMessage()
```

Furthermore, in the case of user-defined exceptions, the thrown object could include any number of data fields that might be useful in the handler.

The **`throws`** clause of Java has the appearance and placement (in a program) that is similar to that of the **`throw`** specification of C++. However, the semantics of **`throws`** is somewhat different from that of the C++ **`throw`** clause.

The appearance of an exception class name in the **`throws`** clause of a Java method specifies that that exception class or any of its descendant exception classes can be thrown but not handled by the method. For example, when a method specifies that it can throw `IOException`, it means it can throw an `IOException` object or an object of any of its descendant classes, such as `EOFException`, and it does not handle the exception it throws.

Exceptions of class `Error` and `RuntimeException` and their descendants are called **unchecked exceptions**. All other exceptions are called **checked exceptions**. Unchecked exceptions are never a concern of the compiler. However, the compiler ensures that all checked exceptions a method can throw are either listed in its **`throws`** clause or handled in the method. Note that checking this at compile time differs from C++, in which it is done at run time. The reason why exceptions of the classes `Error` and `RuntimeException` and their descendants are unchecked is that any method could throw them. A program can catch unchecked exceptions, but it is not required.

As is the case with C++, a method cannot declare more exceptions in its **`throws`** clause than the method it overrides, though it may declare fewer. So if a method has no **`throws`** clause, neither can any method that overrides it. A method can throw any exception listed in its **`throws`** clause, along with any of the descendant classes of those exceptions.

A method that does not directly throw a particular exception, but calls another method that could throw that exception, must list the exception in its **throws** clause. This is the reason the `buildDist` method (in the example in the next subsection), which uses the `readLine` method, must specify `IOException` in the **throws** clause of its header.

A method that does not include a **throws** clause cannot propagate any checked exception. Recall that in C++, a function without a **throw** clause can throw *any* exception.

A method that calls a method that lists a particular checked exception in its **throws** clause has three alternatives for dealing with that exception: First, it can catch the exception and handle it. Second, it can catch the exception and throw an exception that is listed in its own **throws** clause. Third, it could declare the exception in its own **throws** clause and not handle it, which effectively propagates the exception to an enclosing **try** clause, if there is one, or to the method's caller, if there is no enclosing **try** clause.

There are no default exception handlers, and it is not possible to disable exceptions. Continuation in Java is exactly as in C++.

14.3.5 An Example

Following is the Java program with the capabilities of the C++ program in Section 14.2.5:

```
// Grade Distribution
//  Input: A list of integer values that represent
//         grades, followed by a negative number
// Output: A distribution of grades, as a percentage for
//         each of the categories 0-9, 10-19, . . .,
//         90-100.
import java.io.*;
// The exception definition to deal with the end of data
class NegativeInputException extends Exception {
  public NegativeInputException() {
    System.out.println("End of input data reached");
  }  //** end of constructor
}  //** end of NegativeInputException class
```

```
class GradeDist {
  int newGrade,
      index,
       limit_1,
       limit_2;
  int [] freq = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
void buildDist() throws  IOException {
  DataInputStream in =  new  DataInputStream(System.in);
  try {
    while (true) {
      System.out.println("Please input a grade");
      newGrade = Integer.parseInt(in.readLine());
      if (newGrade < 0)
        throw new NegativeInputException();
      index = newGrade / 10;
      try {
        freq[index]++;
      }  //** end of inner try clause
      catch(ArrayIndexOutOfBoundsException e) {
        if (newGrade == 100)
          freq [9]++;
        else
          System.out.println("Error - new grade: " +
                     newGrade + " is out of range");
      }  //** end of catch (ArrayIndex. . .
    }  //** end of while (true) . . .
  }  //** end of outer try clause
  catch(NegativeInputException e) {
    System.out.println ("\nLimits    Frequency\n");
    for (index = 0; index < 10; index++) {
      limit_1 = 10 * index;
      limit_2 = limit_1 + 9;
      if (index == 9)
        limit_2 = 100;
      System.out.println("" + limit_1 + " - " +
                      limit_2 + "     " + freq [index]);
    }  //** end of for (index = 0; ...
  }  //** end of catch (NegativeInputException ...
}  //** end of method buildDist
```

The exception for a negative input, `NegativeInputException`, is defined in the program. Its constructor displays a message when an object of the class is created. Its handler produces the output of the method. `ArrayIndexOutOfBoundsException` is a predefined unchecked exception that is thrown by the Java run-time system. In both of these cases, the handler does not include an object name in its parameter. In neither case would a

name serve any purpose. Although all handlers get objects as parameters, they often are not useful.

14.3.6 The `finally` Clause

There are some situations in which a process must be executed regardless of whether a **`try`** clause throws an exception or the exception is handled in the method. One example of such a situation is a file that must be closed. Another is if the method has some external resource that must be freed in the method regardless of how the execution of the method terminates. The **`finally`** clause was designed for these kinds of needs. A **`finally`** clause is placed at the end of the list of handlers just after a complete **`try`** construct. In general, the **`try`** construct and its **`finally`** clause appear as

```
try {
  . . .
}
catch (. . .) {
  . . .
}
. . . //** More handlers
finally  {
  . . .
}
```

The semantics of this construct is as follows: If the **`try`** clause throws no exceptions, the **`finally`** clause is executed before execution continues after the **`try`** construct. If the **`try`** clause throws an exception and it is caught by a following handler, the **`finally`** clause is executed after the handler completes its execution. If the **`try`** clause throws an exception but it is not caught by a handler following the **`try`** construct, the **`finally`** clause is executed before the exception is propagated.

A **`try`** construct with no exception handlers can be followed by a **`finally`** clause. This makes sense, of course, only if the compound statement has a **`throw`**, **`break`**, **`continue`**, or **`return`** statement. Its purpose in these cases is the same as when it is used with exception handling. For example, consider the following:

```
try {
  for (index = 0; index < 100; index++) {
    . . .
    if (. . . ) {
      return;
    }  //** end of if
    . . .
  }  //** end of for
}  //** end of try clause
finally  {
 . . .
}  //** end of try construct
```

The **finally** clause here will be executed, regardless of whether the **return** terminates the loop or it ends normally.

14.3.7 Assertions

In the discussion of Plankalkül in Chapter 2, we mentioned that it included assertions. Assertions were added to Java in version 1.4. To use them, it is necessary to enable them by running the program with the `enableassertions` (or `ea`) flag, as in

```
java -enableassertions MyProgram
```

There are two possible forms of the **assert** statement:

assert condition;

assert condition : expression;

In the first case, the condition is tested when execution reaches the **assert**. If the condition evaluates to true, nothing happens. If it evaluates to false, the `AssertionError` exception is thrown. In the second case, the action is the same, except that the value of the expression is passed to the `AssertionError` constructor as a string and becomes debugging output.

The **assert** statement is used for defensive programming. A program may be written with many **assert** statements, which ensure that the program's

computation is on track to produce correct results. Many programmers put in such checks when they write a program, as an aid to debugging, even though the language they are using does not support assertions. When the program is sufficiently tested, these checks are removed. The advantage of **`assert`** statements, which have the same purpose, is that they can be disabled without removing them from the program. This saves the effort of removing them and also allows their use during subsequent program maintenance.

14.3.8 Evaluation

The Java mechanisms for exception handling are an improvement over the C++ version on which they are based.

First, a C++ program can throw any type defined in the program or by the system. In Java, only objects that are instances of `Throwable` or some class that descends from it can be thrown. This separates the objects that can be thrown from all of the other objects (and nonobjects) that inhabit a program. What significance can be attached to an exception that causes an **`int`** value to be thrown?

Second, a C++ program unit that does not include a **`throw`** clause can throw any exception, which tells the reader nothing. A Java method that does not include a **`throws`** clause cannot throw any checked exception that it does not handle. Therefore, the reader of a Java method knows from its header what exceptions it could throw but does not handle. A C++ compiler ignores **`throw`** clauses, but a Java compiler ensures that all exceptions that a method can throw are listed in its **`throws`** clause.

Third, the **`finally`** clause is a useful addition. It allows cleanup kinds of actions to take place regardless of how a compound statement terminated.

Finally, the Java run-time system implicitly throws a variety of predefined exceptions, such as for array indices out of range and dereferencing null references, which can be handled by any user program. A C++ program can handle only those exceptions that it explicitly throws (or that are thrown by library classes it uses).

C# includes exception-handling constructs that are very much like those of Java, except that C# does not have a **`throws`** clause.

14.4 Exception Handling in Python and Ruby

This section provides brief overviews of the exception-handling mechanisms of Python and Ruby.

14.4.1 Python

In Python, exceptions are objects. The base class of all exception classes is `BaseException`, from which the `Exception` class is derived. `BaseException` provides some services that are useful for all exception classes, but it is not usually directly subclassed. All predefined exception classes are derived from `Exception` and user-defined exception classes also are derived from it. The most commonly used predefined subclasses of `Exception` are `ArithmeticError`, whose primary subclasses are `OverflowError`, `ZeroDivisionError`, and `FloatingPointError`, and `LookupError`, whose main subclasses are `IndexError` and `KeyError`.

The statements for dealing with exceptions are similar to those of Java. The general form of a **`try`** construct is as follows:

```
try:
  The try block (the range of statements to be watched for except
except Exception1:
  Handler for Exception1
except Exception2:
  Handler for Exception2
...
else:
  The else block (what to do when no exception is raised)
finally:
  The finally block (what must be done regardless of what happene
```

Both the **`else`** and the **`finally`** clauses are optional.

One difference between handlers in Java and Python is that Python uses **except** to introduce them, rather than **catch**. The **else** clause is executed if no exception is raised in the **try** block. The **finally** clause has the same semantics as its counterpart in Java: If an exception is raised in the **try** block but is not handled by an immediately following handler, the exception is propagated after the **finally** block is executed. Because a handler handles its named exception, as well as all subclasses of that exception, a handler that names `Exception` handles all predefined and user-defined exceptions.

An unhandled exception is propagated to progressively larger enclosing **try** constructs, searching for an appropriate handler. If none is found, the exception is propagated to the function's caller, again searching for a handler in a nesting **try** construct. If no handler is found at any level, the default handler is called, which produces an error message and a stack trace and terminates the program.

The **raise** statement of Python is similar to the **throw** statement of Java and C++. The parameter for **raise** is the class name of the exception to be raised. For example, we could have the following:

```
raise IndexError
```

This statement implicitly creates an instance of the named class, `IndexError`.

An exception handler can gain access to the object of the raised exception by providing an **as** clause and a variable name, as in the following:

```
except Exception as ex_obj:
```

This is a universal handler, as it handles all exceptions. The exception object can be printed with a **print** statement in the handler, which produces the message of the object. For example, if the exception was `ZeroDivisionError`, the message would be `division by zero`.

Python's **assert** statement provides a mechanism for making some exception handling optional. The general form of **assert** is as follows:

```
assert test, data
```

In this statement, `test` is a Boolean flag or expression and `data` is the value that is sent to the constructor for the exception object to be raised. The meaning of this statement, which optionally raises the `AssertionError` exception, can be described with the following code:

```
if __debug__:
  if not test:
    raise AssertionError(data)
```

`__debug__` is a predefined flag that is set to `True` unless the `-0` flag is used on the command that runs the program. This allows one to disable all **`assert`** statements for a particular run of the program. If an `AssertionError` exception is not handled by the program, like other unhandled exceptions, it terminates the program after using the default handler.

Python does not have an equivalent to the **`throws`** clause of Java.

14.4.2 Ruby

Like Python, Ruby exceptions are objects and it has a large collection of predefined exception classes. All of the exceptions that are handled by application programs are either objects of the `StandardError` class or a class that descends from it. `StandardError` is derived from `Exception`, which provides two useful methods to all its descendants. These are `message`, which returns the human-readable error message, and `backtrace`, which returns a stack trace starting from the method where the exception was raised. Some of the predefined subclasses of `StandardError` are `ArgumentError`, `IndexError`, `IOError`, and `ZeroDivisionError`.

Exceptions are explicitly raised with the `raise` method. `raise` is often called with a string parameter. In this case, it raises a new `RuntimeError` object with the string as its message. For example, we could have the following:

```
raise "bad parameter"  if  count == 0
```

`raise` could also have two parameters, the first of which would be an object of an exception class. The `exception` method of this object is called and the

returned `Exception` object is raised. In this case, the second parameter would be the string message to be displayed. For example, we could have the following:

```
raise TypeError, "Float parameter expected"
if not param.is_a? Float
```

An exception handler is specified with a **`rescue`** clause, which is attached to a statement. To attach an exception handler to a segment of code, the code is placed in a **`begin-end`** block. The **`rescue`** clause is placed in the block after the code of the block. In general, this appears as in the following:

```
begin
  The sequence of statements in the block
rescue
  The handler
end
```

A **`begin-end`** block can include an **`else`** clause and/or an **`ensure`** clause. The **`else`** clause is exactly like that of Python. The **`ensure`** clause is exactly like a `finally` clause. A method can act as a container for exception handling in place of a **`begin-end`** block.

In a clear departure from most other languages, Ruby allows a segment of code that raised an exception to be rerun after the exception is handled. This is specified with a **`retry`** statement at the end of the handler.

14.5 Introduction to Event Handling

Event handling is similar to exception handling. In both cases, the handlers are implicitly called by the occurrence of something, either an exception or an event. While exceptions can be raised either explicitly by user code or implicitly by hardware or a software interpreter, events are created by external actions, such as user interactions through a graphical user interface (GUI). In this section, the fundamentals of event handling, which are less complex than those of exception handling, are introduced.

In conventional (non-event-driven) programming, the program code itself specifies the order in which that code is executed, although the order is usually affected by the program's input data. In event-driven programming, parts of the program are executed at completely unpredictable times, often triggered by user interactions with the executing program.

The particular kind of event handling discussed in this chapter is related to GUIs. Therefore, most of the events are caused by user interactions through graphical objects or components, often called *widgets*. The most common widgets are buttons. Implementing reactions to user interactions with GUI components is the most common form of event handling.

An **event** is a notification that something specific has occurred, such as a mouse click on a graphical button. Strictly speaking, an event is an object that is implicitly created by the run-time system in response to a user action, at least in the context in which event handling is being discussed here.

An **event handler** is a segment of code that is executed in response to the appearance of an event. Event handlers enable a program to be responsive to user actions.

Although event-driven programming was being used long before GUIs appeared, it has become a widely used programming methodology only in response to the popularity of these interfaces. As an example, consider the GUIs presented to users of Web browsers. Many Web documents presented

to browser users are now dynamic. Such a document may present an order form to the user, who chooses the merchandise by clicking buttons. The required internal computations associated with these button clicks are performed by event handlers that react to the click events.

Another common use of event handlers is to check for simple errors and omissions in the elements of a form, either when they are changed or when the form is submitted to the Web server for processing. Using event handling on the browser to check the validity of form data saves the time of sending that data to the server, where their correctness then must be checked by a server-resident program or script before they can be processed. This kind of event-driven programming is often done using a client-side scripting language, such as JavaScript.

14.6 Event Handling with Java

In addition to Web applications, non-Web Java applications can present GUIs to users. GUIs in Java applications are discussed in this section.

The initial version of Java provided a somewhat primitive form of support for GUI components. In version 1.2 of the language, released in late 1998, a new collection of components was added. These were collectively called Swing.

14.6.1 Java Swing GUI Components[4]

4. Over the last few years, Swing is slowly being replaced by a new GUI toolset named JAVAFX.

The Swing collection of classes and interfaces, defined in `javax.swing`, includes GUI components, or widgets. Because our interest here is event handling, not GUI components, we discuss only two kinds of widgets: text boxes and radio buttons.

A text box is an object of class `JTextField`. The simplest `JTextField` constructor takes a single parameter, the length of the box in characters. For example,

```
JTextField name = new JTextField(32);
```

The `JTextField` constructor can also take a literal string as an optional first parameter. This string parameter, when present, is displayed as the initial contents of the text box.

Radio buttons are special buttons that are placed in a button group container. A button group is an object of class `ButtonGroup`, whose constructor takes no parameters. In a radio button group, only one button can be pressed at a time.

If any button in the group becomes pressed, the previously pressed button is implicitly unpressed. The `JRadioButton` constructor, used for creating radio buttons, takes two parameters: a label and the initial state of the radio button (`true` or `false`, for pressed and not pressed, respectively). If one radio button in a group is initially set to pressed, the others in the group default to unpressed. After the radio buttons are created, they are placed in their button group with the `add` method of the group object. Consider the following example:

```
ButtonGroup payment = new ButtonGroup();
JRadioButton box1 = new JRadioButton("Visa", true);
JRadioButton box2 = new JRadioButton("Master Charge");
JRadioButton box3 = new JRadioButton("Discover");
payment.add(box1);
payment.add(box2);
payment.add(box3);
```

A `JFrame` object is a frame, which is displayed as a separate window. The `JFrame` class defines the data and methods that are needed for frames. So, a class that uses a frame can be a subclass of `JFrame`. A `JFrame` has several layers called **panes**. We are interested in just one of those layers, the content pane. Components of a GUI are placed in a `JPanel` object (a panel), which is used to organize and define the layout of the components. A frame is created and the panel containing the components is added to that frame's content pane.

Predefined graphic objects, such as GUI components, are placed directly in a panel. The following creates the panel object used in the following discussion of components:

```
JPanel myPanel = new JPanel();
```

After the components have been created with constructors, they are placed in the panel with the `add` method, as in

```
myPanel.add(button1);
```

14.6.2 The Java Event Model

When a user interacts with a GUI component, for example by clicking a button, the component creates an event object and calls an event handler through an object called an event listener, passing the event object. The event handler provides the associated actions. GUI components are event generators. In Java, events are connected to event handlers through **event listeners**. Event listeners are connected to event generators through event listener registration. Listener registration is done with a method of the class that implements the listener interface, as described later in this section. Only event listeners that are registered for a specific event are notified when that event occurs.

The listener method that receives the message implements an event handler. To make the event-handling methods conform to a standard protocol, an interface is used. An interface prescribes standard method protocols but does not provide implementations of those methods.

A class that needs to implement an event handler must implement an interface for the listener for that handler. There are several classes of events and listener interfaces. One class of events is `ItemEvent`, which is associated with the event of clicking a checkbox or a radio button, or selecting a list item. The `ItemListener` interface includes the protocol of a method, `itemStateChanged`, which is the handler for `ItemEvent` events. So, to provide an action that is triggered by a radio button click, the interface `ItemListener` must be implemented, which requires a definition of the method, `itemStateChanged`.

As stated previously, the connection of a component to an event listener is made with a method of the class that implements the listener interface. For example, because `ItemEvent` is the class name of event objects created by user actions on radio buttons, the `addItemListener` method is used to register a listener for radio buttons. The listener for button events created in a panel could be implemented in the panel or a subclass of `JPanel`. So, for a radio button named `button1` in a panel named `myPanel` that implements the `ItemEvent` event handler for buttons, we would register the listener with the following statement:

```
button1.addItemListener(this);
```

Each event handler method receives an event parameter, which provides information about the event. Event classes have methods to access that information. For example, when called through a radio button, the `isSelected` method returns **`true`** or **`false`**, depending on whether the button was on or off (pressed or not pressed), respectively.

All the event-related classes are in the `java.awt.event` package, so it is imported to any class that uses events.

The following is an example application, `RadioB`, that illustrates the use of events and event handling. This application constructs radio buttons that control the font style of the contents of a text field. It creates a `Font` object for each of four font styles. Each of these has a radio button to enable the user to select the font style.

The purpose of this example is to show how events raised by GUI components can be handled to change the output display of the program dynamically. Because of our narrow focus on event handling, some parts of this program are not explained here.

```
/* RadioB.java
   An example to illustrate event handling with interactive
   radio buttons that control the font style of a textfield
  */
package radiob;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class RadioB extends JPanel implements
        ItemListener {
    private JTextField text;
    private Font plainFont, boldFont, italicFont,
                 boldItalicFont;
    private JRadioButton plain, bold, italic, boldItalic;
    private ButtonGroup radioButtons;
// The constructor method is where the display is initially
//  built
    public  RadioB() {
// Create the test text string and set its font
        text = new JTextField(
             "In what font style should I appear?", 25);
        text.setFont(plainFont);
// Create radio buttons for the fonts and add them to
```

```
//  a new button group
        plain = new JRadioButton("Plain", true);
        bold = new JRadioButton("Bold");
        italic = new JRadioButton("Italic");
        boldItalic = new JRadioButton("Bold Italic");
        radioButtons = new ButtonGroup();
        radioButtons.add(plain);
        radioButtons.add(bold);
        radioButtons.add(italic);
        radioButtons.add(boldItalic);
        // Create a panel and put the text and the radio
        //  buttons in it; then add the panel to the frame
        JPanel radioPanel = new JPanel();
        radioPanel.add(text);
        radioPanel.add(plain);
        radioPanel.add(bold);
        radioPanel.add(italic);
        radioPanel.add(boldItalic);
        add(radioPanel, BorderLayout.LINE_START);
// Register the event handlers
        plain.addItemListener(this);
        bold.addItemListener(this);
        italic.addItemListener(this);
        boldItalic.addItemListener(this);
// Create the fonts
        plainFont = new Font("Serif", Font.PLAIN, 16);
        boldFont = new Font("Serif", Font.BOLD, 16);
        italicFont = new Font("Serif", Font.ITALIC, 16);
        boldItalicFont = new  Font("Serif", Font.BOLD +
                                   Font.ITALIC, 16);
    }  // End of the constructor for RadioB
// The event handler
    public void itemStateChanged (ItemEvent e) {
// Determine which button is on and set the font
//  accordingly
       if (plain.isSelected())
          text.setFont(plainFont);
       else if (bold.isSelected())
          text.setFont(boldFont);
       else if (italic.isSelected())
          text.setFont(italicFont);
       else if  (boldItalic.isSelected())
          text.setFont(boldItalicFont);
    } // End of itemStateChanged
// The main method
     public static void main(String[] args) {
// Create the window frame
        JFrame myFrame = new JFrame(" Radio button
```

```
                                          example");
// Create the content pane and set it to the frame
        JComponent myContentPane = new RadioB();
        myContentPane.setOpaque(true);
        myFrame.setContentPane(myContentPane);
// Display the window.
        myFrame.pack();
        myFrame.setVisible(true);
    }
} // End of RadioB
```

The `RadioB.java` application produces the screen shown in Figure 14.2.

Figure 14.2 Output of `RadioB.java`

Source: Java radio applet screenshot.

14.7 Event Handling in C#

Event handling in C# (and in the other .NET languages) is similar to that of Java. .NET provides two approaches to creating GUIs in applications, the original Windows Forms and the more recent Windows Presentation Foundation. The latter is the more sophisticated and complex of the two. Because our interest is just in event handling, we will use the simpler Windows Forms to discuss our subject.

Using Windows Forms, a C# application that constructs a GUI is created by subclassing the `Form` predefined class, which is defined in the `System.Windows.Forms` namespace. This class implicitly provides a window to contain our components. There is no need to build frames or panels explicitly.

Text can be placed in a `Label` object and radio buttons are objects of the `RadioButton` class. The size of a `Label` object is not explicitly specified in the constructor; rather it can be specified by setting the `AutoSize` data member of the `Label` object to **`true`**, which sets the size according to what is placed in it.

Components can be placed at a particular location in the window by assigning a new `Point` object to the `Location` property of the component. The `Point` class is defined in the `System.Drawing` namespace. The `Point` constructor takes two parameters, which are the coordinates of the object in pixels. For example, `Point(100, 200)` is a position that is `100` pixels from the left edge of the window and `200` pixels from the top. The label of a component is set by assigning a string literal to the `Text` property of the component. After creating a component, it is added to the form window by sending it to the `Add` method of the `Controls` subclass of the form. Therefore, the following code creates a radio button with the label `Plain` at the `(100, 300)` position in the output window:

```
private RadioButton plain = new RadioButton();
plain.Location = new Point(100, 300);
plain.Text = "Plain";
```

```
Controls.Add(plain);
```

All C# event handlers have the same protocol: the return type is **`void`** and the two parameters are of types **`object`** and `EventArgs`. Neither of the parameters needs to be used for a simple situation. An event handler method can have any name. A radio button is tested to determine whether it is clicked with the Boolean `Checked` property of the button. Consider the following skeletal example of an event handler:

```
private void rb_CheckedChanged (object o, EventArgs e){
  if (plain.Checked) . . .
  . . .
}
```

To register an event, a new `EventHandler` object must be created. The constructor for this class is sent the name of the handler method. The new object is added to the predefined delegate for the event on the component object (using the `+=` assignment operator). For example, when a radio button changes from unchecked to checked, the `CheckedChanged` event is raised and the handlers registered on the associated delegate, which is referenced by the name of the event, are called. If the event handler is named `rb_CheckedChanged`, the following statement would register the handler for the `CheckedChanged` event on the radio button `plain`:

```
plain. CheckedChanged +=
       new EventHandler(rb_CheckedChanged);
```

Following is the `RadioB` example from Section 14.6 rewritten in C#. Once again, because our focus is on event handling, we do not explain all of the details of the program.

```
// RadioB.cs
// An example to illustrate event handling with
//   interactive radio buttons that control the font
//  style of a string of text
namespace RadioB {
    using System;
    using System.Drawing;
    using System.Windows.Forms;
  public class RadioB : Form {
    private Label text = new Label();
    private RadioButton plain = new RadioButton();
```

```
private RadioButton bold = new RadioButton();
private RadioButton italic = new RadioButton();
private RadioButton boldItalic = new RadioButton();
// Constructor for RadioB
public RadioB() {
  // Initialize the attributes of the text and radio
  //  buttons
  text.AutoSize = true;
  text.Text = "In what font style should I appear?";
  plain.Location = new Point(220,0);
  plain.Text = "Plain";
  plain.Checked = true;
  bold.Location = new Point(350, 0);
  bold.Text = "Bold";
  italic.Location = new Point(480, 0);
  italic.Text = "Italics";
  boldItalic.Location = new Point(610, 0);
  boldItalic.Text = "Bold/Italics";
  // Add the text and the radio buttons to the form
  Controls.Add(text);
  Controls.Add(plain);
  Controls.Add(bold);
  Controls.Add(italic);
  Controls.Add(boldItalic);
  // Register the event handler for the radio buttons
  plain.CheckedChanged +=
       new EventHandler(rb_CheckedChanged);
  bold.CheckedChanged +=
       new EventHandler(rb_CheckedChanged);
  italic.CheckedChanged +=
       new EventHandler(rb_CheckedChanged);
  boldItalic.CheckedChanged +=
       new EventHandler(rb_CheckedChanged);
}
// The main method is where execution begins
static void Main() {
  Application.EnableVisualStyles();
  Application.SetCompatibleTextRenderingDefault (false);
  Application.Run(new RadioB());
}
// The event handler
private void rb_CheckedChanged (object o,
                                EventArgs e) {
// Determine which button is on and set the font
//  accordingly
 if (plain.Checked)
     text.Font =
          new Font( text.Font.Name, text.Font.Size,
```

```
                        FontStyle.Regular);
        if (bold.Checked)
            text.Font =
                new Font(text.Font.Name, text.Font.Size,
                        FontStyle.Bold);
        if (italic.Checked)
            text.Font =
                new Font(text.Font.Name, text.Font.Size,
                        FontStyle.Italic);
        if (boldItalic.Checked)
            text.Font =
                new Font(text.Font.Name, text.Font.Size,
                        FontStyle.Italic ^ FontStyle.Bold);
      } // End of radioButton_CheckedChanged
    } // End of RadioB
}
```

The output from this program is exactly like that shown in Figure 14.2.

SUMMARY

C++ includes no predefined exceptions (except those defined in the standard library). C++ exceptions are objects of a primitive type, a predefined class, or a user-defined class. Exceptions are bound to handlers by connecting the type of the expression in the **`throw`** statement to that of the formal parameter of the handler. Handlers all have the same name—**`catch`**. The C++ **`throw`** clause of a method lists the types of exceptions that the method could throw.

Java exceptions are objects whose ancestry must trace back to a class that descends from the `Throwable` class. There are two categories of exceptions—checked and unchecked. Checked exceptions are a concern for the user program and the compiler. Unchecked exceptions can occur anywhere and are often ignored by user programs.

The Java **`throws`** clause of a method lists the checked exceptions that it could throw and does not handle. It must include exceptions that methods it calls could raise and propagate back to its caller.

The Java **`finally`** clause provides a mechanism for guaranteeing that some code will be executed regardless of how the execution of a **`try`** compound terminates.

Java now includes an **`assert`** statement, which facilitates defensive programming.

Python's exception handling is similar to that of Java, although it adds the **`else`** clause to the **`try`** construct. Also, it uses **`except`** clauses rather than **`catch`** clauses to define handlers and **`raise`** instead of **`throw`**. Access to the data of an exception object is gained by assigning the object to a variable with an **`as`** clause. Python's **`assert`** statement is a conditional **`raise`**.

Exception handling in Ruby is similar to that of Python. Every exception class has two methods, `message` and `backtrace`. Exceptions are often raised with a **`raise`** statement with a single string parameter. This creates a new

`RuntimeError` object with the string as its message. The **`raise`** statement can be made conditional by adding a conditional expression to it. The scope of exception handlers usually is specified with a **`begin-end`** block. Handlers are defined in **`rescue`** clauses. A **`begin-end`** block can include an **`else`** clause and an **`ensure`** clause, which is like the **`finally`** clause of Python and Java.

An event is a notification that something has occurred that requires special processing. Events are often created by user interactions with a program through a graphical user interface. Java event handlers are called through event listeners. An event listener must be registered for an event if it is to be notified when the event occurs. Two of the most commonly used event listeners interfaces are `actionPerformed` and `itemStateChanged`.

Windows Forms is the original approach to building GUI components and handling events in .NET languages. A C# application builds a GUI in this approach by subclassing the `Form` class. All .NET event handlers use the same protocol. Event handlers are registered by creating an `EventHandler` object and assigning it to the predefined delegate associated with the GUI object that can raise the event.

BIBLIOGRAPHIC NOTES

One of the most important papers on exception handling that is not connected with a particular programming language is the work by Goodenough (1975). The problems with the PL/I design for exception handling are covered in MacLaren (1977). Exception handling in C++ is described by Stroustrup (1997). Exception handling in Java is described by Campione et al. (2001).

REVIEW QUESTIONS

1. Define *exception*, *exception handler*, *raising an exception*, *continuation*, *finalization*, and *built-in exception*.

2. What are the two alternatives for designing continuation?

3. What are the advantages of having support for exception handling built in to a language?

4. What are the design issues for exception handling?

5. What does it mean for an exception to be bound to an exception handler?

6. What is the name of all C++ exception handlers?

7. How can exceptions be explicitly raised in C++?

8. How are exceptions bound to handlers in C++?

9. How can an exception handler be written in C++ so that it handles any exception?

10. Where does execution control go when a C++ exception handler has completed its execution?

11. Does C++ include built-in exceptions?

12. Why is the raising of an exception in C++ not called raise?

13. What is the root class of all Java exception classes?

14. What is the parent class of most Java user-defined exception classes?

15. How can an exception handler be written in Java so that it handles any

exception?

16. What are the differences between a C++ **`throw`** specification and a Java **`throws`** clause?

17. What is the difference between checked and unchecked exceptions in Java?

18. How can an exception handler be written in Java so that it handles any exception?

19. What is the purpose of the Java **`finally`** clause?

20. What advantage do language-defined assertions have over simple **`if-then`** constructs?

21. Explain what an **`else`** block in Python does.

22. What is the purpose of an **`as`** clause in Python?

23. Explain what an **`assert`** statement in Python does.

24. What does the `message` method of Ruby's `StandardError` class do?

25. What exactly happens when a **`raise`** statement with a string parameter is executed?

26. What exactly does a Ruby **`ensure`** clause do?

27. In what ways are exception handling and event handling related?

28. Define event and event handler.

29. What is event-driven programming?

30. What is the purpose of a Java `JFrame`?

31. What is the purpose of a Java `JPanel`?

32. What object is often used as the event listener in Java GUI applications?

33. What is the origin of the protocol for an event handler in Java?

34. What method is used to register an event handler in Java?

35. Using .NET's Windows Forms, what namespace is required to build a GUI for a C# application?

36. How is a component positioned in a form using Windows Forms?

37. What is the protocol of a .NET event handler?

38. What class of object must be created to register a .NET event handler?

39. What role do delegates play in the process of registering event handlers?

PROBLEM SET

1. What did the designers of C get in return for not requiring subscript range checking?

2. Describe three approaches to exception handling in languages that do not provide direct support for it.

3. From textbooks on the PL/I and Ada programming languages, look up the respective sets of built-in exceptions. Do a comparative evaluation of the two, considering both completeness and flexibility.

4. From a textbook on COBOL, determine how exception handling is done in COBOL programs.

5. In languages without exception-handling facilities, it is common to have most subprograms include an "error" parameter, which can be set to some value representing "OK" or some other value representing "error in procedure." What advantage does a linguistic exception-handling facility like that of Java have over this method?

6. In a language without exception-handling facilities, we could send an error-handling procedure as a parameter to each procedure that can detect errors that must be handled. What disadvantages are there to this method?

7. Compare the methods suggested in Problems 5 and 6. Which do you think is better and why?

8. Write a comparative analysis of the **`throw`** clause of C++ and the **`throws`** clause of Java.

9. Consider the following C++ skeletal program:

```
class Big {
  int i;
```

```
  float f;
  void fun1() throw int {
      . . .
      try {
        . . .
        throw i;
      . . .
      throw f;
      . . .
    }
    catch(float) { . . . }
  . . .
  }
}
class Small {
  int j;
  float g;
  void fun2() throw float {
    . . .
    try {
      . . .
      try {
        Big.fun1();
        . . .
        throw j;
        . . .
        throw g;
        . . .
     }
     catch(int) { . . . }
     . . .
    }
    catch(float) { . . . }
  }
}
```

In each of the four **`throw`** statements, where is the exception handled? Note that `fun1` is called from `fun2` in class `Small`.

10. Write a detailed comparison of the exception-handling capabilities of C++ and those of Java.

11. With the help of a book on ML, write a detailed comparison of the exception-handling capabilities of ML and those of Java.

12. Summarize the arguments in favor of the termination and resumption models of continuation.

PROGRAMMING EXERCISES

1. Suppose you are writing a C++ function that has three alternative approaches for accomplishing its requirements. Write a skeletal version of this function so that if the first alternative raises any exception, the second is tried, and if the second alternative raises any exception, the third is executed. Write the code as if the three methods were procedures named `alt1`, `alt2`, and `alt3`.

2. Write a Java program that inputs a list of integer values in the range of −100 to 100 from the keyboard and computes the sum of the squares of the input values. This program must use exception handling to ensure that the input values are in range and are legal integers, to handle the error of the sum of the squares becoming larger than a standard `Integer` variable can store, and to detect end-of-file and use it to cause the output of the result. In the case of overflow of the sum, an error message must be printed and the program terminated.

3. Write a C++ program for the specification of Programming Exercise 2.

4. Revise the Java program of Section 14.3.5 to use `EOFException` to detect the end of the input.

5. Rewrite the Java code of Section 14.3.6 that uses a **`finally`** clause in C++.

15 Functional Programming Languages

This chapter introduces functional programming and some of the programming languages that have been designed for this approach to software development. We begin by reviewing the fundamental ideas of mathematical functions, because functional languages are based on them. Next, the idea of a functional programming language is introduced, followed by a look at the first functional language, Lisp. The next, somewhat lengthy section, is devoted to an introduction to Scheme, including some of its primitive functions, special forms, functional forms, and some examples of simple functions written in Scheme. Next, we provide brief introductions to

Common Lisp, ML, Haskell, and F#. Then, we discuss support for functional programming that is included in some imperative languages. The following section describes some of the applications of functional programming languages. Finally, we present a brief comparison of functional and imperative languages.

15.1 Introduction

Most of the earlier chapters of this book have been concerned primarily with the imperative programming languages. The high degree of similarity among the imperative languages arises in part from one of the common bases of their design: the von Neumann architecture, as discussed in Chapter 1. Imperative languages can be thought of collectively as a progression of developments to improve the basic model, which was Fortran I. All have been designed to make efficient use of von Neumann architecture computers. Although the imperative style of programming has been found acceptable by most programmers, its heavy reliance on the underlying architecture is thought by some to be an unnecessary restriction on the alternative approaches to software development.

Other bases for language design exist, some of them oriented more to particular programming paradigms or methodologies than to efficient execution on a particular computer architecture. Thus far, however, only a relatively small minority of programs have been written in nonimperative languages.

The functional programming paradigm, which is based on mathematical functions, is the design basis of the most important nonimperative styles of languages. This style of programming is supported by functional programming languages.

The 1977 ACM Turing Award was given to John Backus for his work in the development of Fortran. Each recipient of this award presents a lecture when the award is formally given, and the lecture is subsequently published in the *Communications of the ACM*. In his Turing Award lecture, Backus (1978) made a case that purely functional programming languages are better than imperative languages because they result in programs that are more readable, more reliable, and more likely to be correct. The crux of his argument was that purely functional programs are easier to understand, both during and after development, largely because the meanings of expressions are independent of their context (one characterizing feature of a pure functional

programming language is that neither expressions nor functions have side effects).

In this lecture, Backus proposed a pure functional language, FP (*f*unctional *p*rogramming), which he used to frame his argument. Although the language did not succeed, at least in terms of achieving widespread use, his idea motivated debate and research on pure functional programming languages. The point here is that some well-known computer scientists have attempted to promote the concept that functional programming languages are superior to the traditional imperative languages, though those efforts have obviously fallen short of their goals. However, over the last decade, prompted in part by the maturing of the typed functional languages, such as ML, Haskell, OCaml, and F#, there has been an increase in the interest in and use of functional programming languages.

One of the fundamental characteristics of programs written in imperative languages is that they have state, which changes throughout the execution process. This state is represented by the program's variables. The author and all readers of the program must understand the uses of its variables and how the program's state changes through execution. For a large program, this is a daunting task. This is one problem with programs written in an imperative language that is not present in a program written in a pure functional language, for such programs have neither variables nor state.

Lisp began as a pure functional language but soon acquired some important imperative features that increased its execution efficiency. It is still the most important of the functional languages, at least in the sense that it is the only one that has achieved widespread use. It dominates in the areas of knowledge representation, machine learning, intelligent training systems, and the modeling of speech. Common Lisp is an amalgam of several early 1980s dialects of Lisp.

Scheme is a small, static-scoped dialect of Lisp. Scheme has been widely used to teach functional programming. It is also used in some universities to teach introductory programming courses.

The development of the typed functional programming languages, primarily ML, Haskell, OCaml, and F#, has led to a significant expansion of the areas

of computing in which functional languages are now used. As these languages have matured, their practical use is growing. They are now being used in areas such as database processing, financial modeling, statistical analysis, and bioinformatics.

One objective of this chapter is to provide an introduction to functional programming using the core of Scheme, intentionally leaving out its imperative features. Sufficient material on Scheme is included to allow the reader to write some simple but interesting programs. It is difficult to acquire an actual feel for functional programming without some actual programming experience, so that is strongly encouraged.

15.2 Mathematical Functions

A mathematical function is a mapping of members of one set, called the **domain set**, to another set, called the **range set**. A function definition specifies the domain and range sets, either explicitly or implicitly, along with the mapping. The mapping is described by an expression or, in some cases, by a table. Functions are often applied to a particular element of the domain set, given as a parameter to the function. Note that the domain set may be the cross product of several sets (reflecting that there can be more than one parameter). A function yields an element of the range set.

One of the fundamental characteristics of mathematical functions is that the evaluation order of their mapping expressions is controlled by recursion and conditional expressions, rather than by the sequencing and iterative repetition that are common to programs written in the imperative programming languages.

Another important characteristic of mathematical functions is that because they have no side effects and cannot depend on any external values, they always map a particular element of the domain to the same element of the range. However, a subprogram in an imperative language may depend on the current values of several nonlocal or global variables. This makes it difficult to determine statically what values the subprogram will produce and what side effects it will have on a particular execution.

In mathematics, there is no such thing as a variable that models a memory location. Local variables in functions in imperative programming languages maintain the state of the function. Computation is accomplished by evaluating expressions in assignment statements that change the state of the program. In mathematics, there is no concept of the state of a function.

A mathematical function maps its parameter(s) to a value (or values), rather than specifying a sequence of operations on values in memory to produce a value.

15.2.1 Simple Functions

Function definitions are often written as a function name, followed by a list of parameters in parentheses, followed by the mapping expression. For example,

cube(x)≡x * x * x, where x is a real number.

In this definition, the domain and range sets are the real numbers. The symbol ≡ is used to mean “is defined as.” The parameter x can represent any member of the domain set, but it is fixed to represent one specific element during evaluation of the function expression. This is one way the parameters of mathematical functions differ from the variables in imperative languages.

Function applications are specified by pairing the function name with a particular element of the domain set. The range element is obtained by evaluating the function-mapping expression with the domain element substituted for the occurrences of the parameter. Once again, it is important to note that during evaluation, the mapping of a function contains no unbound parameters, where a bound parameter is a name for a particular value. Every occurrence of a parameter is bound to a value from the domain set and is a constant during evaluation. For example, consider the following evaluation of cube(x):

cube (2.0)=2.0 * 2.0 * 2.0=8

The parameter x is bound to 2.0 during the evaluation and there are no unbound parameters. Furthermore, x is a constant (its value cannot be changed) during the evaluation.

Early theoretical work on functions separated the task of defining a function from that of naming the function. Lambda notation, as devised by Alonzo Church (1941), provides a method for defining nameless functions. A **lambda expression** specifies the parameters and the mapping of a function. The lambda expression is the function itself, which is nameless. For example, consider the following lambda expression:

λ(x)x * x * x

Church defined a formal computation model (a formal system for function definition, function application, and recursion) using lambda expressions. This is called **lambda calculus**. Lambda calculus can be either typed or untyped. Untyped lambda calculus serves as the inspiration for the functional programming languages.

As stated earlier, before evaluation a parameter represents any member of the domain set, but during evaluation it is bound to a particular member. When a lambda expression is evaluated for a given parameter, the expression is said to be applied to that parameter. The mechanics of such an application are the same as for any function evaluation. Application of the example lambda expression is denoted as in the following example:

(λ(x)x * x * x)(2)

which results in the value 8.

Lambda expressions, like other function definitions, can have more than one parameter.

15.2.2 Functional Forms

A **higher-order function**, or **functional form**, is one that either takes one or more functions as parameters or yields a function as its result, or both. One common kind of functional form is **function composition**, which has two functional parameters and yields a function whose value is the first actual parameter function applied to the result of the second. Function composition is written as an expression, using º as an operator, as in

h≡f ° g

For example, if

f(x)≡x+2 g(x)≡3*x

then *h* is defined as

h(x)≡f(g(x)), or h(x)≡(3*x)+2

Apply-to-all is a functional form that takes a single function as a parameter.[1] If applied to a list of parameters, apply-to-all applies its functional parameter to each of the values in the list parameter and collects the results in a list or sequence. Apply-to-all is denoted by α. Consider the following example:

1. In programming languages, these are often called *map* functions.

Let

h(x)≡x*x

then

α(h,(2,3,4)) yields `(4, 9, 16)`

There are other functional forms, but these two examples illustrate the basic characteristics of all of them.

15.3 Fundamentals of Functional Programming Languages

The objective of the design of a functional programming language is to mimic mathematical functions to the greatest extent possible. This results in an approach to problem solving that is fundamentally different from approaches used with imperative languages. In an imperative language, an expression is evaluated and the result is stored in a memory location, which is represented as a variable in a program. This is the purpose of assignment statements. This necessary attention to memory cells, whose values represent the state of the program, results in a relatively low-level programming methodology.

A program in an assembly language often must also store the results of partial evaluations of expressions. For example, to evaluate

$(x+y)/(a-b)$

the value of $(x+y)$ is computed first. That value must then be stored while $(a-b)$ is evaluated. The compiler handles the storage of intermediate results of expression evaluations in high-level languages. The storage of intermediate results is still required, but the details are hidden from the programmer.

A purely functional programming language does not use variables or assignment statements, thus freeing the programmer from concerns related to the memory cells, or state, of the program. Without variables, iterative constructs are not possible, for they are controlled by variables. Repetition must be specified with recursion rather than with iteration. Programs are function definitions and function application specifications, and executions consist of evaluating function applications. Without variables, the execution of a purely functional program has no state in the sense of operational and denotational semantics. The execution of a function always produces the same result when given the same parameters. This characteristic is called **referential transparency**. It makes the semantics of purely functional

languages far simpler than the semantics of the imperative languages (and the functional languages that include imperative features). It also makes testing easier, because each function can be tested separately, without any concern for its context.

A functional language provides a set of primitive functions, a set of functional forms to construct complex functions from those primitive functions, a function application operation, and some structure or structures for representing data. These structures are used to represent the parameters and values computed by functions. If a functional language is well designed, it requires only a relatively small number of primitive functions.

As we have seen in earlier chapters, the first functional programming language, Lisp, uses a syntactic form for both data and code that is very different from that of the imperative languages. However, many functional languages designed later use syntax for their code that is similar to that of the imperative languages.

Although there are a few purely functional languages, for example, Haskell, most of the languages that are called functional include some imperative features, for example, mutable variables and constructs that act as assignment statements.

Some concepts and constructs that originated in functional languages, such as lazy evaluation and anonymous subprograms, have now found their way into some languages that are considered imperative.

Although early functional languages were often implemented with interpreters, many programs written in functional programming languages are now compiled.

15.4 The First Functional Programming Language: Lisp

Many functional programming languages have been developed. The oldest and most widely used is Lisp (or one of its descendants), which was developed by John McCarthy at MIT in 1959. Studying functional languages through Lisp is somewhat akin to studying the imperative languages through Fortran: Lisp was the first functional language, but although it has steadily evolved for half a century, it no longer represents the latest design concepts for functional languages. In addition, with the exception of the first version, all Lisp dialects include imperative-language features, such as imperative-style variables, assignment statements, and iteration. (Imperative-style variables are used to name memory cells, whose values can change many times during program execution.) Despite this and their somewhat odd form, the descendants of the original Lisp represent well the fundamental concepts of functional programming and are therefore worthy of study.

15.4.1 Data Types and Structures

There were only two categories of data objects in the original Lisp: atoms and lists. List elements are pairs, where the first part is the data of the element, which is a pointer to either an atom or a nested list. The second part of a pair can be a pointer to an atom, a pointer to another element, or a special value, nil. Elements are linked together in lists with the second parts. Atoms and lists are not types in the sense that imperative languages have types. In fact, the original Lisp was a typeless language. Atoms are either symbols, in the form of identifiers, or numeric literals.

Recall from Chapter 2, that Lisp originally used lists as its data structure because they were thought to be an essential part of list processing. As it eventually developed, however, Lisp rarely requires the general list operations of insertion and deletion at positions other than the beginning of a

list.

Lists are specified in Lisp by delimiting their elements with parentheses. The elements of **simple lists** are restricted to atoms, as in

```
(A B C D)
```

Nested list structures are also specified by parentheses. For example, the list

```
(A (B C) D (E (F G)))
```

is a list of four elements. The first is the atom `A`; the second is the sublist `(B C)`; the third is the atom `D`; the fourth is the sublist `(E (F G))`, which has as its second element the sublist `(F G)`.

In a Lisp implementation, a list is usually stored as linked list structure in which each node has two pointers, one to reference the data of the node and the other to form the linked list. A list is referenced by a pointer to its first element.

The internal representations of our two example lists are shown in Figure 15.1. Note that the elements of a list are shown horizontally. The last element of a list has no successor, so its link is nil. Sublists are shown with the same structure.

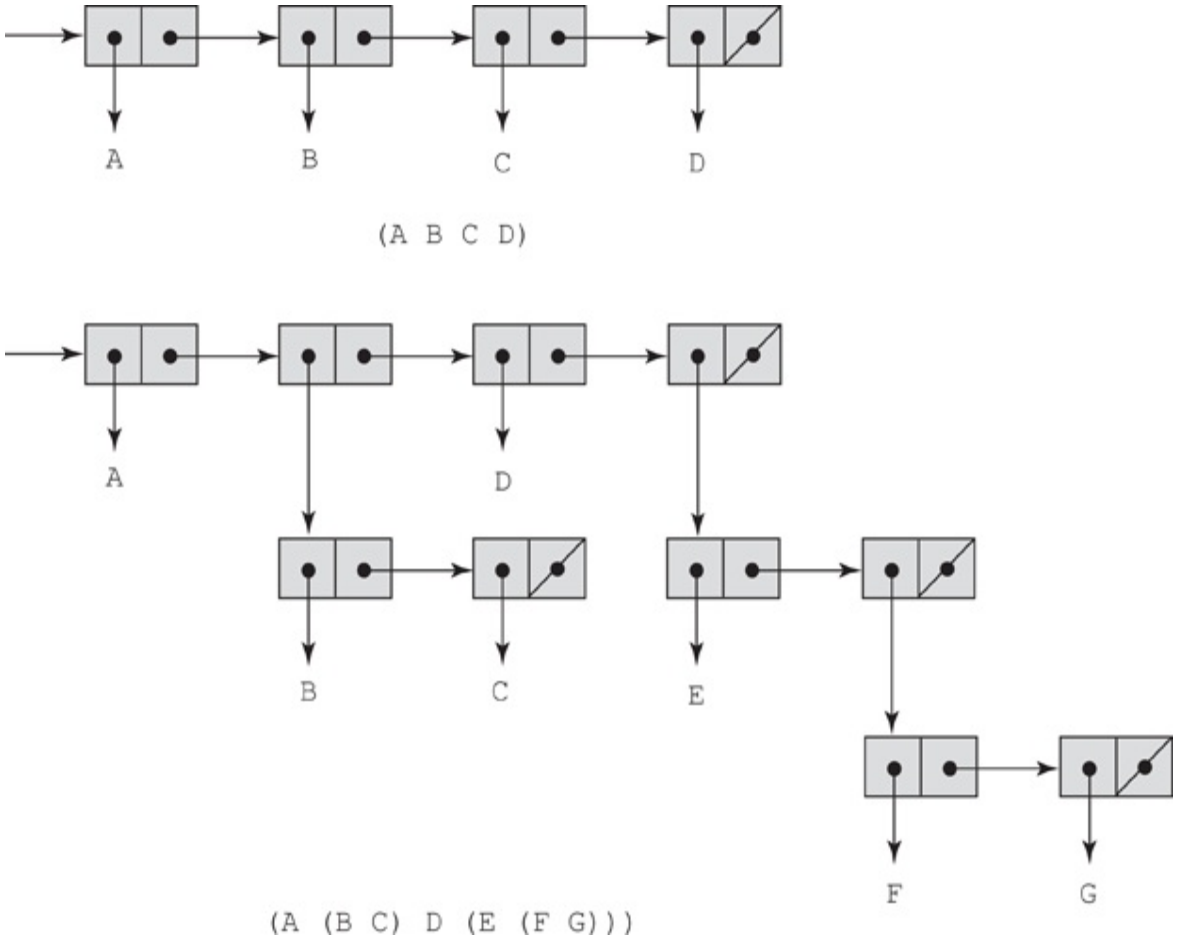


Figure 15.1 Internal representation of two Lisp lists

Figure 15.1 Full Alternative Text

15.4.2 The First Lisp Interpreter

The original intent of Lisp's design was to have a notation for programs that would be as close to Fortran's as possible, with additions when necessary. This notation was called M-notation, for meta-notation. There was to be a compiler that would translate programs written in M-notation into semantically equivalent machine code programs for the IBM 704.

Early in the development of Lisp, McCarthy wrote a paper to promote list processing as an approach to general symbolic processing. McCarthy believed that list processing could be used to study computability, that at the time was usually studied using Turing machines, which are based on the imperative model of computation. McCarthy thought that the functional processing of symbolic lists was a more natural model of computation than Turing machines, that operated on symbols written on tapes, which represented state. One of the common requirements of the study of computation is that one must be able to prove certain computability characteristics of the whole class of whatever model of computation is being used. In the case of the Turing machine model, one can construct a universal Turing machine that can mimic the operations of any other Turing machine. From this concept came the idea of constructing a universal Lisp function that could evaluate any other function in Lisp.

The first requirement for the universal Lisp function was a notation that allowed functions to be expressed in the same way data was expressed. The parenthesized list notation described in Section 15.4.1 had already been adopted for Lisp data, so it was decided to invent conventions for function definitions and function calls that could also be expressed in list notation. Function calls were specified in a prefix list form originally called **Cambridge Polish**,2 as in the following:

2. This name first was used in the early development of Lisp. The name was chosen because Lisp lists resemble the prefix notation used by the Polish logician Jan Lukasiewicz, and because Lisp was born at MIT in Cambridge, Massachusetts. Some now prefer to call the notation *Cambridge prefix*.

(function_name parameter1 . . . cparametern)

For example, if + is a function that takes two or more numeric parameters, the following two expressions evaluate to `12` and `20`, respectively:

```
(+ 5 7)
(+ 3 4 7 6)
```

The lambda notation described in Section 15.2.1 was chosen to specify function definitions. It had to be modified, however, to allow the binding of

functions to names so that functions could be referenced by other functions and by themselves. This name binding was specified by a list consisting of the function name and a list containing the lambda expression, as in

```
(function_name (LAMBDA ( param1...paramn) expression))
```

If you have had no prior exposure to functional programming, it may seem odd to even consider a nameless function. However, nameless functions are sometimes useful in functional programming (as well as in mathematics and imperative programming). For example, consider a function whose action is to produce a function for immediate application to a parameter list. The produced function has no need for a name, for it is applied only at the point of its construction. Such an example is given in Section 15.5.14.

Lisp functions specified in this new notation were called S-expressions, for *s*ymbolic expressions. Eventually, all Lisp structures, both data and code, were called S-expressions. An S-expression can be either a list or an atom. We will usually refer to S-expressions simply as expressions.

McCarthy successfully developed a universal function that could evaluate any other function. This function was named EVAL and was itself in the form of an expression. Two of the people in the AI Project, which was developing Lisp, Stephen B. Russell and Daniel J. Edwards, noticed that an implementation of EVAL could serve as a Lisp interpreter, and they promptly constructed such an implementation (McCarthy et al., 1965).

There were several important results of this quick, easy, and unexpected implementation. First, all early Lisp implementations copied EVAL and were therefore interpretive. Second, the definition of M-notation, which was the planned programming notation for Lisp, was never completed or implemented, so S-expressions became Lisp's only notation. The use of the same notation for data and code has important consequences, one of which will be discussed in Section 15.5.14. Third, much of the original language design was effectively frozen, keeping certain odd features in the language, such as the conditional expression form and the use of () for both the empty list and logical false.

Another feature of early Lisp systems that was apparently accidental was the

use of dynamic scoping. Functions were evaluated in the environments of their callers. No one at the time knew much about scoping, and there may have been little thought given to the choice. Dynamic scoping was used for most dialects of Lisp before 1975. Contemporary dialects either use static scoping or allow the programmer to choose between static and dynamic scoping.

An interpreter for Lisp can be written in Lisp. Such an interpreter, which is not a large program, describes the operational semantics of Lisp, in Lisp. This is vivid evidence of the semantic simplicity of the language.

15.5 An Introduction to Scheme

In this section, we describe the core part of Scheme (Dybvig, 2011). We have chosen Scheme because it is relatively simple, it is popular in colleges and universities, and Scheme interpreters are readily available (and free) for a wide variety of computers. The version of Scheme described in this section is Scheme 4. Note that this section covers only a small part of Scheme, and it includes none of Scheme's imperative features.

15.5.1 Origins of Scheme

The Scheme language, which is a dialect of Lisp, was developed at MIT in the mid-1970s (Sussman and Steele, 1975). It is characterized by its small size, its exclusive use of static scoping, and its treatment of functions as first-class entities. As first-class entities, Scheme functions can be the values of expressions, elements of lists, passed as parameters, and returned from functions. Early versions of Lisp did not provide all of these capabilities.

As an essentially typeless small language with simple syntax and semantics, Scheme is well suited to educational applications, such as courses in functional programming, and also to general introductions to programming.

Most of the Scheme code in the following sections would require only minor modifications to be converted to valid Lisp code.

15.5.2 The Scheme Interpreter

A Scheme interpreter in interactive mode is an infinite read-evaluate-print loop (often abbreviated as REPL). It repeatedly reads an expression typed by the user (in the form of a list), interprets the expression, and displays the resulting value. This form of interpreter is also used by Ruby and Python. Expressions are interpreted by the function EVAL. Literals evaluate to

themselves. So, if you type a number to the interpreter, it simply displays the number. Expressions that are calls to primitive functions are evaluated in the following way: First, each of the parameter expressions is evaluated, in no particular order. Then, the primitive function is applied to the parameter values, and the resulting value is displayed.

Of course, Scheme programs that are stored in files can be loaded and interpreted.

Comments in Scheme are any text following a semicolon on any line.

15.5.3 Primitive Numeric Functions

Scheme includes primitive functions for the basic arithmetic operations. These are `+`, `-`, `*`, and `/`, for add, subtract, multiply, and divide. `*` and `+` can have zero or more parameters. If `*` is given no parameters, it returns `1`; if `+` is given no parameters, it returns `0`. `+` adds all of its parameters together. `*` multiplies all its parameters together. `/` and `-` can have two or more parameters. In the case of subtraction, all but the first parameter are subtracted from the first. Division is similar to subtraction. Some examples are:

Expression	*Value*
`42`	`42`
`(* 3 7)`	`21`
`(+ 5 7 8)`	`20`
`(- 5 6)`	`-1`
`(- 15 7 2)`	`6`
`(- 24 (* 4 3))`	`12`

There are a large number of other numeric functions in Scheme, among them `MODULO`, `ROUND`, `MAX`, `MIN`, `LOG`, `SIN`, and `SQRT`. `SQRT` returns the square root of its numeric parameter, if the parameter's value is not negative. If the parameter is negative, `SQRT` yields a complex number.

In Scheme, note that we use uppercase letters for all reserved words and predefined functions. The official definition of the language specifies that there is no distinction between uppercase and lowercase in these. However, some implementations, for example DrRacket's teaching languages, require lowercase for reserved words and predefined functions.

If a function has a fixed number of parameters, such as `SQRT`, the number of parameters in the call must match that number. If not, the interpreter will produce an error message.

15.5.4 Defining Functions

A Scheme program is a collection of function definitions. Consequently, knowing how to define these functions is a prerequisite to writing the

simplest program. In Scheme, a nameless function actually includes the word LAMBDA, and is called a **lambda expression**. For example,

```
(LAMBDA (x) (* x x))
```

is a nameless function that returns the square of its given numeric parameter. This function can be applied in the same way that named functions are: by placing it in the beginning of a list that contains the actual parameters. For example, the following expression yields 49:

```
((LAMBDA (x) (* x x)) 7)
```

In this expression, x is called a **bound variable** within the lambda expression. During the evaluation of this expression, x is bound to 7. A bound variable never changes in the expression after being bound to an actual parameter value at the time evaluation of the lambda expression begins.

Lambda expressions can have any number of parameters. For example, we could have the following:

```
(LAMBDA (a b c x) (+ (* a x x) (* b x) c))
```

The Scheme special form function DEFINE serves two fundamental needs of Scheme programming: to bind a name to a value and to bind a name to a lambda expression. The form of DEFINE that binds a name to a value may make it appear that DEFINE can be used to create imperative language-style variables. However, these name bindings create named values, not variables.

DEFINE is called a special form because it is interpreted (by EVAL) in a different way than the normal primitives like the arithmetic functions, as we shall soon see.

The simplest form of DEFINE is one used to bind a name to the value of an expression. This form is

(DEFINE symbol expression)

For example,

```
(DEFINE pi 3.14159)
```

```
(DEFINE two_pi (* 2 pi))
```

If these two expressions have been typed to the Scheme interpreter and then `pi` is typed, the number `3.14159` will be displayed; when `two_pi` is typed, `6.28318` will be displayed. In both cases, the displayed numbers may have more digits than are shown here.

This form of `DEFINE` is analogous to a declaration of a named constant in an imperative language. For example, in Java, the equivalents to the above defined names are as follows:

```
final float PI = 3.14159;
final float TWO_PI = 2.0 * PI;
```

Names in Scheme can consist of letters, digits, and special characters except parentheses; they are case insensitive and must not begin with a digit.

The second use of the `DEFINE` function is to bind a lambda expression to a name. In this case, the lambda expression is abbreviated by removing the word `LAMBDA`. To bind a name to a lambda expression, `DEFINE` takes two lists as parameters. The first parameter is the prototype of a function call, with the function name followed by the formal parameters, together in a list. The second list contains an expression to which the name is to be bound. The general form of such a `DEFINE` is3

3. Actually, the general form of `DEFINE` has as its body a list containing a sequence of one or more expressions, although in most cases only one is included. We include only one for simplicity's sake.

```
(DEFINE (function_name parameters)

(expression)

)
```

Of course, this form of `DEFINE` is the definition of a named function.

The following example call to `DEFINE` binds the name `square` to a functional expression that takes one parameter:

```
(DEFINE (square number) (* number number))
```

After the interpreter evaluates this function, it can be used, as in

```
(square 5)
```

which displays `25`.

To illustrate the difference between primitive functions and the `DEFINE` special form, consider the following:

```
(DEFINE x 10)
```

If `DEFINE` were a primitive function, `EVAL`'s first action on this expression would be to evaluate the two parameters of `DEFINE`. If `x` were not already bound to a value, this would be an error. Furthermore, if `x` were already defined, it would also be an error, because this `DEFINE` would attempt to redefine `x`, which is illegal. Remember, `x` is the name of a value; it is not a variable in the imperative sense.

Following is another example of a function. It computes the length of the hypotenuse (the longest side) of a right triangle, given the lengths of the two other sides.

```
(DEFINE (hypotenuse side1 side2)
    (SQRT(+(square side1)(square side2)))
)
```

Notice that `hypotenuse` uses `square`, which was defined previously.

15.5.5 Output Functions

Scheme includes a few simple output functions, but when used with the interactive interpreter, most output from Scheme programs is the normal output from the interpreter, displaying the results of applying `EVAL` to top-level functions.

Note that explicit input and output are not part of the pure functional

programming model, because input operations change the program state and output operations have side effects. Neither of these can be part of a pure functional language. Therefore, this chapter does not describe the explicit input or output functions of Scheme.

15.5.6 Numeric Predicate Functions

A predicate function is one that returns a Boolean value (some representation of either true or false). Scheme includes a collection of predicate functions for numeric data. Among them are the following:

Function	*Meaning*
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
EVEN?	Is it an even number?
ODD?	Is it an odd number?
ZERO?	Is it zero?

Notice that the names for all predefined predicate functions that have words for names end with question marks. In Scheme, the two Boolean values are

#T and #F (or #t and #f), although the Scheme predefined predicate functions return the empty list, (), for false.

When a list is interpreted as a Boolean, any nonempty list evaluates to true; the empty list evaluates to false. This is similar to the interpretation of integers in C as Boolean values; zero evaluates to false and any nonzero value evaluates to true.

In the interest of readability, all of our example predicate functions in this chapter return #F, rather than ().

The NOT function is used to invert the logic of a Boolean expression.

15.5.7 Control Flow

Scheme uses three different constructs for control flow: one similar to the selection construct of the imperative languages and two based on the evaluation control used in mathematical functions.

The Scheme two-way selector function, named IF, has three parameters: a predicate expression, a then expression, and an else expression. A call to IF has the form

(IF predicate then_expression else_expression)

For example,

```
(DEFINE (factorial n)
  (IF (<= n 1)
    1
    (* n (factorial (- n 1)))
))
```

Recall that the multiple selection of Scheme, COND, was discussed in Chapter 8. Following is an example of a simple function that uses COND:

```
(DEFINE (leap? year)
  (COND
    ((ZERO? (MODULO year 400)) #T)
```

```
    ((ZERO? (MODULO year 100)) #F)
    (ELSE (ZERO? (MODULO year 4)))
))
```

The following subsections contain additional examples of the use of COND.

The third Scheme control mechanism is recursion, which is used, as in mathematics, to specify repetition. Most of the example functions in Section 15.5.10 use recursion.

15.5.8 List Functions

One of the more common uses of the Lisp-based programming languages is list processing. This subsection introduces the Scheme functions for dealing with lists. Recall that Scheme's list operations were briefly introduced in Chapter 6. Following is a more detailed discussion of list processing in Scheme.

Scheme programs are interpreted by the function application function, EVAL. When applied to a primitive function, EVAL first evaluates the parameters of the given function. This action is necessary when the actual parameters in a function call are themselves function calls, which is frequently the case. In some calls, however, the parameters are data elements rather than function references. When a parameter is not a function reference, it obviously should not be evaluated. We were not concerned with this earlier, because numeric literals always evaluate to themselves and cannot be mistaken for function names.

Suppose we have a function that has two parameters, an atom and a list, and the purpose of the function is to determine whether the given atom is in the given list. Neither the atom nor the list should be evaluated; they are literal data to be processed. To avoid evaluating a parameter, it is first given as a parameter to the primitive function QUOTE, which simply returns it without change. The following examples illustrate QUOTE:

```
(QUOTE A) returns A
(QUOTE (A B C)) returns (A B C)
```

Calls to QUOTE are usually abbreviated by preceding the expression to be quoted with an apostrophe (') and leaving out the parentheses around the expression. Thus, instead of (QUOTE (A B)), '(A B) is used.

The necessity of QUOTE arises because of the fundamental nature of Scheme (and the other Lisp-based languages): data and code have the same form. Although this may seem odd to imperative language programmers, it results in some interesting and powerful processes, one of which is discussed in Section 15.5.14.

The CAR, CDR, and CONS functions were introduced in Chapter 6. Following are additional examples of the operations of CAR and CDR:

```
(CAR '(A B C)) returns A
(CAR '((A B) C D)) returns (A B)
(CAR 'A) is an error because A is not a list
(CAR '(A)) returns A
(CAR '()) is an error
(CDR '(A B C)) returns (B C)
(CDR '((A B) C D)) returns (C D)
(CDR 'A) is an error
(CDR '(A)) returns ()
(CDR '()) is an error
```

The names of the CAR and CDR functions are peculiar at best. The origin of these names lies in the first implementation of Lisp, which was on an IBM 704 computer. The 704's memory words had two fields, named *decrement* and *address*, that were used in various operand addressing strategies. Each of these fields could store a machine memory address. The 704 also included two machine instructions, also named CAR (*c*ontents of the *a*ddress part of a *r*egister) and CDR (*c*ontents of the *d*ecrement part of a *r*egister), that extracted the associated fields. It was natural to use the two fields to store the two pointers of a list node so that a memory word could neatly store a node. Using these conventions, the CAR and CDR instructions of the 704 provided efficient list selectors. The names carried over into the primitives of all dialects of Lisp.

As another example of a simple function, consider

```
(DEFINE (second a_list) (CAR (CDR a_list)))
```

Once this function is evaluated, it can be used, as in

```
(second '(A B C))
```

which returns B.

Some of the most commonly used functional compositions in Scheme are built in as single functions. For example, (CAAR x) is equivalent to (CAR(CAR x)), (CADR x) is equivalent to (CAR (CDR x)), and (CADDAR x) is equivalent to (CAR (CDR (CDR (CAR x)))). Any combination of A's and D's, up to four, are legal between the 'C' and the 'R' in the function's name. As an example, consider the following evaluation of CADDAR:

```
(CADDAR '((A B (C) D) E)) =
(CAR (CDR (CDR (CAR '((A B (C) D) E))))) =
(CAR (CDR (CDR '(A B (C) D)))) =
(CAR (CDR '(B (C) D))) =
(CAR '((C) D)) =
(C)
```

Following are example calls to CONS:

```
(CONS 'A '()) returns (A)
(CONS 'A '(B C)) returns (A B C)
(CONS '() '(A B)) returns (() A B)
(CONS '(A B) '(C D)) returns ((A B) C D)
```

The results of these CONS operations are shown in Figure 15.2. Note that CONS is, in a sense, the inverse of CAR and CDR. CAR and CDR take a list apart, and CONS constructs a new list from two given list parts. The two parameters to CONS become the CAR and CDR of the new list. Thus, if a_list is a list, then

```
(CONS (CAR a_list) (CDR a_list))
```

returns a list with the same structure and same elements as a_list.

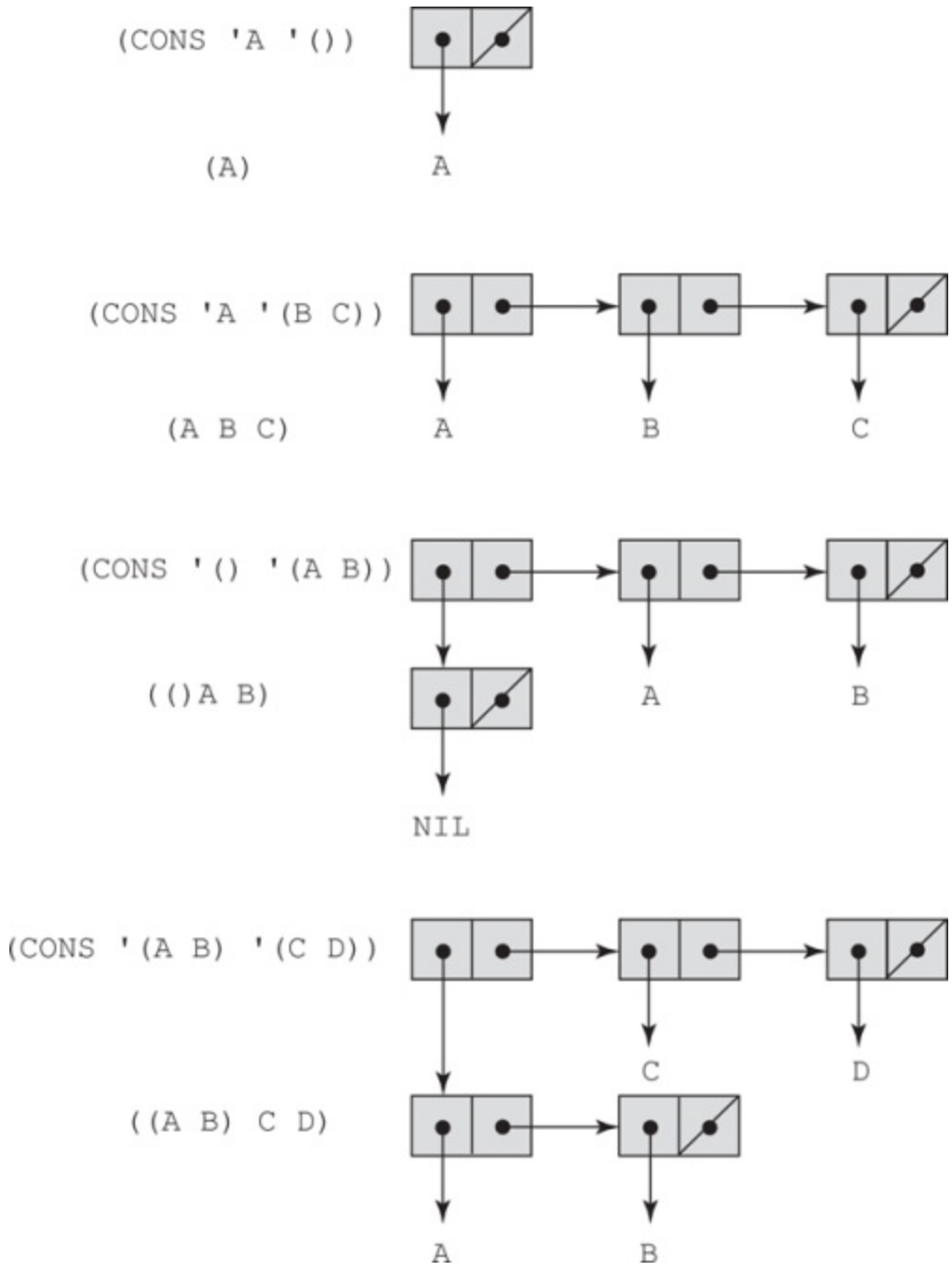


Figure 15.2 The result of several CONS operations

Figure 15.2 Full Alternative Text

Dealing only with the relatively simple problems and programs discussed in this chapter, it is unlikely one would intentionally apply CONS to two atoms, although that is legal. The result of such an application is a dotted pair, so named because of the way it is displayed by Scheme. For example, consider the following call:

```
(CONS 'A 'B)
```

If the result of this is displayed, it would appear as

```
(A . B)
```

This dotted pair indicates that instead of an atom and a pointer or a pointer and a pointer, this cell has two atoms.

LIST is a function that constructs a list from a variable number of parameters. It is a shorthand version of nested CONS functions, as illustrated in the following:

```
(LIST 'apple 'orange 'grape)
```

returns

```
(apple orange grape)
```

Using CONS, the call to LIST above is written as follows:

```
(CONS 'apple (CONS 'orange (CONS 'grape '())))
```

15.5.9 Predicate Functions for Symbolic Atoms and Lists

Scheme has three fundamental predicate functions, EQ?, NULL?, and LIST?, for symbolic atoms and lists.

The EQ? function takes two expressions as parameters, although it is usually used with two symbolic atom parameters. It returns #T if both parameters

have the same pointer value—that is, they point to the same atom or list; otherwise, it returns #F. If the two parameters are symbolic atoms, EQ? returns #T if they are the same symbols (because Scheme does not make duplicates of symbols); otherwise #F. Consider the following examples:

```
(EQ? 'A 'A) returns #T
(EQ? 'A 'B) returns #F
(EQ? 'A '(A B)) returns #F
(EQ? '(A B) '(A B)) returns #F or #T
(EQ? 3.4 (+ 3 0.4)) returns #F or #T
```

As the fourth example indicates, the result of comparing lists with EQ? is not consistent. The reason for this is that two lists that are exactly the same often are not duplicated in memory. At the time the Scheme system creates a list, it checks to see whether there is already such a list. If there is, the new list is nothing more than a pointer to the existing list. In these cases, the two lists will be judged equal by EQ?. However, in some cases, it may be difficult to detect the presence of an identical list, in which case a new list is created. In this scenario, EQ? yields #F.

The last case shows that the addition may produce a new value, in which case it would not be equal (with EQ?) to 3.4, or it may recognize that it already has the value 3.4 and use it, in which case EQ? will use the pointer to the old 3.4 and return #T.

As we have seen, EQ? works for symbolic atoms but does not necessarily work for numeric atoms. The = predicate works for numeric atoms but not symbolic atoms. As discussed previously, EQ? also does not work reliably for list parameters.

Sometimes it is convenient to be able to test two atoms for equality when it is not known whether they are symbolic or numeric. For this purpose, Scheme has a different predicate, EQV?, which works on both numeric and symbolic atoms. Consider the following examples:

```
(EQV? 'A 'A) returns #T
(EQV? 'A 'B) returns #F
(EQV? 3 3) returns #T
(EQV? 'A 3) returns #F
(EQV? 3.4 (+ 3 0.4)) returns #T
```

```
(EQV? 3.0 3) returns #F
```

Notice that the last example demonstrates that floating-point values are different from integer values. `EQV?` is not a pointer comparison, it is a value comparison.

The primary reason to use `EQ?` or `=` rather than `EQV?` when it is possible is that `EQ?` and `=` are faster than `EQV?`.

The `LIST?` predicate function returns `#T` if its single argument is a list and `#F` otherwise, as in the following examples:

```
(LIST? '(X Y)) returns #T
(LIST? 'X) returns #F
(LIST? '()) returns #T
```

The `NULL?` function tests its parameter to determine whether it is the empty list and returns `#T` if it is. Consider the following examples:

```
 (NULL? '(A B)) returns #F
(NULL? '()) returns #T
(NULL? 'A) returns #F
(NULL? '(())) returns #F
```

The last call yields `#F` because the parameter is not the empty list. Rather, it is a list containing a single element, the empty list.

15.5.10 Example Scheme Functions

This section contains several examples of function definitions in Scheme. These programs solve simple list-processing problems.

Consider the problem of membership of a given atom in a given list that does not include sublists. Such a list is called a **simple list**. If the function is named `member`, it could be used as follows:

```
(member 'B '(A B C)) returns #T
```

```
(member 'B '(A C D E)) returns #F
```

Thinking in terms of iteration, the membership problem is simply to compare the given atom and the individual elements of the given list, one at a time in some order, until either a match is found or there are no more elements in the list to be compared. A similar process can be accomplished using recursion. The function can compare the given atom with the CAR of the list. If they match, the value #T is returned. If they do not match, the CAR of the list should be ignored and the search continued on the CDR of the list. This can be done by having the function call itself with the CDR of the list as the list parameter and return the result of this recursive call. This process will end if the given atom is found in the list. If the atom is not in the list, the function will eventually be called (by itself) with a null list as the actual parameter. That event must force the function to return #F. In this process, there are two ways out of the recursion: Either the list is empty on some call, in which case #F is returned, or a match is found and #T is returned.

Altogether, there are three cases that must be handled in the function: an empty input list, a match between the atom and the CAR of the list, or a mismatch between the atom and the CAR of the list, which causes the recursive call. These three are the three parameters to COND, with the last being the default case that is triggered by an ELSE predicate. The complete function follows:4

4. Most Scheme systems define a function named member and do not allow a user to redefine it. So, if the reader wants to try this function, it must be defined with some other name.

```
(DEFINE (member atm a_list)
  (COND
    ((NULL? a_list) #F)
    ((EQ? atm (CAR a_list)) #T)
    (ELSE (member atm (CDR a_list)))
))
```

This form is typical of simple Scheme list-processing functions. In such functions, the data in lists are processed one element at a time. The individual elements are specified with CAR, and the process is continued using recursion on the CDR of the list.

Note that the null test must precede the equal test, because applying CAR to an empty list is an error.

As another example, consider the problem of determining whether two given lists are equal. If the two lists are simple, the solution is relatively easy, although some programming techniques with which the reader may not be familiar are involved. A predicate function, equalsimp, for comparing simple lists is shown here:

```
(DEFINE (equalsimp list1 list2)
  (COND
    ((NULL? list1) (NULL? list2))
    ((NULL? list2) #F)
    ((EQ? (CAR list1) (CAR list2))
          (equalsimp (CDR list1) (CDR list2)))
    (ELSE #F)
))
```

The first case, which is handled by the first parameter to COND, is for when the first list parameter is the empty list. This can occur in an external call if the first list parameter is initially empty. Because a recursive call uses the CDRs of the two parameter lists as its parameters, the first list parameter can be empty in such a call (if the first list parameter is now empty). When the first list parameter is empty, the second list parameter must be checked to see whether it is also empty. If so, they are equal (either initially or the CARs were equal on all previous recursive calls), and NULL? correctly returns #T. If the second list parameter is not empty, it is larger than the first list parameter and #F should be returned, as it is by NULL?.

The next case deals with the second list being empty when the first list is not. This situation occurs only when the first list is longer than the second. Only the second list must be tested, because the first case catches all instances of the first list being empty.

The third case is the recursive step that tests for equality between two corresponding elements in the two lists. It does this by comparing the CARs of the two nonempty lists. If they are equal, then the two lists are equal up to that point, so recursion is used on the CDRs of both. This case fails when two unequal atoms are found. When this occurs, the process need not continue, so

the default case ELSE is selected, which returns #F.

Note that equalsimp expects lists as parameters and does not operate correctly if either or both parameters are atoms.

The problem of comparing general lists is slightly more complex than this, because sublists must be traced completely in the comparison process. In this situation, the power of recursion is uniquely appropriate, because the form of sublists is the same as that of the given lists. Any time the corresponding elements of the two given lists are lists, they are separated into their two parts, CAR and CDR, and recursion is used on them. This is a perfect example of the usefulness of the divide-and-conquer approach. If the corresponding elements of the two given lists are atoms, they can simply be compared using EQ?.

The definition of the complete function follows:

```
(DEFINE (equal list1 list2)
  (COND
    ((NOT (LIST? list1)) (EQ? list1 list2))
    ((NOT (LIST? list2)) #F)
    ((NULL? list1) (NULL? list2))
    ((NULL? list2) #F)
    ((equal (CAR list1) (CAR list2))
            (equal (CDR list1) (CDR list2)))
    (ELSE #F)
))
```

The first two cases of the COND handle the situation where either of the parameters is an atom instead of a list. The third and fourth cases are for the situation where one or both lists are empty. These cases also prevent subsequent cases from attempting to apply CAR to an empty list. The fifth COND case is the most interesting. The predicate is a recursive call with the CARs of the lists as parameters. If this recursive call returns #T, then recursion is used again on the CDRs of the lists. This algorithm allows the two lists to include sublists to any depth.

This definition of equal works on any pair of expressions, not just lists. equal is equivalent to the system predicate function EQUAL?. Note that EQUAL? should be used only when necessary (the forms of the actual parameters are

not known), because it is much slower than EQ? and EQV?.

Another commonly needed list operation is that of constructing a new list that contains all of the elements of two given list arguments. This is usually implemented as a Scheme function named append. The result list can be constructed by repeated use of CONS to place the elements of the first list argument into the second list argument, which becomes the result list. To clarify the action of append, consider the following examples:

```
(append '(A B) '(C D R)) returns (A B C D R)
(append '((A B) C) '(D (E F))) returns ((A B) C D (E F))
```

The definition of append is[5]

5. As was the case with member, a user usually cannot define a function named append.

```
(DEFINE (append list1 list2)
  (COND
    ((NULL? list1) list2)
    (ELSE (CONS (CAR list1) (append (CDR list1) list2)))
))
```

The first COND case is used to terminate the recursive process when the first argument list is empty, returning the second list. In the second case (the ELSE), the CAR of the first parameter list is CONSed onto the result returned by the recursive call, which passes the CDR of the first list as its first parameter.

Consider the following Scheme function, named guess, which uses the member function described in this section. Try to determine what it does before reading the description that follows it. Assume the parameters are simple lists.

```
(DEFINE (guess list1 list2)
  (COND
    ((NULL? list1) '())
    ((member (CAR list1) list2)
             (CONS (CAR list1) (guess (CDR list1) list2)))
    (ELSE (guess (CDR list1) list2))
))
```

guess yields a simple list that contains the common elements of its two parameter lists. So, if the parameter lists represent sets, guess computes a list that represents the intersection of those two sets.

15.5.11 LET

LET is a function (initially described in Chapter 5) that creates a local scope in which names are temporarily bound to the values of expressions. It is often used to factor out the common subexpressions from more complicated expressions. These names can then be used in the evaluation of another expression, but they cannot be rebound to new values in LET. The following example illustrates the use of LET. It computes the roots of a given quadratic equation, assuming the roots are real.6 The mathematical definitions of the real (as opposed to complex) roots of the quadratic equation ax2+bx+c are as follows: root1=(−b+sqrt(b2−4 ac))/2a and root 2=(−b−sqrt(b2−4 ac))/2 a

6. Some versions of Scheme include “complex” as a data type and will compute the roots of the equation, regardless of whether they are real or complex.

```
(DEFINE (quadratic_roots a b c)
  (LET (
    (root_part_over_2a
                (/ (SQRT (- (* b b) (* 4 a c))) (* 2 a)))
    (minus_b_over_2a (/ (- 0 b) (* 2 a)))
        )
  (LIST (+ minus_b_over_2a root_part_over_2a)
             (- minus_b_over_2a root_part_over_2a))
))
```

This example uses LIST to create the list of the two values that make up the result.

Because the names bound in the first part of a LET construct cannot be changed in the following expression, they are not the same as local variables in a block in an imperative language. They could all be eliminated by textual substitution of their respective expressions for their names in the LET expression.

LET is actually shorthand for a LAMBDA expression applied to a parameter. The following two expressions are equivalent:

```
(LET ((alpha 7))(* 5 alpha))
((LAMBDA (alpha) (* 5 alpha)) 7)
```

In the first expression, 7 is bound to alpha with LET; in the second, 7 is bound to alpha through the parameter of the LAMBDA expression.

15.5.12 Tail Recursion in Scheme

A function is **tail recursive** if its recursive call is the last operation in the function. This means that the return value of the recursive call is the return value of the nonrecursive call to the function. For example, the member function of Section 15.5.10, repeated here, is tail recursive.

```
(DEFINE (member atm a_list)
  (COND
    ((NULL? a_list) #F)
    ((EQ? atm (CAR a_list)) #T)
    (ELSE (member atm (CDR a_list)))
))
```

This function can be automatically converted by a compiler to use iteration, resulting in faster execution than in its recursive form.

However, many functions that use recursion for repetition are not tail recursive. Programmers who are concerned with efficiency have discovered ways to rewrite some of these functions so that they are tail recursive. One example of this uses an accumulating parameter and a helper function. As an example of this approach, consider the factorial function from Section 15.5.7, which is repeated here:

```
(DEFINE (factorial n)
  (IF (<= n 1)
    1
    (* n (factorial (- n 1)))
))
```

The last operation of this function is the multiplication. The function works by creating the list of numbers to be multiplied together and then doing the multiplications as the recursion unwinds to produce the result. Each of these numbers is created by an activation of the function and each is stored in an activation record instance. As the recursion unwinds the numbers are multiplied together. Recall that the stack is shown after several recursive calls to factorial in Chapter 9. This factorial function can be rewritten with an auxiliary helper function, which uses a parameter to accumulate the partial factorial. The helper function, which is tail recursive, also takes `factorial`'s parameter. These functions are as follows:

```
(DEFINE (facthelper n factpartial)
  (IF (<= n 1)
    factpartial
    (facthelper (- n 1) (* n factpartial))
))
(DEFINE (factorial n)
  (facthelper n 1)
)
```

With these functions, the result is computed during the recursive calls, rather than as the recursion unwinds. Because there is nothing useful in the activation record instances, they are not necessary. Regardless of how many recursive calls are requested, only one activation record instance is necessary. This makes the tail-recursive version far more efficient than the non-tail-recursive version.

The Scheme language definition requires that Scheme language processing systems convert all tail-recursive functions to replace that recursion with iteration. Therefore, it is important, at least for efficiency's sake, to define functions that use recursion to specify repetition to be tail recursive. Some optimizing compilers for some functional languages can even perform conversions of some non-tail-recursive functions to equivalent tail-recursive functions and then code these functions to use iteration instead of recursion for repetition.

15.5.13 Functional Forms

This section describes two common mathematical functional forms that are provided by Scheme: composition and apply-to-all. Both are mathematically defined in Section 15.2.2.

15.5.13.1 Functional Composition

Functional composition is the only primitive functional form provided by the original Lisp. All subsequent Lisp dialects, including Scheme, also provide it. As stated in Section 15.2.2, function composition is a functional form that takes two functions as parameters and returns a function that first applies the second parameter function to its parameter and then applies the first parameter function to the return value of the second parameter function. In other words, the function `h` is the composition function of `f` and `g` if `h(x) = f(g(x))`. For example, consider the following example:

```
(DEFINE (g x) (* 3 x))
(DEFINE (f x) (+ 2 x))
```

Now the functional composition of `f` and `g` can be written as follows:

```
(DEFINE (h x) (+ 2 (* 3 x)))
```

In Scheme, the functional composition function `compose` can be written as follows:

```
(DEFINE (compose f g) (LAMBDA (x)(f (g x))))
```

For example, we could have the following:

```
((compose CAR CDR) '((a b) c d))
```

This call would yield `c`. This is an alternative, though less efficient, form of `CADR`. Now consider another call to `compose`:

```
((compose CDR CAR) '((a b) c d))
```

This call would yield `(b)`. This is an alternative to `CDAR`.

As yet another example of the use of compose, consider the following:

```
(DEFINE (third a_list)
 ((compose CAR (compose CDR CDR)) a_list))
```

This is an alternative to CADDR.

15.5.13.2 An Apply-to-All Functional Form

The most common functional forms provided in functional programming languages are variations of mathematical apply-to-all functional forms. The simplest of these is map, which has two parameters: a function and a list. map applies the given function to each element of the given list and returns a list of the results of these applications. A Scheme definition of map follows:7

7. As was the case with member, map is a predefined function that cannot be redefined by users.

```
(DEFINE (map fun a_list)
  (COND
    ((NULL? a_list) '())
    (ELSE (CONS (fun (CAR a_list)) (map fun (CDR a_list))))
))
```

Note the simple form of map, which expresses a complex functional form.

As an example of the use of map, suppose we want all of the elements of a list cubed. We can accomplish this with the following:

```
(map (LAMBDA (num) (* num num num)) '(3 4 2 6))
```

This call returns (27 64 8 216).

Note that in this example, the first parameter to mapcar is a LAMBDA expression. When EVAL evaluates the LAMBDA expression, it constructs a function that has the same form as any predefined function except that it is

nameless. In the example expression, this nameless function is immediately applied to each element of the parameter list and the results are returned in a list.

15.5.14 Functions That Build Code

The fact that programs and data have the same structure can be exploited in constructing programs. Recall that the Scheme interpreter uses a function named EVAL. The Scheme system applies EVAL to every expression typed, whether it is at the Scheme prompt in the interactive interpreter or is part of a program being interpreted. The EVAL function can also be called directly by Scheme programs. This provides the possibility of a Scheme program creating expressions and calling EVAL to evaluate them. This is not something that is unique to Scheme, but the simple forms of its expressions make it easy to create them during execution.

One of the simplest examples of this process involves numeric atoms. Recall that Scheme includes a function named +, which takes any number of numeric atoms as arguments and returns their sum. For example, `(+ 3 7 10 2)` returns `22`.

Our problem is the following: Suppose that in a program we have a list of numeric atoms and need the sum. We cannot apply + directly on the list, because + can take only atomic parameters, not a list of numeric atoms. We could, of course, write a function that repeatedly adds the CAR of the list to the sum of its CDR, using recursion to go through the list. Such a function follows:

```
(DEFINE (adder a_list)
  (COND
    ((NULL? a_list) 0)
    (ELSE (+ (CAR a_list) (adder (CDR a_list))))
))
```

Following is an example call to `adder`, along with the recursive calls and returns:

```
(adder '(3 4 5))
(+ 3 (adder (4 5)))
```

```
(+ 3 (+ 4 (adder (5))))
(+ 3 (+ 4 (+ 5 (adder ()))))
(+ 3 (+ 4 (+ 5 0)))
(+ 3 (+ 4 5))
(+ 3 9)
(12)
```

An alternative solution to the problem is to write a function that builds a call to + with the proper parameter forms. This can be done by using CONS to build a new list that is identical to the parameter list except it has the atom + inserted at its beginning. This new list can then be submitted to EVAL for evaluation, as in the following:

```
(DEFINE (adder a_list)
  (COND
    ((NULL? a_list) 0)
    (ELSE (EVAL (CONS '+ a_list)))
))
```

Note that the + function's name is quoted to prevent EVAL from evaluating it in the evaluation of CONS. Following is an example call to this new version of adder, along with the call to EVAL and the return value:

```
(adder '(3 4 5))
(EVAL (+ 3 4 5)
(12)
```

In all earlier versions of Scheme, the EVAL function evaluated its expression in the outermost scope of the program. The later versions of Scheme, beginning with Scheme 4, requires a second parameter to EVAL that specifies the scope in which the expression is to be evaluated. For simplicity's sake, we left the scope parameter out of our example, and we do not discuss scope names here.

15.6 Common Lisp

Common Lisp (Steele, 1990) was created in an effort to combine the features of several early 1980s dialects of Lisp, including Scheme, into a single language. Being something of a union of languages, it is quite large and complex, similar in these regards to C++ and C#. Its basis, however, is the original Lisp, so its syntax, primitive functions, and fundamental nature come from that language.

Following is the factorial function written in Common Lisp:

```
(DEFUN factorial (x)
  (IF (<= n 1)
    1
    (* n factorial (- n 1)))
))
```

Only the first line of this function differs syntactically from the Scheme version of the same function.

The list of features of Common Lisp is long: a large number of data types and structures, including records, arrays, complex numbers, and character strings; powerful input and output operations; and a form of packages for modularizing collections of functions and data, and also for providing access control. Common Lisp includes several imperative constructs, as well as some mutable types.

Recognizing the occasional flexibility provided by dynamic scoping, as well as the simplicity of static scoping, Common Lisp allows both. The default scoping for variables is static, but by declaring a variable to be “special,” that variable becomes dynamically scoped.

Macros are often used in Common Lisp to extend the language. In fact, some of the predefined functions are actually macros. For example, `DOLIST`, which takes two parameters, a variable and a list, is a macro. For example, consider the following:

```
(DOLIST (x '(1 2 3)) (print x))
```

This produces the following:

```
1
2
3
NIL
```

`NIL` here is the return value of `DOLIST`.

Macros create their effect in two steps: First, the macro is expanded. Second, the expanded macro, which is Lisp code, is evaluated. Users can define their own macros with `DEFMACRO`.

The Common Lisp backquote operator (`` ` ``) is similar to Scheme's `QUOTE`, except some parts of the parameter can be unquoted by preceding them with commas. For example, consider the following two examples:

```
`(a (* 3 4) c)
```

This expression evaluates to `(a (* 3 4) c)`. However, the following expression:

```
`(a ,(* 3 4) c)
```

evaluates to `(a 12 c)`.

Lisp implementations have a front end called the *reader* that transforms the text of Lisp programs into a code representation. Then, the macro calls in the code representation are expanded into code representations. The output of this step is then either interpreted or compiled into the machine language of the host computer, or perhaps into an intermediate code than can be interpreted. There is a special kind of macro, named *reader macros* or *read macros*, that are expanded during the reader phase of a Lisp language processor. A reader macro expands a specific character into a string of Lisp code. For example, the apostrophe in Lisp is a read macro that expands to a call to `QUOTE`. Users can define their own reader macros to create other shorthand constructs.

Common Lisp, as well as other Lisp-based languages, have a symbol data type. The reserved words are symbols that evaluate to themselves, as are `T` and `NIL`. Technically, symbols are either bound or unbound. Parameter symbols are bound while the function is being evaluated. Also, symbols that are the names of imperative-style variables and have been assigned values are bound. Other symbols are unbound. For example, consider the following expression:

```
(LIST '(A B C))
```

The symbols `A`, `B`, and `C` are unbound. Recall that Ruby also has a symbol data type.

In a sense, Scheme and Common Lisp are opposites. Scheme is far smaller and semantically simpler, in part because of its exclusive use of static scoping, but also because it was designed to be used for teaching programming, whereas Common Lisp was meant to be a commercial language. Common Lisp has succeeded in being a widely used language for AI applications, among other areas. Scheme, on the other hand, is more frequently used in college courses on functional programming. It is also more likely to be studied as a functional language because of its relatively small size. An important design goal of Common Lisp that caused it to be a large language was the desire to make it compatible with several of the dialects of Lisp from which it was derived.

The Common Lisp Object System (CLOS) (Paepeke, 1993) was developed in the late 1980s as an object-oriented version of Common Lisp. This language supports generic functions and multiple inheritance, among other constructs.

15.7 ML

ML (Milner et al., 1997) is a static-scoped functional programming language, like Scheme. However, it differs from Lisp and its dialects, including Scheme, in a number of significant ways. One important difference is that ML is a strongly typed language, whereas Scheme is essentially typeless. ML has type declarations for function parameters and the return types of functions, although because of its type inferencing they are often not used. The type of every variable and expression can be statically determined. ML, like other functional programming languages, does not have variables in the sense of the imperative languages. It does have identifiers, which have the appearance of names of variables in imperative languages. However, these identifiers are best thought of as names for values. Once set, they cannot be changed. They are like the named constants of imperative languages like **`final`** declarations in Java. ML identifiers do not have fixed types—any identifier can be the name of a value of any type.

A table called the **evaluation environment** stores the names of all implicitly and explicitly declared identifiers in a program, along with their types. This is like a run-time symbol table. When an identifier is declared, either implicitly or explicitly, it is placed in the evaluation environment.

Another important difference between Scheme and ML is that ML uses a syntax that is more closely related to that of an imperative language than that of Lisp. For example, arithmetic expressions are written in ML using infix notation.

Function declarations in ML appear in the general form

```
fun function_name(formal parameters) = expression;
```

When called, the value of the expression is returned by the function. Actually, the expression can be a list of expressions, separated by semicolons and surrounded by parentheses. The return value in this case is that of the last expression. Of course, unless they have side effects, the expressions before the last serve no purpose. Because we are not considering the parts of ML

that have side effects, we only consider function definitions with a single expression.

Now we can discuss type inference. Consider the following ML function declaration:

```
fun circumf(r) = 3.14159 * r * r;
```

This specifies a function named `circumf` that takes a floating-point (**real** in ML) parameter and produces a floating-point result. The types are inferred from the type of the literal in the expression. Likewise, in the function

```
fun times10(x) = 10 * x;
```

the parameter and functional value are inferred to be of type **int**.

Consider the following ML function:

```
fun square(x) = x * x;
```

ML determines the type of both the parameter and the return value from the * operator in the function definition. Because this is an arithmetic operator, the type of the parameter and the function are assumed to be numeric. In ML, the default numeric type is **int**. So, it is inferred that the type of the parameter and the return value of `square` is **int**.

If `square` were called with a floating-point value, as in

```
square(2.75);
```

it would cause an error, because ML does not coerce **real** values to **int** type. If we wanted `square` to accept **real** parameters, it could be rewritten as

```
fun square(x) : real = x * x;
```

Because ML does not allow overloaded functions, this version could not coexist with the earlier **int** version. The last version defined would be the only one.

The fact that the functional value is typed **real** is sufficient to infer that the

parameter is also **real** type. Each of the following definitions is also legal:

```
fun square(x : real) = x * x;
fun square(x) = (x : real) * x;
fun square(x) = x * (x : real);
```

Type inference is also used in the functional languages Miranda, Haskell, and F#.

The ML selection control flow construct is similar to that of the imperative languages. It has the following general form:

```
if expression then then_expression else else_expression
```

The first expression must evaluate to a Boolean value.

The conditional expressions of Scheme can appear at the function definition level in ML. In Scheme, the `COND` function is used to determine the value of the given parameter, which in turn specifies the value returned by `COND`. In ML, the computation performed by a function can be defined for different forms of the given parameter. This feature is meant to mimic the form and meaning of conditional function definitions in mathematics. In ML, the particular expression that defines the return value of a function is chosen by pattern matching against the given parameter. For example, without using this pattern matching, a function to compute factorial could be written as follows:

```
fun fact(n : int): int = if n <= 1 then 1
                         else n * fact(n - 1);
```

Multiple definitions of a function can be written using parameter pattern matching. The different function definitions that depend on the form of the parameter are separated by an OR symbol (`|`). For example, using pattern matching, the factorial function could be written as follows:

```
fun fact(0) = 1
|   fact(1) = 1
|  fact(n : int): int = n * fact(n - 1);
```

If `fact` is called with the actual parameter `0`, the first definition is used; if the

actual parameter is 1, the second definition is used; if an **`int`** value that is neither `0` nor `1` is sent, the third definition is used.

As discussed in Chapter 6, ML supports lists and list operations. Recall that `hd`, `tl`, and `::` are ML's versions of Scheme's `CAR`, `CDR`, and `CONS`.

Because of the availability of patterned function parameters, the `hd` and `tl` functions are much less frequently used in ML than `CAR` and `CDR` are used in Scheme. For example, in a formal parameter, the expression

```
(h :: t)
```

is actually two formal parameters, the head and tail of given list parameter, while the single corresponding actual parameter is a list. For example, the number of elements in a given list can be computed with the following function:

```
fun length([]) = 0
|   length(h :: t) = 1 + length(t);
```

As another example of these concepts, consider the `append` function, which does what the Scheme `append` function does:

```
fun append([], lis2) = lis2
|   append(h :: t, lis2) = h :: append(t, lis2);
```

The first case in this function handles the situation of the function being called with an empty list as the first parameter. This case also terminates the recursion when the initial call has a nonempty first parameter. The second case of the function breaks the first parameter list into its head and tail (`hd` and `tl`). The head is `CONS`ed onto the result of the recursive call, which uses the tail as its first parameter.

In ML, names are bound to values with value declaration statements of the form

```
val new_name = expression;
```

For example,

```
val distance = time * speed;
```

Do not get the idea that this statement is exactly like the assignment statements in the imperative languages, for it is not. The **val** statement binds a name to a value, but the name cannot be later rebound to a new value. Well, in a sense it can. Actually, if you do rebind a name with a second **val** statement, it causes a new entry in the evaluation environment that is not related to the previous version of the name. In fact, after the new binding, the old evaluation environment entry (for the previous binding) is no longer visible. Also, the type of the new binding need not be the same as that of the previous binding. **val** statements do not have side effects. They simply add a name to the current evaluation environment and bind it to a value.

The normal use of **val** is in a **let** expression.8 Consider the following example:

8. Let expressions in ML were introduced in Chapter 5.

```
let val radius = 2.7
    val pi = 3.14159
in pi * radius * radius
end;
```

ML includes several higher-order functions that are commonly used in functional programming. Among these are a filtering function for lists, `filter`, which takes a predicate function as its parameter. The predicate function is often given as a lambda expression, which in ML is defined exactly like a function, except with the **fn** reserved word, instead of **fun**, and of course the lambda expression is nameless. `filter` returns a function that takes a list as a parameter. It tests each element of the list with the predicate. Each element on which the predicate returns true is added to a new list, which is the return value of the function. Consider the following use of `filter`:

```
filter(fn(x) => x < 100, [25, 1, 50, 711, 100, 150, 27,
    161, 3]);
```

This application would return `[25, 1, 50, 27, 3]`.

The `map` function takes a single parameter, which is a function. The resulting function takes a list as a parameter. It applies its function to each element of

the list and returns a list of the results of those applications. Consider the following code:

```
fun cube x = x * x * x;
val cubeList = map cube;
val newList = cubeList [1, 3, 5];
```

After execution, the value of `newList` is `[1, 27, 125]`. This could be done more simply by defining the cube function as a lambda expression, as in the following:

```
val newList = map (fn x => x * x * x, [1, 3, 5]);
```

ML has a binary operator for composing two functions, `o` (a lowercase "oh"). For example, to build a function `h` that first applies function `f` and then applies function `g` to the returned value from `f`, we could use the following:

```
val h = g o f;
```

Strictly speaking, ML functions take a single parameter. When a function is defined with more than one parameter, ML considers the parameters to be a tuple, even though the parentheses that normally delimit a tuple value are optional. The commas that separate the parameters (tuple elements) are required.

The process of **currying** replaces a function with more than one parameter with a function with one parameter that returns a function that takes the other parameters of the initial function.

ML functions that take more than one parameter can be defined in curried form by leaving out the commas between the parameters (and the delimiting parentheses).9 For example, we could have the following:

9. This form of functions is named for Haskell Curry, a British mathematician who studied them.

```
fun add a b = a + b;
```

Although this appears to define a function with two parameters, it actually defines one with just one parameter. The `add` function takes an integer

parameter (`a`) and returns a function that also takes an integer parameter (`b`). A call to this function also excludes the commas between the parameters, as in the following:

```
add 3 5;
```

This call to `add` returns `8`, as expected.

Curried functions are interesting and useful because new functions can be constructed from them by partial evaluation. **Partial evaluation** means that the function is evaluated with actual parameters for one or more of the leftmost formal parameters. For example, we could define a new function as follows:

```
fun add5 x = add 5 x;
```

The `add5` function takes the actual parameter `5` and evaluates the `add` function with `5` as the value of its first formal parameter. It returns a function that adds `5` to its single parameter, as in the following:

```
val num = add5 10;
```

The value of `num` is now `15`. We could create any number of new functions from the curried function `add` to add any specific number to a given parameter.

Curried functions also can be written in Scheme, Haskell, and F#. Consider the following Scheme function:

```
(DEFINE (add x y) (+ x y))
```

A curried version of this would be as follows:

```
DEFINE (add y) (LAMBDA (x) (+ y x)))
```

This can be called as follows:

```
((add 3) 4)
```

ML has enumerated types, arrays, and tuples. ML also has exception

handling and a module facility for implementing abstract data types.

ML has had a significant impact on the evolution of programming languages. For language researchers, it has become one of the most studied languages. Furthermore, it has spawned several subsequent languages, among them Haskell, Caml, OCaml, and F#.

15.8 Haskell

Haskell (Thompson, 1999) is similar to ML in that it uses a similar syntax, is static scoped, is strongly typed, and uses the same type inferencing method. There are three characteristics of Haskell that set it apart from ML: First, functions in Haskell can be overloaded (functions in ML cannot). Second, nonstrict semantics are used in Haskell, whereas in ML (and most other programming languages) strict semantics are used. Third, Haskell is a pure functional programming language, meaning it has no expressions or statements that have side effects, whereas ML allows some side effects (for example, ML has mutable arrays). Both nonstrict semantics and function overloading are further discussed later in this section.

The code in this section is written in version 1.4 of Haskell.

Consider the following definition of the factorial function, which uses pattern matching on its parameters:

```
fact 0 = 1
fact 1 = 1
fact n = n * fact (n - 1)
```

Note the differences in syntax between this definition and its ML version in Section 15.7. First, there is no reserved word to introduce the function definition (**fun** in ML). Second, alternative definitions of functions (with different formal parameters) all have the same appearance.

Using pattern matching, we can define a function for computing the nth Fibonacci number with the following:

```
fib 0 = 1
fib 1 = 1
fib (n + 2) = fib (n + 1) + fib n
```

Guards can be added to lines of a function definition to specify the circumstances under which the definition can be applied. For example,

```
fact n
  | n == 0 = 1
  | n == 1 = 1
  | n > 1 = n * fact(n - 1)
```

This definition of factorial is more precise than the previous one, as it restricts the range of actual parameter values to those for which it works. This form of a function definition is called a *conditional expression*, after the mathematical expressions on which it is based.

An **`otherwise`** can appear as the last condition in a conditional expression, with the obvious semantics. For example,

```
sub n
  | n < 10    = 0
  | n > 100   = 2
  | otherwise = 1
```

Notice the similarity between the guards here and the guarded commands discussed in Chapter 8.

Consider the following function definition, whose purpose is the same as the corresponding ML function in Section 15.7:

```
square x = x * x
```

In this case, however, because of Haskell's support for polymorphism, this function can take a parameter of any numeric type.

As with ML, lists are written in brackets in Haskell, as in

```
colors = [”blue”, ”green”, ”red”, ”yellow”]
```

Haskell includes a collection of list operators. For example, lists can be catenated with `++`, `:` serves as an infix version of `CONS`, and `..` is used to specify an arithmetic series in a list. For example,

```
5:[2, 7, 9] results in [5, 2, 7, 9]
[1, 3..11] results in [1, 3, 5, 7, 9, 11]
[1, 3, 5] ++ [2, 4, 6] results in [1, 3, 5, 2, 4, 6]
```

Notice that the : operator is just like ML's :: operator.[10] Using : and pattern matching, we can define a simple function to compute the product of a given list of numbers:

10. It is interesting that ML uses : for attaching a type name to a name and :: for CONS, while Haskell uses these two operators in exactly opposite ways.

```
product [] = 1
product (a:x) = a * product x
```

Using `product`, we can write a factorial function in the simpler form

```
fact n = product [1..n]
```

Haskell includes a **`let`** construct that is similar to ML's **`let`** and **`val`**. For example, we could write

```
quadratic_root a b c =
    let minus_b_over_2a = - b / (2.0 * a)
     root_part_over_2a =
                sqrt(b ^ 2 - 4.0 * a * c) / (2.0 * a)
    in
     minus_b_over_2a - root_part_over_2a,
     minus_b_over_2a + root_part_over_2a
```

Haskell's list comprehensions were introduced in Chapter 6. For example, consider the following example of a list comprehension:

```
[n * n * n | n <- [1..50]]
```

This defines a list of the cubes of the numbers from 1 to 50. It is read as "a list of all `n*n*n` such that `n` is taken from the range of 1 to 50." In this case, the qualifier is in the form of a **generator**. It generates the numbers from `1` to `50`. In other cases, the qualifiers are in the form of Boolean expressions and they are called **tests**. This notation can be used to describe algorithms for doing many things, such as finding permutations of lists and sorting lists. For example, consider the following function, which when given a number `n` returns a list of all its factors:

```
factors n = [ i | i <-  [1..n `div` 2], n `mod` i == 0]
```

The list comprehension in `factors` creates a list of numbers, each temporarily bound to the name `i`, ranging from `1` to `n/2`, such that `n `mod` i` is zero. This is indeed a very exacting and short definition of the factors of a given number. The backticks (backward apostrophes) surrounding **`div`** and **`mod`** are used to specify the infix use of these functions. When they are called in functional notation, as in **`div`** `n 2`, the backticks are not used.

Next, consider the concision of Haskell illustrated in the following implementation of the quicksort algorithm:

```
sort [] = []
sort (h:t) = sort [b | b <- t, b <- h]
             ++ [h] ++
             sort [b | b <- t, b > h]
```

In this program, the set of list elements that are smaller or equal to the list head are sorted and catenated with the head element, then the set of elements that are greater than the list head are sorted and catenated onto the previous result. This definition of quicksort is significantly shorter and simpler than the same algorithm coded in an imperative language.

A programming language is **strict** if it requires all actual parameters to be fully evaluated, which ensures that the value of a function does not depend on the order in which the parameters are evaluated. A language is **nonstrict** if it does not have the strict requirement. Nonstrict languages can have several distinct advantages over strict languages. First, nonstrict languages are generally more efficient, because some evaluation is avoided.11 Second, some interesting capabilities are possible with nonstrict languages that are not possible with strict languages. Among these are infinite lists. Nonstrict languages can use an evaluation form called **lazy evaluation**, which means that expressions are evaluated only if and when their values are needed.

11. Notice how this is related to short-circuit evaluation of Boolean expressions, which is done in some imperative languages.

Recall that in Scheme the parameters to a function are fully evaluated before the function is called, so it has strict semantics. Lazy evaluation means that an actual parameter is evaluated only when its value is necessary to evaluate the function. So, if a function has two parameters, but on a particular

execution of the function the first parameter is not used, the actual parameter passed for that execution will not be evaluated. Furthermore, if only a part of an actual parameter must be evaluated for an execution of the function, the rest is left unevaluated. Finally, actual parameters are evaluated only once, if at all, even if the same actual parameter appears more than once in a function call.

As stated previously, lazy evaluation allows one to define infinite data structures. For example, consider the following:

```
positives = [0..]
evens = [2, 4..]
squares = [n * n | n <- [0..]]
```

Of course, no computer can actually represent all of the numbers of these lists, but that does not prevent their use if lazy evaluation is used. For example, if we wanted to know if a particular number was a perfect square, we could check the `squares` list with a membership function. Suppose we had a predicate function named `member` that determined whether a given atom is contained a given list. Then we could use it as in

```
member 16 squares
```

which would return `True`. The `squares` definition would be evaluated until the `16` was found. The `member` function would need to be carefully written. Specifically, suppose it were defined as follows:

```
member b [] = False
member b (a:x)= (a == b) || member b x
```

The second line of this definition breaks the first parameter into its head and tail. Its return value is true if either the head matches the element for which it is searching (`b`) or if the recursive call with the tail of the list returns `True`.

This definition of `member` would work correctly with `squares` only if the given number were a perfect square. If not, `squares` would keep generating squares forever, or until some memory limitation was reached, looking for the given number in the list. The following function performs the membership test of an ordered list, abandoning the search and returning

`False` if a number greater than the searched-for number is found.12

12. This assumes that the list is in ascending order.

```
member2 n (m:x)
  | m < n     = member2 n x
  | m == n    = True
  | otherwise = False
```

Lazy evaluation sometimes provides a modularization tool. Suppose that in a program there is a call to function `f` and the parameter to `f` is the return value of a function `g`.13 So, we have `f(g(x))`. Further suppose that `g` produces a large amount of data, a little at a time, and that `f` must then process this data, a little at a time. In a conventional imperative language, `g` would run on the whole input producing a file of its output. Then `f` would run using the file as its input. This approach requires the time to both write and read the file, as well as the storage for the file. With lazy evaluation, the executions of `f` and `g` implicitly would be tightly synchronized. Function `g` will execute only long enough to produce enough data for `f` to begin its processing. When `f` is ready for more data, `g` will be restarted to produce more, while `f` waits. If `f` terminates without getting all of `g`'s output, `g` is aborted, thereby avoiding useless computation. Also, `g` need not be a terminating function, perhaps because it produces an infinite amount of output. `g` will be forced to terminate when `f` terminates. So, under lazy evaluation, `g` runs as little as possible. This evaluation process supports the modularization of programs into generator units and selector units, where the generator produces a large number of possible results and the selector chooses the appropriate subset.

13. This example appears in Hughes (1989).

Lazy evaluation is not without its costs. It would certainly be surprising if such expressive power and flexibility were free. In this case, the cost is in a far more complicated semantics, which results in much slower speed of execution.

15.9 F#

F# is a .NET functional programming language whose core is based on OCaml, which is a descendant of ML and Haskell. Although it is fundamentally a functional language, it includes imperative features and supports object-oriented programming. One of the most important characteristics of F# is that it has a full-featured IDE, an extensive library of utilities that supports imperative, object-oriented, and functional programming, and has interoperability with a collection of nonfunctional languages (all of the .NET languages).

F# is a first-class .NET language. This means that F# programs can interact in every way with other .NET languages. For example, F# classes can be used and subclassed by programs in other languages, and vice versa. Furthermore, F# programs have access to all of the .NET Framework APIs. The F# implementation is available free from Microsoft (`http://research.microsoft.com/fsharp/fsharp.aspx`). It is also supported by Visual Studio.

F# includes a variety of data types. Among these are tuples, like those of Python and the functional languages ML and Haskell, lists, discriminated unions, an expansion of ML's unions, and records, like those of ML, which are like tuples except the components are named. F# has both mutable and immutable arrays.

Recall from Chapter 6, that F#'s lists are similar to those of ML, except that the elements are separated by semicolons and `hd` and `tl` must be called as methods of `List`.

F# supports sequence values, which are types from the .NET namespace `System.Collections.Generic.IEnumerable`. In F#, sequences are abbreviated as `seq<type>`, where <type> indicates the type of the generic. For example, the type `seq<int>` is a sequence of integer values. Sequence values can be created with generators and they can be iterated. The simplest sequences are generated with range expressions, as in the following example:

```
let x = seq {1..4};;
```

In the examples of F#, we assume that the interactive interpreter is used, which requires the two semicolons at the end of each statement. The expression above generates `seq[1; 2; 3; 4]`. (List and sequence elements are separated by semicolons.) The generation of a sequence is lazy; for example, the following defines `y` to be a very long sequence, but only the needed elements are generated. For display, only the first four are generated.

```
let y = seq {0..100000000};;
y;;
val it: seq<int> = seq[0; 1; 2; 3;...]
```

The first line above defines `y`; the second line requests that the value of `y` be displayed; the third is the output of the F# interactive interpreter.

The default step size for integer sequence definitions is `1`, but it can be set by including it in the middle of the range specification, as in the following example:

```
seq {1..2..7};;
```

This generates `seq [1; 3; 5; 7]`.

The values of a sequence can be iterated with a **`for-in`** construct, as in the following example:

```
let seq1 = seq {0..3..11};;
for value in seq1 do printfn ”value = %d” value;;
```

This produces the following:

```
value = 0
value = 3
value = 6
value = 9
```

Iterators can also be used to create sequences, as in the following example:

```
let cubes = seq {for i in 1..5 -> (i, i * i * i)};;
```

This generates the following list of tuples:

```
seq [(1, 1); (2, 8); (3, 27); (4, 64); (5, 125)]
```

This use of iterators to generate collections is a form of list comprehension.

Sequencing can also be used to generate lists and arrays, although in these cases the generation is not lazy. In fact, the primary difference between lists and sequences in F# is that sequences are lazy, and thus can be infinite, whereas lists are not lazy. Lists are in their entirety stored in memory. That is not the case with sequences.

The functions of F# are similar to those of ML and Haskell. If named, they are defined with **`let`** statements. If unnamed, which means technically they are lambda expressions, they are defined with the **`fun`** reserved word. The following lambda expression illustrates their syntax:

```
(fun  a b -> a / b)
```

Note that there is no difference between a name defined with **`let`** and a function without parameters defined with **`let`**.

Indentation is used to show the extent of a function definition. For example, consider the following function definition:

```
let f =
    let pi = 3.14159
    let twoPi = 2.0 * pi
    twoPi;;
```

Note that F#, like ML, does not coerce numeric values, so if this function used `2` in the second-last line, rather than `2.0`, an error would be reported.

If a function is recursive, the reserved word **`rec`** must precede its name in its definition. Following is an F# version of factorial:

```
let rec factorial x =
    if x <= 1  then 1
    else n * factorial(n - 1)
```

Names defined in functions can be outscoped, which means they can be

redefined, which ends their former scope. For example, we could have the following:

```
let x4 x =
    let x = x * x
    let x = x * x
    x;;
```

In this function, the first **let** in the body of the x4 function creates a new version of x, defining it to have the value of the square of the parameter x. This terminates the scope of the parameter. So, the second **let** in the function body uses the new x in its right side and creates yet another version of x, thereby terminating the scope of the x created in the previous **let**.

There are two important functional operators in F#, pipeline (|>) and function composition (>>). The pipeline operator is a binary operator that sends the value of its left operand, which is an expression, to the last parameter of the function call, which is the right operand. It is used to chain together function calls while flowing the data being processed to each call. Consider the following example code, which uses the high-order functions filter and map:

```
let myNums = [1; 2; 3; 4; 5]
let evensTimesFive = myNums
    |> List.filter (fun n -> n % 2 = 0)
    |> List.map (fun n -> 5 * n)
```

The evensTimesFive function begins with the list myNums, filters out the numbers that are not even with filter, and uses map to map a lambda expression that multiplies the numbers in a given list by five. The return value of evensTimesFive is [10; 20].

The function composition operator builds a function that applies its left operand to a given parameter, which is a function, and then passes the result returned from that function to its right operand, which is also a function. So, the F# expression (f >> g)x is equivalent to the mathematical expression g(f(x)).

Like ML, F# supports curried functions and partial evaluation. The ML example in Section 15.7 could be written in F# as follows:

```
let add a b = a + b;;
let add5 = add 5;;
```

Note that, unlike ML, the syntax of the formal parameter list in F# is the same for all functions, so all functions with more than one parameter can be curried.

F# is interesting for several reasons: First, it builds on the past functional languages as a functional language. Second, it supports virtually all programming methodologies in widespread use today. Third, it is the first functional language that is designed for interoperability with other widely used languages. Fourth, it starts out with an elaborate and well-developed IDE and library of utility software with .NET and its framework.

15.10 Support for Functional Programming in Primarily Imperative Languages

Imperative programming languages have typically provided only limited support for functional programming. That limited support has resulted in little use of those languages for functional programming. The most important restriction, related to functional programming, of imperative languages of the past was the lack of support for higher-order functions.

One indication of the increasing interest and use of functional programming is the partial support for it that has begun to appear over the last decade in programming languages that are primarily imperative. For example, anonymous functions, which are like lambda expressions, are now part of JavaScript, Python, Ruby, Java, and C#.

In JavaScript, named functions are defined with the following syntax:

```
function name (formal-parameters) {
 body
}
```

An anonymous function is defined in JavaScript with the same syntax, except that the name of the function is omitted.

In C#, a lambda expression is an instance of a delegate. They can be anonymous or named. An anonymous lambda expression is simpler than an anonymous method because methods must define their parameter types and return type, but lambda expressions use the C# inferencing process to avoid those necessities. The syntax of an unnamed lambda expression in C# is as follows:

```
parameter(s) => expression
```

If there is more than one parameter, they must be enclosed in parentheses. If there are no parameters, empty parentheses must appear in the place of the parameters. If the system cannot infer the types of the parameters, they can be preceded by type names. The type of the return value is never specified; it is always inferred by the context of the lambda expression. The expression is either a single expression or a compound statement enclosed in braces. Such a compound statement must include a **`return`** statement.

One common use of anonymous lambda expressions is as actual parameters to methods that are specified to take delegates as parameters. For example, C# has a collection of methods for arrays that perform searching operations. For example, the `FindAll` method finds all of the elements of an array that satisfy a given instance of a delegate that takes a parameter of the type of the array's elements and returns a Boolean value. For example, we could have the following:

```
int[] numbers = {-3, 0, 4, 5, 1, 7, -3, -6, -9, 0, 3};
int[] positives = Array.FindAll(numbers, n => n > 0);
// Now, positives is {4, 5, 1, 7, 3}
```

Lambda expressions in C# can also be named. The language has generic delegates that make defining such lambda expressions simple, although they do not cover all possibilities. One commonly used generic delegate is `Func`, which can take up to sixteen generic parameters for the parameters of the lambda expression, plus one more for the return type. For example, consider the following example of a named lambda expression and an invocation of it:

```
Func<int, int, int> eval1 = (a, b) => 3 * a + (b / 2);
int result = eval1(6, 22);
```

C# lambda expressions can access variables defined outside their definitions. When they do, the lifetimes of the accessed external variables, which are called **captured variables**, are extended so that they still exist when the lambda expression is used. A lambda expression that captures outside variables is a closure.

Lambda expressions were added to Java 8. The general syntax of these expressions is like that of C#, except that `->` is used instead of `=>`. The syntax of parameters, the inferencing of parameter types and the return type, and the

expression or block with a **`return`** are the same as with C#. Prior to Java 8, there was no convenient way to pass a block of code to a method or have the method return a block of code.14

14. One inconvenient way was to define a class with a method that contained the code, instantiate the class, and pass a reference to it.

Python's lambda expressions define simple one-statement anonymous functions. The syntax of a lambda expression in Python is exemplified by the following:

```
lambda a, b : 2 * a - b
```

Note that the formal parameters are separated from function body by a colon.

Python includes the higher-order functions `filter` and `map`. Both often use lambda expressions as their first parameter. The second parameter of these is a sequence type, and both return the same sequence type as their second parameter. In Python, strings, lists, and tuples are considered sequences. Consider the following example of using the map function in Python:

```
map(lambda x: x ** 3, [2, 4, 6, 8])
```

This call returns `[8, 64, 216, 512]`.

Python also supports partial function applications. Consider the following example:

```
from operator import add
add5 = partial (add, 5)
```

The **`from`** declaration here imports the functional version of the addition operator, which is named `add`, from the `operator` module.

After defining `add5`, it can be used with one parameter, as in the following:

```
add5(15)
```

This call returns `20`.

As described in Chapter 6, Python includes lists and list comprehensions.

Ruby's blocks are effectively subprograms that are sent to methods, which makes the method a higher-order subprogram. A Ruby block can be converted to a subprogram object with **`lambda`**. For example, consider the following:

```
times = lambda {|a, b| a * b}
```

Following is an example of using `times`:

```
x = times.(3, 4)
```

This sets `x` to `12`. The `times` object can be curried with the following:

```
times5 = times.curry.(5)
```

This function can be used as in the following:

```
x5 = times5.(3)
```

This sets `x5` to `15`.

15.11 A Comparison of Functional and Imperative Languages

This section discusses some of the differences between imperative and functional languages.

Functional languages can have a very simple syntactic structure. The list structure of Lisp, which is used for both code and data, clearly illustrates this. The syntax of the imperative languages is much more complex. This makes them more difficult to learn and to use.

The semantics of functional languages is also simpler than that of the imperative languages. For example, in the denotational semantics description of an imperative loop construct given in Section 3.5.2, the loop is converted from an iterative construct to a recursive construct. This conversion is unnecessary in a pure functional language, in which there is no iteration. Furthermore, we assumed there were no expression side effects in all of the denotational semantic descriptions of imperative constructs in Chapter 3. This restriction is unrealistic for imperative languages, because all of the C-based languages include expression side effects. This restriction is not needed for the denotational descriptions of pure functional languages.

Some in the functional programming community have claimed that the use of functional programming results in an order-of-magnitude increase in productivity, largely due to functional programs being claimed to be only 10 percent as large as their imperative counterparts. While such numbers have been actually shown for certain problem areas, for other problem areas, functional programs are more like 25 percent as large as imperative solutions to the same problems (Wadler, 1998). These factors allow proponents of functional programming to claim productivity advantages over imperative programming of 4 to 10 times. However, program size alone is not necessarily a good measure of productivity. Certainly not all lines of source code have equal complexity, nor do they take the same amount of time to produce. In fact, because of the necessity of dealing with variables,

imperative programs have many trivially simple lines for initializing and making small changes to variables.

Execution efficiency is another basis for comparison. When functional programs are interpreted, they are of course much slower than their compiled imperative counterparts. However, there are now compilers for most functional languages, so that execution speed disparities between functional languages and compiled imperative languages are no longer so great. One might be tempted to say that because functional programs are significantly smaller than equivalent imperative programs, they should execute much faster than the imperative programs. However, this often is not the case, because of a collection of language characteristics of the functional languages, such as lazy evaluation, that have a negative impact on execution efficiency. Considering the relative efficiency of functional and imperative programs, it is reasonable to estimate that an average functional program will execute in about twice the time of its imperative counterpart (Wadler, 1998). This may sound like a significant difference, one that would often lead one to dismiss the functional languages for a given application. However, this factor-of-two difference is important only in situations where execution speed is of the utmost importance. There are many situations where a factor of two in execution speed is not considered important. For example, consider that many programs written in imperative languages, such as the Web software written in JavaScript and PHP, are interpreted and therefore are much slower than equivalent compiled versions. For these applications, execution speed is not the first priority.

Another source of the difference in execution efficiency between functional and imperative programs is the fact that imperative languages were designed to run efficiently on von Neumann architecture computers, while the design of functional languages is based on mathematical functions. This gives the imperative languages a large advantage.

Functional languages have a potential advantage in readability. In many imperative programs, the details of dealing with variables obscure the logic of the program. Consider a function that computes the sum of the cubes of the first n positive integers. In C, such a function would likely appear similar to the following:

```
int sum_cubes(int  n){
  int sum = 0;
  for(int index = 1; index <= n; index++)
    sum += index * index * index;
  return sum;
}
```

In Haskell, the function could be:

```
sumCubes n = sum (map (^3) [1..n])
```

This version simply specifies three steps:

1. Build the list of numbers (`[1..n]`).

2. Create a new list by mapping a function that computes the cube of a number onto each number in the list.

3. Sum the new list.

Because of the lack of details of variables and iteration control, this version is more readable than the C version.15

15. Of course, the C version could have been written in a more functional style, but most C programmers probably would not write it that way.

Concurrent execution in the imperative languages is difficult to design and difficult to use, as we saw in Chapter 13. In an imperative language, the programmer must make a static division of the program into its concurrent parts, which are then written as tasks, whose execution often must be synchronized. This can be a complicated process. Programs in functional languages are naturally divided into functions. In a pure functional language, these functions are independent in the sense that they do not create side effects and their operations do not depend on any nonlocal or global variables. Therefore, it is much easier to determine which of them can be concurrently executed. The actual parameter expressions in calls often can be evaluated concurrently. Simply by specifying that it can be done, a function can be implicitly evaluated in a separate thread, as in Multilisp. And, of course, access to shared immutable data does not require synchronization.

One simple factor that strongly affects the complexity of imperative, or procedural programming, is the necessary attention of the programmer to the state of the program at each step of its development. In a large program, the state of the program is a large number of values (for the large number of program variables). In pure functional programming, there is no state; hence, no need to devote attention to keeping it in mind.

It is not a simple matter to determine precisely why functional languages have not attained greater popularity. The inefficiency of the early implementations was clearly a factor then, and it is likely that at least some contemporary imperative programmers still believe that programs written in functional languages are slow. In addition, the vast majority of programmers learn programming using imperative languages, which makes functional programs appear to them to be strange and difficult to understand. For many who are comfortable with imperative programming, the switch to functional programming is an unattractive and potentially difficult move. On the other hand, those who begin with a functional language never notice anything strange about functional programs.

SUMMARY

Mathematical functions are named or unnamed mappings that use only conditional expressions and recursion to control their evaluations. Complex functions can be defined using higher-order functions or functional forms, in which functions are used as parameters, returned values, or both.

Functional programming languages are modeled on mathematical functions. In their pure form, they do not use variables or assignment statements to produce results; rather, they use function applications, conditional expressions, and recursion for execution control and functional forms to construct complex functions. Lisp began as a purely functional language but soon acquired a number of imperative-language features added in order to increase its efficiency and ease of use.

The first version of Lisp grew out of the need for a list-processing language for AI applications. Lisp is still the most widely used language for that application area.

The first implementation of Lisp was serendipitous: The original version of EVAL was developed solely to demonstrate that a universal Lisp function could be written.

Because Lisp data and Lisp programs have the same form, it is possible to have a program build another program. The availability of EVAL allows dynamically constructed programs to be executed immediately.

Scheme is a relatively simple dialect of Lisp that uses static scoping exclusively. Like Lisp, Scheme's primary primitives include functions for constructing and dismantling lists, functions for conditional expressions, and simple predicates for numbers, symbols, and lists.

Common Lisp is a Lisp-based language that was designed to include most of the features of the Lisp dialects of the early 1980s. It allows both static- and dynamic-scoped variables and includes many imperative features. Common

Lisp uses macros to define some of its functions. Users are allowed to define their own macros. The language includes reader macros, which are also user definable. Reader macros define single-symbol macros.

ML is a static-scoped and strongly typed functional programming language that uses a syntax that is more closely related to that of an imperative language than to Lisp. It includes a type-inferencing system, exception handling, a variety of data structures, and abstract data types.

ML does not do any type coercions and does not allow function overloading. Multiple definitions of functions can be defined using pattern matching of the actual parameter form. Currying is the process of replacing a function that takes multiple parameters with one that takes a single parameter and returns a function that takes the other parameters. ML, as well as several other functional languages, supports currying.

Haskell is similar to ML, except that all expressions in Haskell are evaluated using a lazy method, which allows programs to deal with infinite lists. Haskell also supports list comprehensions, which provide a convenient and familiar syntax for describing sets. Unlike ML and Scheme, Haskell is a pure functional language.

F# is a .NET programming language that supports functional and imperative programming, including object-oriented programming. Its functional programming core is based on OCaml, a descendent of ML and Haskell. F# is supported by an elaborate and widely used IDE. It also interoperates with other .NET languages and has access to the .NET class library.

BIBLIOGRAPHIC NOTES

- The first published version of Lisp can be found in McCarthy (1960). A widely used version from the mid-1960s until the late 1970s is described in McCarthy et al. (1965) and Weissman (1967). Common Lisp is described in Steele (1990). The Scheme language is described in Dybvig (2011). ML is defined in Milner et al. (1997). Ullman (1998) is an excellent introductory textbook for ML. Programming in Haskell is introduced in Thompson (1999). F# is described in Syme et al. (2010).

- The Scheme programs in this chapter were developed using DrRacket's legacy language R5RS.

- A rigorous discussion of functional programming in general can be found in Henderson (1980). The process of implementing functional languages through graph reduction is discussed in detail in Peyton Jones (1987).

REVIEW QUESTIONS

1. Define *functional form*, *simple list*, *bound variable*, and *referential transparency*.
2. What does a lambda expression specify?
3. What data types were parts of the original Lisp?
4. In what common data structure are Lisp lists normally stored?
5. Explain why `QUOTE` is needed for a parameter that is a data list.
6. What is a simple list?
7. What does the abbreviation REPL stand for?
8. What are the three parameters to `IF`?
9. What are the differences between `=`, `EQ?`, `EQV?`, and `EQUAL`?
10. What are the differences between the evaluation method used for the Scheme special form `DEFINE` and that used for its primitive functions?
11. What are the two forms of `DEFINE`?
12. Describe the syntax and semantics of `COND`.
13. Why are `CAR` and `CDR` so named?
14. If `CONS` is called with two atoms, say `'A` and `'B`, what is the returned?
15. Describe the syntax and semantics of `LET` in Scheme.
16. What are the differences between `CONS`, `LIST`, and `APPEND`?
17. Describe the syntax and semantics of `mapcar` in Scheme.

18. What is tail recursion? Why is it important to define functions that use recursion to specify repetition to be tail recursive?

19. Why were imperative features added to most dialects of Lisp?

20. In what ways are Common Lisp and Scheme opposites?

21. What scoping rule is used in Scheme? In Common Lisp? In ML? In Haskell? In F#?

22. What happens during the reader phase of a Common Lisp language processor?

23. What are two ways that ML is fundamentally different from Scheme?

24. What is stored in an ML evaluation environment?

25. What is the difference between an ML **`val`** statement and an assignment statement in C?

26. What is type inferencing, as used in ML?

27. What is the use of the **`fn`** reserved word in ML?

28. Can ML functions that deal with scalar numerics be generic?

29. What is a curried function?

30. What does partial evaluation mean?

31. Describe the actions of the ML `filter` function.

32. What operator does ML use for Scheme's CAR?

33. What operator does ML use for functional composition?

34. What are the three characteristics of Haskell that make it different from ML?

35. What does lazy evaluation mean?

36. What is a strict programming language?

37. What programming paradigms are supported by F#?

38. With what other programming languages can F# interoperate?

39. What does F#'s `let` do?

40. How is the scope of a F# `let` construct terminated?

41. What is the underlying difference between a sequence and a list in F#?

42. What is the difference between the let of ML and that of F#, in terms of extent?

43. What is the syntax of a lambda expression in F#?

44. Does F# coerce numeric values in expressions? Argue in support of the design choice.

45. What support does Python provide for functional programming?

46. What function in Ruby is used to create a curried function?

47. Is the use of functional programming expanding or shrinking?

48. What is one characteristic of functional programming languages that makes their semantics simpler than that of imperative languages?

49. What is the flaw in using lines of code to compare the productivity of functional languages and that of imperative languages?

50. Why can concurrency be easier with functional languages than imperative languages?

PROBLEM SET

1. Read John Backus's paper on FP (Backus, 1978) and compare the features of Scheme discussed in this chapter with the corresponding features of FP.

2. Find definitions of the Scheme functions `EVAL` and `APPLY`, and explain their actions.

3. One of the most modern and complete programming environments is the INTERLISP system for Lisp, as described in "The INTERLISP Programming Environment," by Teitelmen and Masinter (*IEEE Computer*, Vol. 14, No. 4, April 1981). Read this article carefully and compare the difficulty of writing Lisp programs on your system with that of using INTERLISP (assuming that you do not normally use INTERLISP).

4. Refer to a book on Lisp programming and determine what arguments support the inclusion of the `PROG` feature in Lisp.

5. Find at least one example of a typed functional programming language being used to build a commercial system in each of the following areas: database processing, financial modeling, statistical analysis, and bioinformatics.

6. A functional language could use some data structure other than the list. For example, it could use strings of single-character symbols. What primitives would such a language have in place of the `CAR`, `CDR`, and `CONS` primitives of Scheme?

7. Make a list of the features of F# that are not in ML.

8. If Scheme were a pure functional language, could it include `DISPLAY`? Why or why not?

9. What does the following Scheme function do?

```
(define (y s lis)
  (cond
    ((null? lis) '() )
    ((equal? s (car lis)) lis)
    (else (y s (cdr lis)))
))
```

10. What does the following Scheme function do?

```
(define (x lis)
  (cond
    ((null? lis) 0)
    ((not (list? (car lis)))
       (cond
         ((eq? (car lis) #f) (x (cdr lis)))
         (else (+ 1 (x (cdr lis))))))
    (else (+ (x (car lis)) (x (cdr lis))))
```

PROGRAMMING EXERCISES

1. Write a Scheme function that computes the volume of a sphere, given its radius.

2. Write a Scheme function that computes the real roots of a given quadratic equation. If the roots are complex, the function must display a message indicating that. This function must use an `IF` function. The three parameters to the function are the three coefficients of the quadratic equation.

3. Repeat Programming Exercise 2 using a `COND` function, rather than an `IF` function.

4. Write a Scheme function that takes two numeric parameters, `A` and `B`, and returns `A` raised to the `B` power.

5. Write a Scheme function that returns the number of zeros in a given simple list of numbers.

6. Write a Scheme function that takes a simple list of numbers as a parameter and returns a list with the largest and smallest numbers in the input list.

7. Write a Scheme function that takes a list and an atom as parameters and returns a list identical to its parameter list except with all top-level instances of the given atom deleted.

8. Write a Scheme function that takes a list as a parameter and returns a list identical to the parameter except the last element has been deleted.

9. Repeat Programming Exercise 7, except that the atom can be either an atom or a list.

10. Write a Scheme function that takes two atoms and a list as parameters and returns a list identical to the parameter list except all occurrences of

the first given atom in the list are replaced with the second given atom, no matter how deeply the first atom is nested.

11. Write a Scheme function that returns the reverse of its simple list parameter.

12. Write a Scheme predicate function that tests for the structural equality of two given lists. Two lists are structurally equal if they have the same list structure, although their atoms may be different.

13. Write a Scheme function that returns the union of two simple list parameters that represent sets.

14. Write a Scheme function with two parameters, an atom and a list, that returns a list identical to the parameter list except with all occurrences, no matter how deep, of the given atom deleted. The returned list cannot contain anything in place of the deleted atoms.

15. Write a Scheme function that takes a list as a parameter and returns a list identical to the parameter list except with the second top-level element removed. If the given list does not have two elements, the function should return `()`.

16. Write a Scheme function that takes a simple list of numbers as its parameter and returns a list identical to the parameter list except with the numbers in ascending order.

17. Write a Scheme function that takes a simple list of numbers as its parameter and returns the largest and smallest numbers in the list.

18. Write a Scheme function that takes a simple list as its parameter and returns a list of all permutations of the given list.

19. Write the quicksort algorithm in Scheme.

20. Rewrite the following Scheme function as a tail-recursive function:

```
(DEFINE (doit n)
  (IF (= n 0)
```

```
        0
        (+ n (doit (- n 1)))
    ))
```

21. Write any of the first 19 Programming Exercises in F#.

22. Write any of the first 19 Programming Exercises in ML.

16 Logic Programming Languages

The objectives of this chapter are to introduce the concepts of logic programming and logic programming languages, including a brief description of a subset of Prolog. We begin with an introduction to predicate calculus, which is the basis for logic programming languages. This is followed by a discussion of how predicate calculus can be used for automatic theorem-proving systems. Then, we present a general overview of logic programming. Next, a lengthy section introduces the basics of the Prolog programming language, including arithmetic, list processing, and a trace tool that can be used to help debug programs and also to illustrate how the Prolog system works. The final two sections describe some of the problems of Prolog as a logic language and some of the application areas in which Prolog has been used.

16.1 Introduction

Chapter 15 discusses the functional programming paradigm, which is significantly different from the software development methodologies used with the imperative languages. In this chapter, we describe another different programming methodology. In this case, the approach is to express programs in a form of symbolic logic and use a logical inferencing process to produce results. Logic programs are declarative rather than procedural, which means that only the specifications of the desired results are stated rather than detailed procedures for producing them. Programs in logic programming languages are collections of facts and rules. Such a program is used by asking it questions, which it attempts to answer by consulting the facts and rules. "Consulting" is perhaps misleading here, for the process is far more complex than that word connotes. This approach to problem solving may sound like it addresses only a very narrow category of problems, but it is more flexible than might be thought.

Programming that uses a form of symbolic logic as a programming language is often called **logic programming**, and languages based on symbolic logic are called **logic programming languages**, or **declarative languages**. We have chosen to describe the logic programming language Prolog, because it is the only widely used logic language.

The syntax of logic programming languages is remarkably different from that of the imperative and functional languages. The semantics of logic programs also bears little resemblance to that of imperative-language programs. These observations should lead the reader to some curiosity about the nature of logic programming and declarative languages.

16.2 A Brief Introduction to Predicate Calculus

Before we can discuss logic programming, we must briefly investigate its basis, which is formal logic. This is not our first contact with formal logic in this text; it was used extensively in the axiomatic semantics described in Chapter 3.

A **proposition** can be thought of as a logical statement that may or may not be true. It consists of objects and the relationships among objects. Formal logic was developed to provide a method for describing propositions, with the goal of allowing those formally stated propositions to be checked for validity.

Symbolic logic can be used for the three basic needs of formal logic: to express propositions, to express the relationships between propositions, and to describe how new propositions can be inferred from other propositions that are assumed to be true.

There is a close relationship between formal logic and mathematics. In fact, much of mathematics can be thought of in terms of logic. The fundamental axioms of number and set theory are the initial set of propositions, which are assumed to be true. Theorems are the additional propositions that can be inferred from the initial set.

The particular form of symbolic logic that is used for logic programming is called **first-order predicate calculus** (though it is a bit imprecise, we will usually refer to it as *predicate calculus*). In the following subsections, we present a brief look at predicate calculus. Our goal is to lay the groundwork for a discussion of logic programming and the logic programming language Prolog.

16.2.1 Propositions

The objects in logic programming propositions are represented by simple terms, which are either constants or variables. A constant is a symbol that represents an object. A variable is a symbol that can represent different objects at different times, although in a sense that is far closer to mathematics than the variables in an imperative programming language.

The simplest propositions, which are called **atomic propositions**, consist of compound terms. A **compound term** is one element of a mathematical relation, written in a form that has the appearance of mathematical function notation. Recall from Chapter 15 that a mathematical function is a mapping, which can be represented either as an expression or as a table or list of tuples. Compound terms are elements of the tabular definition of a function.

A compound term is composed of two parts: a **functor**, which is the function symbol that names the relation, and an ordered list of parameters, which together represent an element of the relation. A compound term with a single parameter is a 1-tuple; one with two parameters is a 2-tuple, and so forth. For example, we might have the two propositions

- man(jake)
- like(bob, steak)

which state that {jake} is a 1-tuple in the relation named man, and that {bob, steak} is a 2-tuple in the relation named like. If we added the proposition

- man(fred)

to the two previous propositions, then the relation man would have two distinct elements, {jake} and {fred}. All of the simple terms in these propositions—man, jake, like, bob, and steak—are constants. Note that these propositions have no intrinsic semantics. They mean whatever we want them to mean. For example, the second example may mean that bob likes steak, or that steak likes bob, or that bob is in some way similar to a steak.

Propositions can be stated in two modes: one in which the proposition is defined to be true, and one in which the truth of the proposition is something that is to be determined. In other words, propositions either can be facts or

queries. The example propositions above could be either.

Compound propositions have two or more atomic propositions, which are connected by logical connectors, or operators, in the same way compound logic expressions are constructed in imperative languages. The names, symbols, and meanings of the predicate calculus logical connectors are as follows:

Name	*Symbol*	*Example*	*Meaning*
negation	⊐	⊐ a	not *a*
conjunction	⋂	*a* ⋂ *b*	*a* and *b*
disjunction	⋃	*a* ⋃ *b*	*a* or *b*
equivalence	≡	*a* ≡ *b*	*a* is equivalent to *b*
implication	⊃	*a* ⊃ *b*	*a* implies *b*
	⊂	*a* ⊂ *b*	*b* implies *a*

The following are examples of compound propositions:

a ∩ b ⊃ c a ∩⊐b⊃ d

The ⊐ operator has the highest precedence. The operators ∩, ∪, and ≡ all have higher precedence than ⊃ and ⊂. So, the second example above is equivalent to

(a ∩(⊐ b)) ⊃ d

Variables can appear in propositions but only when introduced by special symbols called *quantifiers*. Predicate calculus includes two quantifiers, as described below, where *X* is a variable and *P* is a proposition:

Name	*Example*	*Meaning*
universal	∀ *X.P*	For all *X*, *P* is true.
existential	∃ *X.P*	There exists a value of *X* such that *P* is true.

The period between *X* and *P* simply separates the variable from the proposition. For example, consider the following:

- $\forall$ X. (woman (*X*) $\supset$ human (*X*))
- $\exists$ X. (mother (mary, *X*) $\cap$ male (*X*))

The first of these propositions means that for any value of *X*, if *X* is a woman, then *X* is a human. The second means that there exists a value of *X* such that mary is the mother of *X* and *X* is a male; in other words, mary has a son. The scope of the universal and existential quantifiers is the atomic propositions to which they are attached. This scope can be extended using parentheses, as in the two compound propositions just described. So, the universal and existential quantifiers have higher precedence than any of the operators.

16.2.2 Clausal Form

We are discussing predicate calculus because it is the basis for logic programming languages. As with other languages, logic languages are best in their simplest form, meaning that redundancy should be minimized.

One problem with predicate calculus as we have described it thus far is that there are too many different ways of stating propositions that have the same meaning; that is, there is a great deal of redundancy. This is not such a problem for logicians, but if predicate calculus is to be used in an automated (computerized) system, it is a serious problem. To simplify matters, a standard form for propositions is desirable. Clausal form, which is a relatively simple form of propositions, is one such standard form. All propositions can be expressed in clausal form. A proposition in clausal form has the following general syntax:

B1 $\cup$ B2 $\cup$ … $\cup$ Bn $\subset$ A1 $\cap$ A2 $\cap$ … $\cap$ Am

in which the *A*'s and *B*'s are terms. The meaning of this clausal form proposition is as follows: If all of the *A*'s are true, then at least one *B* is true. The primary characteristics of clausal form propositions are the following:

Existential quantifiers are not required; universal quantifiers are implicit in the use of variables in the atomic propositions; and no operators other than conjunction and disjunction are required. Also, conjunction and disjunction need appear only in the order shown in the general clausal form: disjunction on the left side and conjunction on the right side. All predicate calculus propositions can be algorithmically converted to clausal form. Nilsson (1971) gives proof that this can be done, as well as a simple conversion algorithm for doing it.

The right side of a clausal form proposition is called the **antecedent**. The left side is called the **consequent** because it is the consequence of the truth of the antecedent. As examples of clausal form propositions, consider the following:

- likes (bob, trout) $\subset$ likes (bob, fish) $\cap$ fish (trout)
- father (louis, al) $\cup$ father (louis, violet) $\subset$
- father (al, bob) $\cap$ mother (violet, bob) $\cap$ grandfather (louis, bob)

The English version of the first of these states that if bob likes fish and a trout is a fish, then bob likes trout. The second states that if al is bob's father and violet is bob's mother and louis is bob's grandfather, then louis is either al's father or violet's father.

16.3 Predicate Calculus and Proving Theorems

Predicate calculus provides a method of expressing collections of propositions. One use of collections of propositions is to determine whether any interesting or useful facts can be inferred from them. This is exactly analogous to the work of mathematicians, who strive to discover new theorems that can be inferred from known axioms and theorems.

The early days of computer science (the 1950s and early 1960s) saw a great deal of interest in automating the theorem-proving process. One of the most significant breakthroughs in automatic theorem proving was the discovery of the resolution principle by Alan Robinson (1965) at Syracuse University.

Resolution is an inference rule that allows inferred propositions to be computed from given propositions, thus providing a method with potential application to automatic theorem proving. Resolution was devised to be applied to propositions in clausal form. The concept of resolution is the following: Suppose there are two propositions with the forms

$P1 \subset P2$ $Q1 \subset Q2$

Their meaning is that P2 implies P1 and Q2 implies Q1. Furthermore, suppose that P1 is identical to Q2, so that we could rename P1 and Q2 as *T*. Then, we could rewrite the two propositions as

$T \subset P2$ $Q1 \subset T$

Now, because P2 implies *T* and *T* implies Q1, it is logically obvious that P2 implies Q1, which we could write as

$Q1 \subset P2$

The process of inferring this proposition from the original two propositions is resolution.

As another example, consider the two propositions:

- older (joanne, jake) $\subset$ mother (joanne, jake)
- wiser (joanne, jake) $\subset$ older (joanne, jake)

From these propositions, the following proposition can be constructed using resolution:

- wiser(joanne, jake) $\subset$ mother(joanne, jake)

The mechanics of this resolution construction are simple: The terms of the left sides of the two clausal propositions are OR'd together to make the left side of the new proposition. Then the right sides of the two clausal propositions are AND'd together to get the right side of the new proposition. Next, any term that appears on both sides of the new proposition is removed from both sides. The process is exactly the same when the propositions have multiple terms on either or both sides. The left side of the new inferred proposition initially contains all of the terms of the left sides of the two given propositions. The new right side is similarly constructed. Then the term that appears on both sides of the new proposition is removed. For example, if we have

- father(bob, jake) $\cup$ mother(bob, jake) $\subset$ parent(bob, jake)
- grandfather(bob, fred) $\subset$ father(bob, jake) $\cap$ father(jake, fred)

resolution says that

- mother(bob, jake) $\cup$ grandfather(bob, fred) $\subset$
- parent(bob, jake) $\cap$ father(jake, fred)

which has all but one of the atomic propositions of both of the original propositions. The one atomic proposition that allowed the operation father (bob, jake) in the left side of the first and in the right side of the second is left out. In English, we would say

if: bob is the parent of jake implies that bob is either the father or mother of jake

and: bob is the father of jake and jake is the father of fred implies that bob is the grandfather of fred

then: *if* bob is the parent of jake and jake is the father of fred *then:* either bob is jake's mother or bob is fred's grandfather

Resolution is actually more complex than these simple examples illustrate. In particular, the presence of variables in propositions requires resolution to find values for those variables that allow the matching process to succeed. This process of determining useful values for variables is called **unification**. The temporary assigning of values to variables to allow unification is called **instantiation**.

It is common for the resolution process to instantiate a variable with a value, fail to complete the required matching, and then be required to backtrack and instantiate the variable with a different value. We will discuss unification and backtracking more extensively in the context of Prolog.

A critically important property of resolution is its ability to detect any inconsistency in a given set of propositions. This is based on the formal property of resolution called **refutation complete**. What this means is that given a set of inconsistent propositions, resolution can prove them to be inconsistent. This allows resolution to be used to prove theorems, which can be done as follows: We can envision a theorem proof in terms of predicate calculus as a given set of pertinent propositions, with the negation of the theorem itself stated as a new proposition. The theorem is negated so that resolution can be used to prove the theorem by finding an inconsistency. This is proof by contradiction, a frequently used approach to proving theorems in mathematics. Typically, the original propositions are called the **hypotheses**, and the negation of the theorem is called the **goal**.

Theoretically, this process is valid and useful. The time required for resolution, however, can be a problem. Although resolution is a finite process when the set of propositions is finite, the time required to find an inconsistency in a large database of propositions may be huge.

Theorem proving is the basis for logic programming. Much of what is

computed can be couched in the form of a list of given facts and relationships as hypotheses, and a goal to be inferred from the hypotheses, using resolution.

Resolution on a hypotheses and a goal that are general propositions, even if they are in clausal form, is often not practical. Although it may be possible to prove a theorem using clausal form propositions, it may not happen in a reasonable amount of time. One way to simplify the resolution process is to restrict the form of the propositions. One useful restriction is to require the propositions to be Horn clauses. **Horn clauses** only can be in one of two forms: They have either a single atomic proposition on the left side or an empty left side.[1] The left side of a clausal form proposition is sometimes called the head, and Horn clauses with left sides are called headed Horn clauses. Headed Horn clauses are used to state relationships, such as

1. Horn clauses are named after Alfred Horn (1951), who studied clauses in this form.

- likes(bob, trout) ⊂ likes(bob, fish) ∩ fish(trout)

Horn clauses with empty left sides, which are often used to state facts, are called *headless Horn clauses*. For example,

- father(bob, jake)

Most, but not all, propositions can be stated as Horn clauses. The restriction to Horn clauses makes resolution a practical process for proving theorems.

16.4 An Overview of Logic Programming

Languages used for logic programming are called *declarative languages*, because programs written in them consist of declarations rather than assignments and control flow statements. These declarations are actually statements, or propositions, in symbolic logic.

One of the essential characteristics of logic programming languages is their semantics, which is called **declarative semantics**. The basic concept of this semantics is that there is a simple way to determine the meaning of each statement, and it does not depend on how the statement might be used to solve a problem. Declarative semantics is considerably simpler than the semantics of the imperative languages. For example, the meaning of a given proposition in a logic programming language can be concisely determined from the statement itself. In an imperative language, the semantics of a simple assignment statement requires examination of local declarations, knowledge of the scoping rules of the language, and possibly even examination of programs in other files just to determine the types of the variables in the assignment statement. Then, assuming the expression of the assignment contains variables, the execution of the program prior to the assignment statement must be traced to determine the values of those variables. The resulting action of the statement, then, depends on its run-time context. Comparing this semantics with that of a proposition in a logic language, with no need to consider textual context or execution sequences, it is clear that declarative semantics is far simpler than the semantics of imperative languages. Thus, declarative semantics is often stated as one of the advantages that declarative languages have over imperative languages (Hogger, 1984, pp. 240–241).

Programming in both imperative and functional languages is primarily procedural, which means that the programmer knows *what* is to be accomplished by a program and instructs the computer on exactly *how* the computation is to be done. In other words, the computer is treated as a simple

device that obeys orders. Everything that is computed must have every detail of that computation spelled out. Some believe that this is the essence of the difficulty of programming using imperative and functional languages.

Programming in a logic programming language is nonprocedural. Programs in such languages do not state exactly *how* a result is to be computed but rather describe the form of the result. The difference is that we assume the computer system can somehow determine *how* the result is to be computed. What is needed to provide this capability for logic programming languages is a concise means of supplying the computer with both the relevant information and a method of inference for computing desired results. Predicate calculus supplies the basic form of communication to the computer, and resolution provides the inference technique.

Sorting is commonly used to illustrate the difference between procedural and nonprocedural systems. In a language like Java, sorting is done by explaining in a Java program all of the details of some sorting algorithm to a computer that has a Java compiler. The computer, after translating the Java program into machine code or some interpretive intermediate code, follows the instructions and produces the sorted list.

In a nonprocedural language, it is necessary only to describe the characteristics of the sorted list: It is some permutation of the given list such that for each pair of adjacent elements, a given relationship holds between the two elements. To state this formally, suppose the list to be sorted is in an array named list that has a subscript range 1 . . . *n*. The concept of sorting the elements of the given list, named old_list, and placing them in a separate array, named new_list, can then be expressed as follows:

- sort(old_list, new_list) $\subset$ permute(old_list, new_list) $\cap$ sorted(new_list)
- sorted (list) $\subset \forall$ j such that 11$\leq$j<n, list(j) $\leq$list(j+1)

where permute is a predicate that returns true if its second parameter array is a permutation of its first parameter array.

From this description, the nonprocedural language system could produce the sorted list. That makes nonprocedural programming sound like the mere

production of concise software requirements specifications, which is a fair assessment. Unfortunately, however, it is not that simple. Logic programs that use only resolution face serious problems of execution efficiency. In our example of sorting, if the list is long, the number of permutations is huge, and they must be generated and tested, one by one, until the one that is in order is found—a very lengthy process. Of course, one must consider the possibility that the best form of a logic language may not yet have been determined, and good methods of creating programs in logic programming languages for large problems have not yet been developed.

16.5 The Origins of Prolog

As was stated in Chapter 2, Alain Colmerauer and Phillippe Roussel at the University of Aix-Marseille, with some assistance from Robert Kowalski at the University of Edinburgh, developed the fundamental design of Prolog. Colmerauer and Roussel were interested in natural-language processing; Kowalski was interested in automated theorem proving. The collaboration between the University of Aix-Marseille and the University of Edinburgh continued until the mid-1970s. Since then, research on the development and use of the language has progressed independently at those two locations, resulting in, among other things, two syntactically different dialects of Prolog.

The development of Prolog and other research efforts in logic programming received limited attention outside of Edinburgh and Marseille until the announcement in 1981 that the Japanese government was launching a large research project called the Fifth Generation Computing Systems (FGCS; Fuchi, 1981; Moto-oka, 1981). One of the primary objectives of the project was to develop intelligent machines, and Prolog was chosen as the basis for this effort. The announcement of FGCS aroused a sudden strong interest in artificial intelligence and logic programming in researchers and the governments of the United States and several European countries.

After a decade of effort, the FGCS project was quietly dropped. Despite the great assumed potential of logic programming and Prolog, little of great significance had been discovered. This led to the decline in the interest in and use of Prolog, although it still has its proponents.

16.6 The Basic Elements of Prolog

There are now a number of different dialects of Prolog. These can be grouped into several categories: those that grew from the Marseille group, those that came from the Edinburgh group, and some dialects that have been developed for microcomputers, such as micro-Prolog, which is described by Clark and McCabe (1984). The syntactic forms of these are somewhat different. Rather than attempt to describe the syntax of several dialects of Prolog or some hybrid of them, we have chosen one particular, widely available dialect, which is the one developed at Edinburgh. This form of the language is sometimes called **Edinburgh syntax**. Its first implementation was on a DEC System-10 (Warren et al., 1979). Prolog implementations are available for virtually all popular computer platforms, for example, from the Free Software Organization (http://www.gnu.org).

16.6.1 Terms

As with programs in other languages, Prolog programs consist of collections of statements. There are only a few kinds of statements in Prolog, but they can be complex. All Prolog statements, as well as Prolog data, are constructed from terms.

A Prolog **term** is a constant, a variable, or a structure. A constant is either an **atom** or an integer. Atoms are the symbolic values of Prolog and are similar to their counterparts in LISP. In particular, an atom is either a string of letters, digits, and underscores that begins with a lowercase letter or a string of any printable ASCII characters delimited by apostrophes.

A variable is any string of letters, digits, and underscores that begins with an uppercase letter or an underscore (_). Variables are not bound to types by declarations. The binding of a value, and thus a type, to a variable is called an **instantiation**. Instantiation occurs only in the resolution process. A variable that has not been assigned a value is called **uninstantiated**. Instantiations last

only as long as it takes to satisfy one complete goal, which involves the proof or disproof of one proposition. Prolog variables are only distant relatives, in terms of both semantics and use, to the variables in the imperative languages.

The last kind of term is called a **structure**. Structures represent the atomic propositions of predicate calculus, and their general form is the same:

```
functor(parameter list)
```

The functor is any atom and is used to identify the structure. The parameter list can be any list of atoms, variables, or other structures. As discussed at length in the following subsection, structures are the means of specifying facts in Prolog. They can also be thought of as objects, in which case they allow facts to be stated in terms of several related atoms. In this sense, structures are relations, for they state relationships among terms. A structure is also a predicate when its context specifies it to be a query (question).

16.6.2 Fact Statements

Our discussion of Prolog statements begins with those statements used to construct the hypotheses, or database of assumed information—the statements from which new information can be inferred.

Prolog has two basic statement forms; these correspond to the headless and headed Horn clauses of predicate calculus. The simplest form of headless Horn clause in Prolog is a single structure, which is interpreted as an unconditional assertion, or fact. Logically, facts are simply propositions that are assumed to be true.

The following examples illustrate the kinds of facts one can have in a Prolog program. Notice that every Prolog statement is terminated by a period.

```
female(shelley).
male(bill).
female(mary).
male(jake).
father(bill, jake).
father(bill, shelley).
```

```
mother(mary, jake).
mother(mary, shelley).
```

These simple structures state certain facts about `jake`, `shelley`, `bill`, and `mary`. For example, the first states that `shelley` is a `female`. The last four connect their two parameters with a relationship that is named in the functor atom; for example, the fifth proposition might be interpreted to mean that `bill` is the `father` of `jake`. Note that these Prolog propositions, like those of predicate calculus, have no intrinsic semantics. They mean whatever the programmer wants them to mean. For example, the proposition

```
father(bill, jake).
```

could mean `bill` and `jake` have the same `father` or that `jake` is the `father` of `bill`. The most common and straightforward meaning, however, might be that `bill` is the `father` of `jake`.

16.6.3 Rule Statements

The other basic form of Prolog statement for constructing the database corresponds to a headed Horn clause. This form can be related to a known theorem in mathematics from which a conclusion can be drawn if the set of given conditions is satisfied. The right side is the antecedent, or *if* part, and the left side is the consequent, or *then* part. If the antecedent of a Prolog statement is true, then the consequent of the statement must also be true. Because they are Horn clauses, the consequent of a Prolog statement is a single term, while the antecedent can be either a single term or a conjunction.

Conjunctions contain multiple terms that are separated by logical AND operations. In Prolog, the AND operation is implied. The structures that specify atomic propositions in a conjunction are separated by commas, so one could consider the commas to be AND operators. As an example of a conjunction, consider the following:

```
female(shelley), child(shelley).
```

The general form of the Prolog headed Horn clause statement is

```
consequence :- antecedent_expression.
```

It is read as follows: “consequence can be concluded if the antecedent expression is true or can be made to be true by some instantiation of its variables.” For example,

```
ancestor(mary, shelley) :- mother(mary, shelley).
```

states that if `mary` is the `mother` of `shelley`, then `mary` is an `ancestor` of `shelley`. Headed Horn clauses are called **rules**, because they state rules of implication between propositions.

As with clausal form propositions in predicate calculus, Prolog statements can use variables to generalize their meaning. Recall that variables in clausal form provide a kind of implied universal quantifier. The following demonstrates the use of variables in Prolog statements:

```
parent(X, Y) :- mother(X, Y).
parent(X, Y) :- father(X, Y).
grandparent(X, Z) :- parent(X, Y) , parent(Y, Z).
```

These statements give rules of implication among some variables, or universal objects. In this case, the universal objects are `X`, `Y`, and `Z`. The first rule states that if there are instantiations of `X` and `Y` such that `mother(X, Y)` is true, then for those same instantiations of `X` and `Y`, `parent(X, Y)` is true.

The `=` operator, which is an infix operator, succeeds if its two term operands are the same. For example, `X = Y`. The `not` operator, which is a unary operator, reverses its operand, in the sense that it succeeds if its operand fails. For example, `not(X = Y)` succeeds if `X` is not equal to `Y`.

16.6.4 Goal Statements

So far, we have described the Prolog statements for logical propositions, which are used to describe both known facts and rules that describe logical relationships among facts. These statements are the basis for the theorem-proving model. The theorem is in the form of a proposition that we want the system to either prove or disprove. In Prolog, these propositions are called

goals, or **queries**. The syntactic form of Prolog goal statements is identical to that of headless Horn clauses. For example, we could have

```
man(fred).
```

to which the system will respond either `yes` or `no`. The answer `yes` means that the system has proved the goal was true under the given database of facts and relationships. The answer `no` means that either the goal was determined to be false or the system was simply unable to prove it.

Conjunctive propositions and propositions with variables are also legal goals. When variables are present, the system not only asserts the validity of the goal but also identifies the instantiations of the variables that make the goal true. For example,

```
father(X, mike).
```

can be asked. The system will then attempt, through unification, to find an instantiation of `X` that results in a true value for the goal.

Because goal statements and some nongoal statements have the same form (headless Horn clauses), a Prolog implementation must have some means of distinguishing between the two. Interactive Prolog implementations do this by simply having two modes, indicated by different interactive prompts: one for entering fact and rule statements and one for entering goals. The user can change the mode at any time.

16.6.5 The Inferencing Process of Prolog

This section examines Prolog resolution. Efficient use of Prolog requires that the programmer know precisely what the Prolog system does with his or her program.

When a goal is a compound proposition, each of the facts (structures) is called a **subgoal**. To prove that a goal is true, the inferencing process must find a chain of inference rules and/or facts in the database that connect the goal to one or more facts in the database. For example, if *Q* is the goal, then either *Q* must be found as a fact in the database or the inferencing process must find a fact P1 and a sequence of propositions P2, P3…, Pn such that

- P2 : - P1
- P3 : - P2
- . . .
- *Q* : - Pn

Of course, the process can be and often is complicated by rules with compound right sides and rules with variables. The process of finding the *P*s, when they exist, is basically a comparison, or matching, of terms with each other.

Because the process of proving a subgoal is done through a proposition--matching process, it is sometimes called **matching**. In some cases, proving a subgoal is called **satisfying** that subgoal.

Consider the following query:

```
man(bob).
```

This goal statement is the simplest kind. It is relatively easy for resolution to determine whether it is true or false: The pattern of this goal is compared with the facts and rules in the database. If the database includes the fact

```
man(bob).
```

the proof is trivial. If, however, the database contains the following fact and inference rule,

```
father(bob).
man(X) :- father(X).
```

Prolog would be required to find these two statements and use them to infer the truth of the goal. This would necessitate unification to instantiate `X` temporarily to `bob`.

Now consider the goal

```
man(X).
```

In this case, Prolog must match the goal against the propositions in the database. The first proposition that it finds that has the form of the goal, with any object as its parameter, will cause `X` to be instantiated with that object's value. `X` is then displayed as the result. If there is no proposition having the form of the goal, the system indicates, by saying `no`, that the goal cannot be satisfied.

There are two opposite approaches to attempting to match a given goal to a fact in the database. The system can begin with the facts and rules of the database and attempt to find a sequence of matches that lead to the goal. This approach is called **bottom-up resolution**, or **forward chaining**. The alternative is to begin with the goal and attempt to find a sequence of matching propositions that lead to some set of original facts in the database. This approach is called **top-down resolution**, or **backward chaining**. In general, backward chaining works well when there is a reasonably small set of candidate answers. The forward chaining approach is better when the number of possibly correct answers is large; in this situation, backward chaining would require a very large number of matches to get to an answer. Prolog implementations use backward chaining for resolution, presumably

because its designers believed backward chaining was more suitable for a larger class of problems than forward chaining.

The following example illustrates the difference between forward and backward chaining. Consider the query:

```
man(bob).
```

Assume the database contains

```
father(bob).
man(X) :- father(X).
```

Forward chaining would search for and find the first proposition. The goal is then inferred by matching the first proposition with the right side of the second rule (`father(X)`) through instantiation of `X` to `bob` and then matching the left side of the second proposition to the goal. Backward chaining would first match the goal with the left side of the second proposition (`man(X)`) through the instantiation of `X` to `bob`. As its last step, it would match the right side of the second proposition (now `father(bob)`) with the first proposition.

The next design question arises whenever the goal has more than one structure, as in our example. The question then is whether the solution search is done depth first or breadth first. A **depth-first** search finds a complete sequence of propositions—a proof—for the first subgoal before working on the others. A **breadth-first** search works on all subgoals of a given goal in parallel. Prolog's designers chose the depth-first approach primarily because it can be done with fewer computer resources. The breadth-first approach is a parallel search that can require a large amount of memory.

The last feature of Prolog's resolution mechanism that must be discussed is backtracking. When a goal with multiple subgoals is being processed and the system fails to show the truth of one of the subgoals, the system abandons the subgoal it cannot prove. It then reconsiders the previous subgoal, if there is one, and attempts to find an alternative solution to it. This backing up in the goal to the reconsideration of a previously proven subgoal is called **backtracking**. A new solution is found by beginning the search where the previous search for that subgoal stopped. Multiple solutions to a subgoal result from different instantiations of its variables. Backtracking can require a

great deal of time and space because it may have to find all possible proofs to every subgoal. These subgoal proofs may not be organized to minimize the time required to find the one that will result in the final complete proof, which exacerbates the problem.

To solidify your understanding of backtracking, consider the following example. Assume that there is a set of facts and rules in the database and that Prolog has been presented with the following compound goal:

```
male(X), parent(X, shelley).
```

This goal asks whether there is an instantiation of `X` such that `X` is a `male` and `X` is a `parent` of `shelley`. As its first step, Prolog finds the first fact in the database with `male` as its functor. It then instantiates `X` to the parameter of the found fact, say `mike`. Then, it attempts to prove that `parent(mike, shelley)` is true. If it fails, it backtracks to the first subgoal, `male(X)`, and attempts to resatisfy it with some alternative instantiation of `X`. The resolution process may have to find every `male` in the database before it finds the one that is a `parent` of `shelley`. It definitely must find all `male`s to prove that the goal cannot be satisfied. Note that our example goal might be processed more efficiently if the order of the two subgoals were reversed. Then, only after resolution had found a `parent` of `shelley` would it try to match that person with the `male` subgoal. This is more efficient if `shelley` has fewer parents than there are `male`s in the database, which seems like a reasonable assumption. Section 16.7.1 discusses a method of limiting the backtracking done by a Prolog system.

Database searches in Prolog always proceed in the direction of first to last.

The following two subsections describe Prolog examples that further illustrate the resolution process.

16.6.6 Simple Arithmetic

Prolog supports integer variables and integer arithmetic. Originally, the arithmetic operators were functors, so that the sum of `7` and the variable `X` was formed with

```
+(7, X)
```

Prolog now allows a more abbreviated syntax for arithmetic with the **is** operator. This operator takes an arithmetic expression as its right operand and a variable as its left operand. All variables in the expression must already be instantiated, but the left-side variable cannot be previously instantiated. For example, in

```
A is B / 17 + C.
```

if `B` and `C` are instantiated but `A` is not, then this clause will cause `A` to be instantiated with the value of the expression. When this happens, the clause is satisfied. If either `B` or `C` is not instantiated or `A` is instantiated, the clause is not satisfied and no instantiation of `A` can take place. The semantics of an **`is`** proposition is considerably different from that of an assignment statement in an imperative language. This difference can lead to an interesting scenario. Because the **`is`** operator makes the clause in which it appears look like an assignment statement, a beginning Prolog programmer may be tempted to write a statement such as

```
Sum is Sum + Number.
```

which is never useful, or even legal, in Prolog. If `Sum` is not instantiated, the reference to it in the right side is undefined and the clause fails. If `Sum` is already instantiated, the clause fails, because the left operand cannot have a current instantiation when **`is`** is evaluated. In either case, the instantiation of `Sum` to the new value will not take place. (If the value of `Sum + Number` is required, it can be bound to some new name.)

Prolog does not have assignment statements in the same sense as imperative languages. They are simply not needed in most of the programming for which Prolog was designed. The usefulness of assignment statements in imperative languages often depends on the capability of the programmer to control the execution control flow of the code in which the assignment statement is embedded. Because this type of control is not always possible in Prolog, such statements are far less useful.

As a simple example of the use of numeric computation in Prolog, consider the following problem: Suppose we know the average speeds of several

automobiles on a particular racetrack and the amount of time they are on the track. This basic information can be coded as facts, and the relationship between speed, time, and distance can be written as a rule, as in the following:

```
speed(ford, 100).
speed(chevy, 105).
speed(dodge, 95).
speed(volvo, 80).
time(ford, 20).
time(chevy, 21).
time(dodge, 24).
time(volvo, 24).
distance(X, Y) :- speed(X, Speed),
                  time(X, Time),
                  Y is Speed * Time.
```

Now, queries can request the distance traveled by a particular car. For example, the query

```
distance(chevy, Chevy_Distance).
```

instantiates `Chevy_Distance` with the value 2205. The first two clauses in the right side of the distance computation statement instantiate the variables `Speed` and `Time` with the corresponding values of the given automobile functor. After satisfying the goal, Prolog also displays the name `Chevy_Distance` and its value.

At this point it is instructive to take an operational look at how a Prolog system produces results. Prolog has a built-in structure named `trace` that displays the instantiations of values to variables at each step during the attempt to satisfy a given goal. `trace` is used to understand and debug Prolog programs. To understand `trace`, it is best to introduce a different model of the execution of Prolog programs, called the **tracing model**.

The tracing model describes Prolog execution in terms of four events: (1) call, which occurs at the beginning of an attempt to satisfy a goal, (2) exit, which occurs when a goal has been satisfied, (3) redo, which occurs when backtrack causes an attempt to resatisfy a goal, and (4) fail, which occurs when a goal fails. Call and exit can be related directly to the execution model

of a subprogram in an imperative language if processes like `distance` are thought of as subprograms. The other two events are unique to logic programming systems. In the following trace example, a trace of the computation of the value for `Chevy_Distance`, the goal requires no redo or fail events:

```
trace.
adistance(chevy, Chevy_Distance).
(1) 1 Call: distance(chevy, _0)?
(2) 2 Call: speed(chevy, _5)?
(2) 2 Exit: speed(chevy, 105)
(3) 2 Call: time(chevy, _6)?
(3) 2 Exit: time(chevy, 21)
(4) 2 Call: _0 is 105*21?
(4) 2 Exit: 2205 is 105*21
(1) 1 Exit: distance(chevy, 2205)
Chevy_Distance = 2205
```

Symbols in the trace that begin with the underscore character (_) are internal variables used to store instantiated values. The first column of the trace indicates the subgoal whose match is currently being attempted. For example, in the example trace, the first line with the indication `(3)` is an attempt to instantiate the temporary variable `_6` with a `time` value for `chevy`, where `time` is the second term in the right side of the statement that describes the computation of `distance`. The second column indicates the call depth of the matching process. The third column indicates the current action.

To illustrate backtracking, consider the following example database and traced compound goal:

```
likes(jake, chocolate).
likes(jake, apricots).
likes(darcie, licorice).
likes(darcie, apricots).
trace.
likes(jake, X), likes(darcie, X).
(1) 1 Call: likes(jake, _0)?
(1) 1 Exit: likes(jake, chocolate)
(2) 1 Call: likes(darcie, chocolate)?
(2) 1 Fail: likes(darcie, chocolate)
(1) 1 Redo: likes(jake, _0)?
(1) 1 Exit: likes(jake, apricots)
(3) 1 Call: likes(darcie, apricots)?
```

```
(3) 1 Exit: likes(darcie, apricots)
X = apricots
```

One can think about Prolog computations graphically as follows: Consider each goal as a box with four ports—call, fail, exit, and redo. Control enters a goal in the forward direction through its call port. Control can also enter a goal from the reverse direction through its redo port. Control can also leave a goal in two ways: If the goal succeeded, control leaves through the exit port; if the goal failed, control leaves through the fail port. A model of the example is shown in Figure 16.1. In this example, control flows through each subgoal twice. The second subgoal fails the first time, which forces a return through redo to the first subgoal.

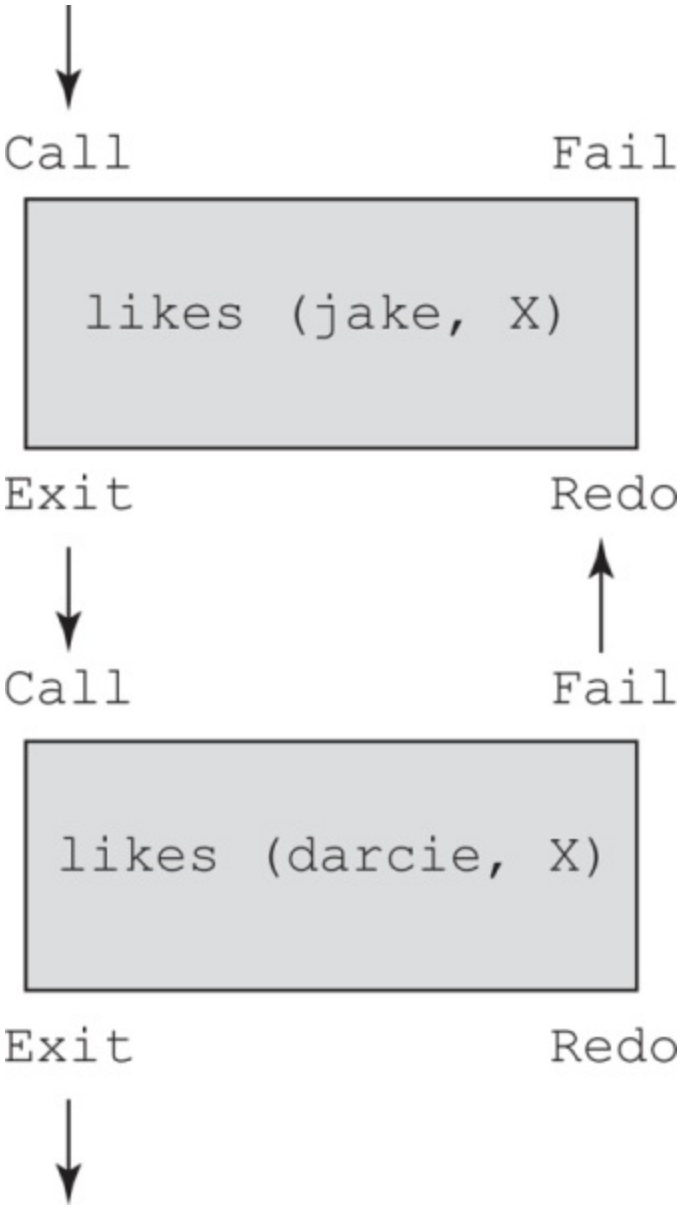


Figure 16.1 Control flow model for the goal `likes (jake, X), likes (darcie, X)`

Figure 16.1 Full Alternative Text

16.6.7 List Structures

So far, the only Prolog data structure we have discussed is the atomic proposition, which looks more like a function call than a data structure. Atomic propositions, which are also called structures, are actually a form of records. The other basic data structure supported is the list. Lists are sequences of any number of elements, where the elements can be atoms, atomic propositions, or any other terms, including other lists.

Prolog uses the syntax of ML and Haskell to specify lists. The list elements are separated by commas, and the entire list is delimited by square brackets, as in

```
[apple, prune, grape, kumquat]
```

The notation `[]` is used to denote the empty list. Instead of having explicit functions for constructing and dismantling lists, Prolog simply uses a special notation. `[X | Y]` denotes a list with head `X` and tail `Y`, where head and tail correspond to `CAR` and `CDR` in LISP. This is similar to the notation used in ML and Haskell.

A list can be created with a simple structure, as in

```
new_list([apple, prune, grape, kumquat]).
```

which states that the constant list `[apple, prune, grape, kumquat]` is a new element of the relation named `new_list` (a name we just made up). This statement does not bind the list to a variable named `new_list`; rather, it does the kind of thing that the proposition

```
male(jake)
```

does. That is, it states that `[apple, prune, grape, kumquat]` is a new element of `new_list`. Therefore, we could have a second proposition with a list argument, such as

```
new_list([apricot, peach, pear])
```

In query mode, one of the elements of `new_list` can be dismantled into head and tail with

```
new_list([New_List_Head | New_List_Tail]).
```

If `new_list` has been set to have the two elements as shown, this statement instantiates `New_List_Head` with the head of the first list element (in this case, `apple`) and `New_List_Tail` with the tail of the list (or `[prune, grape, kumquat]`). If this were part of a compound goal and backtracking forced a new evaluation of it, `New_List_Head` and `New_List_Tail` would be reinstantiated to `apricot` and `[peach, pear]`, respectively, because `[apricot, peach, pear]` is the next element of `new_list`.

The `|` operator used to dismantle lists can also be used to create lists from given instantiated head and tail components, as in

```
[Element_1 | List_2]
```

If `Element_1` has been instantiated with `pickle` and `List_2` has been instantiated with `[peanut, prune, popcorn]`, the sample notation will create, for this one reference, the list `[pickle, peanut, prune, popcorn]`.

As stated previously, the list notation that includes the `|` symbol is universal: It can specify either a list construction or a list dismantling. Note further that the following are equivalent:

```
[apricot, peach, pear | []]
[apricot, peach | [pear]]
[apricot | [peach, pear]]
```

With lists, certain basic operations are often required, such as those found in LISP, ML, and Haskell. As an example of such operations in Prolog, we examine a definition of `append`, which is related to such a function in LISP. In this example, the differences and similarities between functional and declarative languages can be seen. We need not specify how Prolog is to construct a new list from the given lists; rather, we need specify only the characteristics of the new list in terms of the given lists.

In appearance, the Prolog definition of `append` is very similar to the ML

version that appears in Chapter 15, and a kind of recursion in resolution is used in a similar way to produce the new list. In the case of Prolog, the recursion is caused and controlled by the resolution process. As with ML and Haskell, a pattern-matching process is used to choose, based on the actual parameter, between two different definitions of the append process.

The first two parameters to the `append` operation in the following code are the two lists to be appended, and the third parameter is the resulting list:

```
append([], List, List).
append([Head | List_1], List_2, [Head | List_3]) :-
            append(List_1, List_2, List_3).
```

The first proposition specifies that when the empty list is appended to any other list, that other list is the result. This statement corresponds to the -recursion-terminating step of the ML `append` function. Note that the terminating proposition is placed before the recursion proposition. This is done because we know that Prolog will match the two propositions in order, starting with the first (because of its use of the depth-first order).

The second proposition specifies several characteristics of the new list. It corresponds to the recursion step in the ML function. The left-side predicate states that the first element of the new list is the same as the first element of the first given list, because they are both named `Head`. Whenever `Head` is instantiated to a value, all occurrences of `Head` in the goal are, in effect, simultaneously instantiated to that value. The right side of the second statement specifies that the tail of the first given list `(List_1)` has the second given list `(List_2)` appended to it to form the tail `(List_3)` of the resulting list.

One way to read the second statement of `append` is as follows: Appending the list `[Head | List_1]` to any list `List_2` produces the list `[Head | List_3]`, but only if the list `List_3` is formed by appending `List_1` to `List_2`. In LISP, this would be

```
(CONS (CAR FIRST) (APPEND (CDR FIRST) SECOND))
```

In both the Prolog and LISP versions, the resulting list is not constructed until the recursion produces the terminating condition; in this case, the first list

must become empty. Then, the resulting list is built using the append function itself; the elements taken from the first list are added, in reverse order, to the second list. The reversing is done by the unraveling of the recursion.

One fundamental difference between Prolog's append and those of LISP and ML is that Prolog's append is a predicate—it does not return a list, it returns yes or no. The new list is the value of its third parameter.

To illustrate how the append process progresses, consider the following traced example:

```
trace.
append([bob, jo], [jake, darcie], Family).
(1) 1 Call: append([bob, jo], [jake, darcie], _10)?
(2) 2 Call: append([jo], [jake, darcie], _18)?
(3) 3 Call: append([], [jake, darcie], _25)?
(3) 3 Exit: append([], [jake, darcie], [jake, darcie])
(2) 2 Exit: append([jo], [jake, darcie], [jo, jake,
                    darcie])
(1) 1 Exit: append([bob, jo], [jake, darcie],
                   [bob, jo, jake, darcie])
Family = [bob, jo, jake, darcie]
yes
```

The first two calls, which represent subgoals, have List_1 nonempty, so they create the recursive calls from the right side of the second statement. The left side of the second statement effectively specifies the arguments for the recursive calls, or goals, thus dismantling the first list one element per step. When the first list becomes empty, in a call, or subgoal, the current instance of the right side of the second statement succeeds by matching the first statement. The effect of this is to return as the third parameter the value of the empty list appended to the second original parameter list. On successive exits, which represent successful matches, the elements that were removed from the first list are appended to the resulting list, Family. When the exit from the first goal is accomplished, the process is complete, and the resulting list is displayed.

Another difference between Prolog's append and those of LISP and ML is that Prolog's append is more flexible than that of those languages. For example, in Prolog we can use append to determine what two lists can be

appended to get `[a, b, c]` with

```
append(X, Y, [a, b, c]).
```

This results in the following:

```
X = []
Y = [a, b, c]
```

If we type a semicolon at this output we get the alternative result:

```
X = [a]
Y = [b, c]
```

Continuing, we get the following:

```
X = [a, b]
Y = [c];
X = [a, b, c]
Y = []
```

The `append` predicate can also be used to create other list operations, such as the following, whose effect we invite the reader to determine. Note that `list_op_2` is meant to be used by providing a list as its first parameter and a variable as its second, and the result of `list_op_2` is the value to which the second parameter is instantiated.

```
list_op_2([], []).
list_op_2([Head | Tail], List) :-
list_op_2(Tail, Result), append(Result, [Head], List).
```

As you may have been able to determine, `list_op_2` causes the Prolog system to instantiate its second parameter with a list that has the elements of the list of the first parameter, but in reverse order. For example, `([apple, orange, grape], Q)` instantiates `Q` with the list `[grape, orange, apple]`.

Once again, although the LISP and Prolog languages are fundamentally different, similar operations can use similar approaches. In the case of the reverse operation, both the Prolog's `list_op_2` and LISP's `reverse` function include the recursion-terminating condition, along with the basic process of appending the reversal of the `CDR` or tail of the list to the `CAR` or head of the

list to create the result list.

The following is a trace of this process, now named `reverse`:

```
trace.
reverse([a, b, c], Q).
(1) 1 Call: reverse([a, b, c], _6)?
(2) 2 Call: reverse([b, c], _65636)?
(3) 3 Call: reverse([c], _65646)?
(4) 4 Call: reverse([], _65656)?
(4) 4 Exit: reverse([], [])
(5) 4 Call: append([], [c], _65646)?
(5) 4 Exit: append([], [c], [c])
(3) 3 Exit: reverse([c], [c])
(6) 3 Call: append([c], [b], _65636)?
(7) 4 Call: append([], [b], _25)?
(7) 4 Exit: append([], [b], [b])
(6) 3 Exit: append([c], [b], [c, b])
(2) 2 Exit: reverse([b, c], [c, b])
(8) 2 Call: append([c, b], [a], _6)?
(9) 3 Call: append([b], [a], _32)?
(10) 4 Call: append([], [a], _39)?
(10) 4 Exit: append([], [a], [a])
(9) 3 Exit: append([b], [a], [b, a])
(8) 2 Exit: append([c, b], [a], [c, b, a])
(1) 1 Exit: reverse([a, b, c], [c, b, a])
Q = [c, b, a]
```

Suppose we need to be able to determine whether a given symbol is in a given list. A straightforward Prolog description of this is

```
member(Element, [Element | _]).
member(Element, [_ | List]) :- member(Element, List).
```

The underscore indicates an “anonymous” variable; it is used to mean that we do not care what instantiation it might get from unification. The first `-` statement in the previous example succeeds if `Element` is the head of the list, either initially or after several recursions through the second statement. The second statement succeeds if `Element` is in the tail of the list. Consider the `-` following traced examples:

```
trace.
member(a, [b, c, d]).
(1) 1 Call: member(a, [b, c, d])?
```

```
(2) 2 Call: member(a, [c, d])?
(3) 3 Call: member(a, [d])?
(4) 4 Call: member(a, [])?
(4) 4 Fail: member(a, [])
(3) 3 Fail: member(a, [d])
(2) 2 Fail: member(a, [c, d])
(1) 1 Fail: member(a, [b, c, d])
no
member(a, [b, a, c]).
(1) 1 Call: member(a, [b, a, c])?
(2) 2 Call: member(a, [a, c])?
(2) 2 Exit: member(a, [a, c])
(1) 1 Exit: member(a, [b, a, c])
yes
```

16.7 Deficiencies of Prolog

Although Prolog is a useful tool, it is neither a pure nor a perfect logic programming language. This section describes some of the problems with Prolog.

16.7.1 Resolution Order Control

Prolog, for reasons of efficiency, allows the user to control the ordering of - pattern matching during resolution. In a pure logic programming environment, the order of attempted matches that take place during resolution is nondeterministic, and all matches could be attempted concurrently. However, because Prolog always matches in the same order, starting at the beginning of the database and at the left end of a given goal, the user can profoundly affect efficiency by ordering the database statements to optimize a particular application. For example, if the user knows that certain rules are much more likely to succeed than the others during a particular "execution," then the program can be made more efficient by placing those rules at the beginning of the database.

In addition to allowing the user to control database and subgoal ordering, Prolog, in another concession to efficiency, allows some explicit control of backtracking. This is done with the cut operator, which is specified by an exclamation point (`!`). The cut operator is actually a goal, not an operator. As a goal, it always succeeds immediately, but it cannot be resatisfied through backtracking. Thus, a side effect of the cut is that subgoals to its left in a compound goal also cannot be resatisfied through backtracking. For example, in the goal

```
a, b, !, c, d.
```

if both `a` and `b` succeed but `c` fails, the whole goal fails. This goal would be used if it were known that whenever `c` fails, it is a waste of time to resatisfy `b` or `a`.

The purpose of the cut then is to allow the user to make programs more efficient by telling the system when it should not attempt to resatisfy subgoals that presumably could not result in a complete proof.

As an example of the use of the cut operator, consider the `member` rules from Section 16.6.7, which are:

```
member(Element, [Element | _]).
member(Element, [_ | List]) :- member(Element, List).
```

If the list argument to `member` represents a set, then it can be satisfied only once (sets contain no duplicate elements). Therefore, if `member` is used as a subgoal in a multiple subgoal goal statement, there can be a problem. The problem is that if `member` succeeds but the next subgoal fails, backtracking will attempt to resatisfy `member` by continuing a prior match. But because the list argument to `member` has only one copy of the element to begin with, `member` cannot possibly succeed again, which eventually causes the whole goal to fail, in spite of any additional attempts to resatisfy `member`. For example, consider the goal:

```
dem_candidate(X) :- member(X, democrats), tests(X).
```

This goal determines whether a given person is a democrat and is a good candidate to run for a particular position. The `tests` subgoal checks a variety of characteristics of the given democrat to determine the suitability of the person for the position. If the set of democrats has no duplicates, then we do not want to back up to the `member` subgoal if the `tests` subgoal fails, because `member` will search all of the other democrats but fail, because there are no duplicates. The second attempt of `member` subgoal will be a waste of computation time. The solution to this inefficiency is to add a right side to the first statement of the `member` definition, with the cut operator as the sole element, as in

```
member(Element, [Element | _]) :- !.
```

Backtracking will not attempt to resatisfy `member` but instead will cause the entire subgoal to fail.

Cut is particularly useful in a programming strategy in Prolog called

generate and test. In programs that use the generate-and-test strategy, the goal consists of subgoals that generate potential solutions, which are then checked by later "test" subgoals. Rejected solutions require backtracking to "generator" subgoals, which generate new potential solutions. As an example of a generate-and-test program, consider the following, which appears in Clocksin and Mellish (2013):

```
divide(N1, N2, Result) :- is_integer(Result),
        Product1 is Result * N2,
        Product2 is (Result + 1) * N2,
        Product1 =< N1, Product2 >
          N1, !.
```

This program performs integer division, using addition and multiplication. Because most Prolog systems provide division as an operator, this program actually is not useful, other than to illustrate a simple generate-and-test program.

The predicate `is_integer` succeeds as long as its parameter can be instantiated to some nonnegative integer. If its argument is not instantiated, `is_integer` instantiates it to the value 0. If the argument is instantiated to an integer, `is_integer` instantiates it to the next larger integer value.

So, in `divide`, `is_integer` is the generator subgoal. It generates elements of the sequence 0, 1, 2, . . . , one each time it is satisfied. All of the other subgoals are the testing subgoals—they check to determine whether the value produced by `is_integer` is, in fact, the quotient of the first two parameters, `N1` and `N2`. The purpose of the cut as the last subgoal is simple: It prevents `divide` from ever trying to find an alternative solution once it has found *the* solution. Although `is_integer` can generate a huge number of candidates, only one is the solution, so the cut here prevents useless attempts to produce secondary solutions.

Use of the cut operator has been compared to the use of the goto in imperative languages (van Emden, 1980). Although it is sometimes needed, it is possible to abuse it. Indeed, it is sometimes used to make logic programs have a control flow that is inspired by imperative programming styles.

The ability to tamper with control flow in a Prolog program is a deficiency,

because it is directly detrimental to one of the important advantages of logic programming—that programs do not specify how solutions are to be found. Rather, they simply specify what the solution should look like. This design makes programs easier to write and easier to read. They are not cluttered with the details of how the solutions are to be determined and, in particular, the precise order in which the computations are done to produce the solution. So, while logic programming requires no control flow directions, Prolog programs frequently use them, mostly for the sake of efficiency.

16.7.2 The Closed-World Assumption

The nature of Prolog's resolution sometimes creates misleading results. The only truths, as far as Prolog is concerned, are those that can be proved using its database. It has no knowledge of the world other than its database. When the system receives a query and the database does not have information to prove the query absolutely, the query is assumed to be false. Prolog can prove that a given goal is true, but it cannot prove that a given goal is false. It simply assumes that, because it cannot prove a goal true, the goal must be false. In essence, Prolog is a true/fail system, rather than a true/false system.

Actually, the closed-world assumption should not be at all foreign to you—our judicial system operates the same way. Suspects are innocent until proven guilty. They need not be proven innocent. If a trial cannot prove a person guilty, he or she is considered innocent.

The problem of the closed-world assumption is related to the negation problem, which is discussed in the following subsection.

16.7.3 The Negation Problem

Another problem with Prolog is its difficulty with negation. Consider the following database of two facts and a relationship:

```
parent(bill, jake).
parent(bill, shelley).
sibling(X, Y) :- (parent(M, X), parent(M, Y).
```

Now, suppose we typed the query

```
sibling(X, Y).
```

Prolog will respond with

```
X = jake
Y = jake
```

Thus, Prolog "thinks" `jake` is a `sibling` of himself. This happens because the system first instantiates `M` with `bill` and `X` with `jake` to make the first subgoal, `parent(M, X)`, true. It then starts at the beginning of the database again to match the second subgoal, `parent(M, Y)`, and arrives at the instantiations of `M` with `bill` and `Y` with `jake`. Because the two subgoals are satisfied independently, with both matchings starting at the database's beginning, the shown response appears. To avoid this result, `X` must be specified to be a `sibling` of `Y` only if they have the same `parents` *and* they are not the same. Unfortunately, stating that they are not equal is not straightforward in Prolog, as we will discuss. The most exacting method would require adding a fact for every pair of atoms, stating that they were not the same. This can, of course, cause the database to become very large, for there is often far more negative information than positive information. For example, most people have 364 more unbirthdays than they have birthdays.

A simple alternative solution is to state in the goal that `X` must not be the same as `Y`, as in

```
sibling(X, Y) :- parent(M, X), parent(M, Y), not(X = Y).
```

In other situations, the solution is not so simple.

The Prolog `not` operator is satisfied in this case if resolution cannot satisfy the subgoal `X = Y`. Therefore, if the `not` succeeds, it does not necessarily mean that `X` is not equal to `Y`; rather, it means that resolution cannot prove from the database that `X` is the same as `Y`. Thus, the Prolog `not` operator is not equivalent to a logical NOT operator, in which NOT means that its operand is

probably true. This nonequivalency can lead to a problem if we happen to have a goal of the form

```
not(not(some_goal)).
```

which would be equivalent to

```
some_goal.
```

if Prolog's `not` operator were a true logical NOT operator. In some cases, however, they are not the same. For example, consider again the `member` rules:

```
member(Element, [Element | _]) :- !.
member(Element, [_ | List]) :- member(Element, List).
```

To discover one of the elements of a given list, we could use the goal

```
member(X, [mary, fred, barb]).
```

which would cause `X` to be instantiated with `mary`, which would then be printed. But if we used

```
not(not(member(X, [mary, fred, barb]))).
```

the following sequence of events would take place: First, the inner goal would succeed, instantiating `X` to `mary`. Then, Prolog would attempt to satisfy the next goal:

```
not(member(X, [mary, fred, barb])).
```

This statement would fail because `member` succeeded. When this goal failed, `X` would be uninstantiated, because Prolog always uninstantiates all variables in all goals that fail. Next, Prolog would attempt to satisfy the outer `not` goal, which would succeed, because its argument had failed. Finally, the result, which is `X`, would be printed. But `X` would not be currently instantiated, so the system would indicate that. Generally, uninstantiated variables are printed in the form of a string of digits preceded by an underscore. So, the fact that Prolog's `not` is not equivalent to a logical NOT can be, at the very least, misleading.

The fundamental reason why logical NOT cannot be an integral part of Prolog is the form of the Horn clause:

- A :- $B_1 \cap B_2 \cap \ldots \cap B_n$

If all the B propositions are true, it can be concluded that A is true. But regardless of the truth or falseness of any or all of the B's, it cannot be concluded that A is false. From positive logic, one can conclude only positive logic. Thus, the use of Horn clause form prevents any negative conclusions.

16.7.4 Intrinsic Limitations

A fundamental goal of logic programming, as stated in Section 16.4, is to provide nonprocedural programming; that is, a system by which programmers specify what a program is supposed to do but need not specify how that is to be accomplished. The example given there for sorting is rewritten here:

- sort(old_list, new_list) $\subset$ permute(old_list, new_list) $\cap$ sorted(new_list)
- sorted(list) $\subset \forall$ j such that $1 \leq j < n$, list(j) $\leq$ list (j+1)

It is straightforward to write this in Prolog. For example, the sorted subgoal can be expressed as

```
sorted ([]).
sorted ([x]).
sorted ([x, y | list]) :- x <= y, sorted ([y | list]).
```

The problem with this sort process is that it has no idea of how to sort, other than simply to enumerate all permutations of the given list until it happens to create the one that has the list in sorted order—a very slow process, indeed.

So far, no one has discovered a process by which the description of a sorted list can be transformed into some efficient algorithm for sorting. Resolution is capable of many interesting things, but certainly not this. Therefore, a Prolog program that sorts a list must specify the details of how that sorting

can be done, as is the case in an imperative or functional language.

Do all of these problems mean that logic programming should be abandoned? Absolutely not! As it is, it is capable of dealing with many useful applications. Furthermore, it is based on an intriguing concept and is therefore interesting in and of itself. Finally, there is the possibility that new inferencing techniques will be developed that will allow a logic programming language system to efficiently deal with progressively larger classes of problems.

16.8 Applications of Logic Programming

In this section, we briefly describe a few of the larger classes of present and potential applications of logic programming in general and Prolog in particular.

16.8.1 Relational Database Management Systems

Relational database management systems (RDBMSs) store data in the form of tables. Queries on such databases are often stated in Structured Query Language (SQL). SQL is nonprocedural in the same sense that logic programming is nonprocedural. The user does not describe how to retrieve the answer; rather, he or she describes only the characteristics of the answer. The connection between logic programming and RDBMSs should be obvious. Simple tables of information can be described by Prolog structures, and relationships between tables can be conveniently and easily described by Prolog rules. The retrieval process is inherent in the resolution operation. The goal statements of Prolog provide the queries for the RDBMS. Logic programming is thus a natural match to the needs of implementing an RDBMS.

One of the advantages of using logic programming to implement an RDBMS is that only a single language is required. In a typical RDBMS, a database language includes statements for data definitions, data manipulation, and queries, all of which are embedded in a general-purpose programming language, such as COBOL. The general-purpose language is used for processing the data and input and output functions. All of these functions can be done in a logic programming language.

Another advantage of using logic programming to implement an RDBMS is that deductive capability is built in. Conventional RDBMSs cannot deduce anything from a database other than what is explicitly stored in them. They contain only facts, rather than facts *and* inference rules. The primary disadvantage of using logic programming for an RDBMS, compared with a conventional RDBMS, is that the logic programming implementation is slower. Logical inferences take longer than ordinary table look-up methods using imperative programming techniques.

16.8.2 Expert Systems

Expert systems are computer systems designed to emulate human expertise in some particular domain. They consist of a database of facts, an inferencing process, some heuristics about the domain, and some friendly human interface that makes the system appear much like an expert human consultant. In addition to their initial knowledge base, which is provided by a human expert, expert systems learn from the process of being used, so their databases must be capable of growing dynamically. Also, an expert system should include the capability of interrogating the user to get additional information when it determines that such information is needed.

One of the central problems for the designer of an expert system is dealing with the inevitable inconsistencies and incompleteness of the database. Logic programming appears to be well suited to deal with these problems. For example, default inference rules can help deal with the problem of incompleteness.

Prolog can and has been used to construct expert systems. It can easily fulfill the basic needs of expert systems, using resolution as the basis for query processing, using its ability to add facts and rules to provide the learning capability, and using its trace facility to inform the user of the “reasoning” behind a given result. Missing from Prolog is the automatic ability of the system to query the user for additional information when it is needed.

One of the most widely known uses of logic programming in expert systems is the expert system construction system known as APES, which is described

in Sergot (1983) and Hammond (1983). The APES system includes a very flexible facility for gathering information from the user during expert system construction. It also includes a second interpreter for producing explanations to its answers to queries.

APES has been successfully used to produce several expert systems, including one for the rules of a government social benefits program and one for the British Nationality Act, which is the definitive source for rules of British citizenship.

16.8.3 Natural-Language Processing

Certain kinds of natural-language processing can be done with logic programming. In particular, natural-language interfaces to computer software systems, such as intelligent databases and other intelligent knowledge-based systems, can be conveniently done with logic programming. For describing language syntax, forms of logic programming have been found to be equivalent to context-free grammars. Proof procedures in logic programming systems have been found to be equivalent to certain parsing strategies. In fact, backward-chaining resolution can be used directly to parse sentences whose structures are described by context-free grammars. It has also been discovered that some kinds of semantics of natural languages can be made clear by modeling the languages with logic programming. In particular, research in logic-based semantics networks has shown that sets of sentences in natural languages can be expressed in clausal form (Deliyanni and Kowalski, 1979). Kowalski (1979) also discusses logic-based semantic networks.

SUMMARY

Symbolic logic provides the basis for logic programming and logic programming languages. The approach of logic programming is to use as a database a collection of facts and rules that state relationships between facts and to use an automatic inferencing process to check the validity of new propositions, assuming the facts and rules of the database are true. This approach is the one developed for automatic theorem proving.

Prolog is the most widely used logic programming language. The origins of logic programming lie in Robinson's development of the resolution rule for logical inference. Prolog was developed primarily by Colmeraeur and Roussel at Marseille, with some help from Kowalski at Edinburgh.

Logic programs are nonprocedural, which means that the characteristics of the solution are given but the process of getting the solution is not.

Prolog statements are facts, rules, or goals. Most are made up of structures, which are atomic propositions, and logic operators, although arithmetic expressions are also allowed.

Resolution is the primary activity of a Prolog interpreter. This process, which uses backtracking extensively, involves mainly pattern matching among propositions. When variables are involved, they can be instantiated to values to provide matches. This instantiation process is called *unification*.

There are a number of problems with the current state of logic programming. For reasons of efficiency, and even to avoid infinite loops, programmers must sometimes state control flow information in their programs. Also, there are the problems of the closed-world assumption and negation.

Logic programming has been used in a number of different areas, primarily in relational database systems, expert systems, and natural-language processing.

BIBLIOGRAPHIC NOTES

- The Prolog language is described in several books. Edinburgh's form of the language is covered in Clocksin and Mellish (2003). The microcomputer implementation is described in Clark and McCabe (1984).

- Hogger (1991) is an excellent book on the general topic of logic programming. It is the source of the material in this chapter's section on logic programming applications.

REVIEW QUESTIONS

1. What are the three primary uses of symbolic logic in formal logic?
2. What are the two parts of a compound term?
3. What are the two modes in which a proposition can be stated?
4. What is the general form of a proposition in clausal form?
5. What are antecedents? Consequences?
6. Give general (not rigorous) definitions of *resolution* and *unification*.
7. What are the forms of Horn clauses?
8. What is the basic concept of declarative semantics?
9. What does it mean for a language to be nonprocedural?
10. What are the three forms of a Prolog term?
11. What is an uninstantiated variable?
12. What are the syntactic forms and usage of fact and rule statements in Prolog?
13. What is a conjunction?
14. Explain the two approaches to matching goals to facts in a database.
15. Explain the difference between a depth-first and a breadth-first search when discussing how multiple goals are satisfied.
16. Explain how backtracking works in Prolog.
17. Explain what is wrong with the Prolog statement `K is K + 1`.

18. What are the two ways a Prolog programmer can control the order of pattern matching during resolution?

19. Explain the generate-and-test programming strategy in Prolog.

20. Explain the closed-world assumption used by Prolog. Why is this a limitation?

21. Explain the negation problem with Prolog. Why is this a limitation?

22. Explain the connection between automatic theorem proving and Prolog's inferencing process.

23. Explain the difference between procedural and nonprocedural languages.

24. Explain why Prolog systems must do backtracking.

25. What is the relationship between resolution and unification in Prolog?

PROBLEM SET

1. Compare the concept of data typing in C# with that of Prolog.

2. Describe how a multiple-processor machine could be used to implement resolution. Could Prolog, as currently defined, use this method?

3. Write a Prolog description of your family tree (based only on facts), going back to your grandparents and including all descendants. Be sure to include all relationships.

4. Write a set of rules for family relationships, including all relationships from grandparents through two generations. Now add these to the facts of Problem 3, and eliminate as many of the facts as you can.

5. Write the following English conditional statements as Prolog-headed Horn clauses:

    1. If Fred is the father of Mike, then Fred is an ancestor of Mike.

    2. If Mike is the father of Joe and Mike is the father of Mary, then Mary is the sister of Joe.

    3. If Mike is the brother of Fred and Fred is the father of Mary, then Mike is the uncle of Mary.

6. Explain two ways in which the list-processing capabilities of Scheme and Prolog are similar.

7. In what way are the list-processing capabilities of Scheme and Prolog different?

8. Write a comparison of Prolog with ML, including two similarities and two differences.

9. From a book on Prolog, learn and write a description of an occur-check

problem. Why does Prolog allow this problem to exist in its implementation?

10. Find a good source of information on Skolem normal form and write a brief but clear explanation of it.

PROGRAMMING EXERCISES

1. Using the structures `parent(X, Y)`, `male(X)`, and `female(X)`, write a structure that defines `mother(X, Y)`.

2. Using the structures `parent(X, Y)`, `male(X)`, and `female(X)`, write a structure that defines `sister(X, Y)`.

3. Write a Prolog program that finds the maximum of a list of numbers.

4. Write a Prolog program that succeeds if the intersection of two given list parameters is empty.

5. Write a Prolog program that returns a list containing the union of the elements of two given lists.

6. Write a Prolog program that returns the final element of a given list.

7. Write a Prolog program that implements quicksort.

Bibliography

1. ACM. (1979) “Part A: Preliminary Ada Reference Manual” and “Part B: Rationale for the Design of the Ada Programming Language.” SIGPLAN Notices, Vol. 14, No. 6.

2. ACM. (1993a) “History of Programming Language Conference Proceedings.” ACM SIGPLAN Notices, Vol. 28, No. 3, March.

3. ACM. (1993b) “High Performance FORTRAN Language Specification Part 1.” FORTRAN Forum, Vol. 12, No. 4.

4. Aho, A. V., B. W. Kernighan, and P. J. Weinberger. (1988) The AWK Programming Language. Addison-Wesley, Reading, MA.

5. Aho, A. V., M. S. Lam, R. Sethi, and J. D. Ullman. (2006) Compilers: Principles, Techniques, and Tools, 2e. Addison-Wesley, Reading, MA.

6. Albahari, J. and B. Abrahari (2012) C# 5.0 in a Nutshell, O’Reilly Media, Sebastopol, CA.

7. Andrews, G. R., and F. B. Schneider. (1983) “Concepts and Notations for Concurrent Programming.” ACM Computing Surveys, Vol. 15, No. 1, pp. 3–43.

8. ANSI. (1966) American National Standard Programming Language FORTRAN. American National Standards Institute, New York.

9. ANSI. (1976) American National Standard Programming Language PL/I. ANSI X3.53–1976. American National Standards Institute, New York.

10. ANSI. (1978a) American National Standard Programming Language FORTRAN. ANSI X3.9–1978. American National Standards Institute, New York.

11. ANSI. (1978b) American National Standard Programming Language Minimal BASIC. ANSI X3.60–1978. American National Standards Institute, New York.

12. ANSI. (1989) American National Standard Programming Language C. ANSI X3.159–1989. American National Standards Institute, New York.

13. ANSI. (1992) American National Standard Programming Language FORTRAN 90. ANSI X3. 198–1992. American National Standards Institute, New York.

14. Arden, B. W., B. A. Galler, and R. M. Graham. (1961) “MAD at Michigan.” Datamation, Vol. 7, No. 12, pp. 27–28.

15. ARM. (1995) Ada Reference Manual. ISO/IEC/ANSI 8652:19. Intermetrics, Cambridge, MA.

16. Arnold, K., J. Gosling, and D. Holmes. (2006) The Java (TM) Programming Language, 4e. Addison-Wesley, Reading, MA.

17. Backus, J. (1954) “The IBM 701 Speedcoding System.” J. ACM, Vol. 1, pp. 4–6.

18. Backus, J. (1959) “The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference.” Proceedings International Conference on Information Processing. UNESCO, Paris, pp. 125–132.

19. Backus, J. (1978) “Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs.” Commun. ACM, Vol. 21, No. 8, pp. 613–641.

20. Backus, J., F. L. Bauer, J. Green, C. Katz, J. McCarthy, P. Naur, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger. (1963) “Revised Report on the Algorithmic Language ALGOL 60.” Commun. ACM, Vol. 6, No. 1, pp. 1–17.

21. Ben-Ari, M. (1982) Principles of Concurrent Programming. Prentice Hall, Englewood Cliffs, NJ.

22. Birtwistle, G. M., O.-J. Dahl, B. Myhrhaug, and K. Nygaard. (1973) Simula BEGIN. Van Nostrand Reinhold, New York.

23. Bodwin, J. M., L. Bradley, K. Kanda, D. Litle, and U. F. Pleban. (1982) "Experience with an Experimental Compiler Generator Based on Denotational Semantics." ACM SIGPLAN Notices, Vol. 17, No. 6, pp. 216–229.

24. Bohm, C., and G. Jacopini. (1966) "Flow Diagrams, Turing Machines, and Languages with Only Two Formation Rules." Commun. ACM, Vol. 9, No. 5, pp. 366–371.

25. Bolsky, M., and D. Korn. (1995) The New KornShell Command and Programming Language. Prentice Hall, Englewood Cliffs, NJ.

26. Booch, G. (1987) Software Engineering with Ada, 2e. Benjamin/Cummings, Redwood City, CA.

27. Brinch Hansen, P. (1973) Operating System Principles. Prentice Hall, Englewood Cliffs, NJ.

28. Brinch Hansen, P. (1975) "The Programming Language Concurrent-Pascal." IEEE Transactions on Software Engineering, Vol. 1, No. 2, pp. 199–207.

29. Brinch Hansen, P. (1977) The Architecture of Concurrent Programs. Prentice Hall, Englewood Cliffs, NJ.

30. Brinch Hansen, P. (1978) "Distributed Processes: A Concurrent Programming Concept." Commun. ACM, Vol. 21, No. 11, pp. 934–941.

31. Brown, J. A., S. Pakin, and R. P. Polivka. (1988) APL2 at a Glance. Prentice Hall, Englewood Cliffs, NJ.

32. Campione, M., K. Walrath, and A. Huml. (2001) The Java Tutorial, 3e. Addison-Wesley, Reading, MA.

33. Chambers, C., and D. Ungar. (1991) “Making Pure Object-Oriented Languages Practical.” SIGPLAN Notices, Vol. 26, No. 1, pp. 1–15.

34. Chomsky, N. (1956) “Three Models for the Description of Language.” IRE Transactions on Information Theory, Vol. 2, No. 3, pp. 113–124.

35. Chomsky, N. (1959) “On Certain Formal Properties of Grammars.” Information and Control, Vol. 2, No. 2, pp. 137–167.

36. Christiansen, T., B. D. Foy, and L. Wall, with J. Orwant. (2013) Programming Perl, 4e. O’Reilly & Associates, Sebastopol, CA.

37. Church, A. (1941) Annals of Mathematics Studies. Calculi of Lambda Conversion, Vol. 6. Princeton University Press, Princeton, NJ. Reprinted by Klaus Reprint Corporation, New York, 1965.

38. Clark, K. L., and F. G. McCabe. (1984) Micro-PROLOG: Programming in Logic. Prentice Hall, Englewood Cliffs, NJ.

39. Clarke, L. A., J. C. Wileden, and A. L. Wolf. (1980) “Nesting in Ada Is for the Birds.” ACM SIGPLAN Notices, Vol. 15, No. 11, pp. 139–145.

40. Cleaveland, J. C. (1986) An Introduction to Data Types. Addison-Wesley, Reading, MA.

41. Cleaveland, J. C., and R. C. Uzgalis. (1976) Grammars for Programming Languages: What Every Programmer Should Know About Grammar. American Elsevier, New York.

42. Clocksin, W. F., and C. S. Mellish. (2013) Programming in Prolog: Using the ISO Standard. Springer-Verlag, New York.

43. Cohen, J. (1981) “Garbage Collection of Linked Data Structures.” ACM Computing Surveys, Vol. 13, No. 3, pp. 341–368.

44. Conway, R., and R. Constable. (1976) “PL/-CS—A Disciplined Subset of PL/I.” Technical Report TR76/293. Department of Computer Science, Cornell University, Ithaca, NY.

45. Cornell University. (1977) PL/C User's Guide, Release 7.6. Department of Computer Science, Cornell University, Ithaca, NY.

46. Correa, N. (1992) "Empty Categories, Chain Binding, and Parsing." In Principle—Based Parsing, R. C. Berwick, S. P. Abney, and C. Tenny (eds.). Kluwer Academic Publishers, Boston, pp. 83–121.

47. Cousineau, G., M. Mauny, and K. Callaway. (1998) The Functional Approach to Programming. Cambridge University Press, Cambridge, UK.

48. Dahl, O.-J., E. W. Dijkstra, and C. A. R. Hoare. (1972) Structured Programming. Academic Press, New York.

49. Dahl, O.-J., and K. Nygaard. (1967) SIMULA 67 Common Base Proposal. Norwegian Computing Center Document, Oslo.

50. Deliyanni, A., and R. A. Kowalski. (1979) "Logic and Semantic Networks." Commun. ACM, Vol. 22, No. 3, pp. 184–192.

51. Department of Defense. (1960) COBOL, Initial Specifications for a Common Business Oriented Language. U.S. Department of Defense, Washington, D.C.

52. Department of Defense. (1961) COBOL—1961, Revised Specifications for a Common Business Oriented Language. U.S. Department of Defense, Washington, D.C.

53. Department of Defense. (1962) COBOL—1961 EXTENDED, Extended Specifications for a Common Business Oriented Language. U.S. Department of Defense, Washington, D.C.

54. Department of Defense. (1975a) Requirements for High Order Programming Languages, STRAWMAN. July. U.S. Department of Defense, Washington, D.C.

55. Department of Defense. (1975b) Requirements for High Order Programming Languages, WOODENMAN. August. U.S. Department of

Defense, Washington, D.C.

56. Department of Defense. (1976) Requirements for High Order Programming Languages, TINMAN. June. U.S. Department of Defense, Washington, D.C.

57. Department of Defense. (1977) Requirements for High Order Programming Languages, IRONMAN. January. U.S. Department of Defense, Washington, D.C.

58. Department of Defense. (1978) Requirements for High Order Programming Languages, STEELMAN. June. U.S. Department of Defense, Washington, D.C.

59. Department of Defense. (1980) Requirements for High Order Programming Languages, STONEMAN. February. U.S. Department of Defense, Washington, D.C.

60. DeRemer, F. (1971) “Simple LR(k) Grammars.” Commun. ACM, Vol. 14, No. 7, pp. 453–460.

61. DeRemer, F., and T. Pennello. (1982) “Efficient Computation of LALR(1) Look-Ahead Sets.” ACM TOPLAS, Vol. 4, No. 4, pp. 615–649.

62. Deutsch, L. P., and D. G. Bobrow. (1976) “An Efficient Incremental Automatic Garbage Collector.” Commun. ACM, Vol. 11, No. 3, pp. 522–526.

63. Dijkstra, E. W. (1968a) “Goto Statement Considered Harmful.” Commun. ACM, Vol. 11, No. 3, pp. 147–149.

64. Dijkstra, E. W. (1968b) “Cooperating Sequential Processes.” In Programming Languages, F. Genuys (ed.). Academic Press, New York, pp. 43–112.

65. Dijkstra, E. W. (1972) “The Humble Programmer.” Commun. ACM, Vol. 15, No. 10, pp. 859–866.

66. Dijkstra, E. W. (1975) “Guarded Commands, Nondeterminacy, and Formal Derivation of Programs.” Commun. ACM, Vol. 18, No. 8, pp. 453–457.

67. Dijkstra, E. W. (1976) A Discipline of Programming. Prentice Hall, Englewood Cliffs, NJ.

68. Dybvig, R. K. (2011) The Scheme Programming Language, 4e. MIT Press, Boston.

69. Ellis, M. A., and B. Stroustrup. (1990) The Annotated C++ Reference Manual. Addison-Wesley, Reading, MA.

70. Farber, D. J., R. E. Griswold, and I. P. Polonsky. (1964) “SNOBOL, a String Manipulation Language.” J. ACM, Vol. 11, No. 1, pp. 21–30.

71. Farrow, R. (1982) “LINGUIST 86: Yet Another Translator Writing System Based on Attribute Grammars.” ACM SIGPLAN Notices, Vol. 17, No. 6, pp. 160–171.

72. Fischer, C. N., G. F. Johnson, J. Mauney, A. Pal, and D. L. Stock. (1984) “The Poe Language-Based Editor Project.” ACM SIGPLAN Notices, Vol. 19, No. 5, pp. 21–29.

73. Fischer, C. N., and R. J. LeBlanc. (1977) UW-Pascal Reference Manual. Madison Academic Computing Center, Madison, WI.

74. Fischer, C. N., and R. J. LeBlanc. (1980) “Implementation of Runtime Diagnostics in Pascal.” IEEE Transactions on Software Engineering, Vol. SE-6, No. 4, pp. 313–319.

75. Fischer, C. N., and R. J. LeBlanc. (1991) Crafting a Compiler with C. Benjamin-Cummings, Menlo Park, CA.

76. Flanagan, D. (2011) JavaScript: The Definitive Guide, 6e. O’Reilly Media, Sebastopol, CA.

77. Flanagan, D., and Y. Matsumoto. (2008) The Ruby Programming Language, O’Reilly Media, Sebastopol, CA.

78. Floyd, R. W. (1967) “Assigning Meanings to Programs.” Proceedings Symposium Applied Mathematics. Mathematical Aspects of Computer Science, J. T. Schwartz (ed.). American Mathematical Society, Providence, RI.

79. Frege, G. (1892) “Über Sinn und Bedeutung.” Zeitschrift für Philosophie und Philosophisches Kritik, Vol. 100, pp. 25–50.

80. Friedl, J. E. F. (2006) Mastering Regular Expressions, 3e. O’Reilly Media, Sebastopol, CA.

81. Friedman, D. P., and D. S. Wise. (1979) “Reference Counting’s Ability to Collect Cycles Is Not Insurmountable.” Information Processing Letters, Vol. 8, No. 1, pp. 41–45.

82. Fuchi, K. (1981) “Aiming for Knowledge Information Processing Systems.” Proceedings of the International Conference on Fifth Generation Computing Systems. Japan Information Processing Development Center, Tokyo. Republished (1982) by North-Holland Publishing, Amsterdam.

83. Gehani, N. (1983) Ada: An Advanced Introduction. Prentice Hall, Englewood Cliffs, NJ.

84. Gilman, L., and A. J. Rose. (1983) APL: An Interactive Approach, 3e. John Wiley, New York.

85. Goodenough, J. B. (1975) “Exception Handling: Issues and Proposed Notation.” Commun. ACM, Vol. 18, No. 12, pp. 683–696.

86. Goos, G., and J. Hartmanis (eds.). (1983) The Programming Language Ada Reference Manual. American National Standards Institute. ANSI/-MIL-STD-1815-A–1983. Lecture Notes in Computer Science 155. Springer-Verlag, New York.

87. Gordon, M. (1979) The Denotational Description of Programming Languages, An Introduction. Springer-Verlag, New York.

88. Graham, P. (1996) ANSI Common LISP. Prentice Hall, Englewood Cliffs, NJ.

89. Gries, D. (1981) The Science of Programming. Springer-Verlag, New York.

90. Halstead, R. H., Jr. (1985) “Multilisp: A Language for Concurrent Symbolic Computation.” ACM Transactions on Programming Language and Systems, Vol. 7, No. 4, October 1985, pp. 501–538.

91. Halvorson, M. (2013) Microsoft Visual Basic 2013 Step by Step. Microsoft Press, Redmond, WA.

92. Hammond, P. (1983) APES: A User Manual. Department of Computing Report 82/9. Imperial College of Science and Technology, London.

93. Harbison, S. P., III, and G. L. Steele, Jr. (2002) C: A Reference Manual, 5e. Prentice Hall, Upper Saddle River, NJ.

94. Henderson, P. (1980) Functional Programming: Application and Implementation. Prentice Hall, Englewood Cliffs, NJ.

95. Hoare, C. A. R. (1969) “An Axiomatic Basis of Computer Programming.” Commun. ACM, Vol. 12, No. 10, pp. 576–580.

96. Hoare, C. A. R. (1972) “Proof of Correctness of Data Representations.” Acta Informatica, Vol. 1, pp. 271–281.

97. Hoare, C. A. R. (1973) “Hints on Programming Language Design.” Proceedings ACM SIGACT/SIGPLAN Conference on Principles of Programming Languages. Also published as Technical Report STAN-CS-73-403, Stanford University Computer Science Department.

98. Hoare, C. A. R. (1974) “Monitors: An Operating System Structuring Concept.” Commun. ACM, Vol. 17, No. 10, pp. 549–557.

99. Hoare, C. A. R. (1978) “Communicating Sequential Processes.” Commun. ACM, Vol. 21, No. 8, pp. 666–677.

100. Hoare, C. A. R. (1981) “The Emperor’s Old Clothes.” Commun. ACM, Vol. 24, No. 2, pp. 75–83.

101. Hoare, C. A. R., and N. Wirth. (1973) “An Axiomatic Definition of the Programming Language Pascal.” Acta Informatica, Vol. 2, pp. 335–355.

102. Hogger, C. J. (1984) Introduction to Logic Programming. Academic Press, London.

103. Hogger, C. J. (1991) Essentials of Logic Programming. Oxford Science Publications, Oxford, England.

104. Holt, R. C., G. S. Graham, E. D. Lazowska, and M. A. Scott. (1978) Structured Concurrent Programming with Operating Systems Applications. Addison-Wesley, Reading, MA.

105. Horn, A. (1951) “On Sentences Which Are True of Direct Unions of Algebras.” J. Symbolic Logic, Vol. 16, pp. 14–21.

106. Hudak, P., and J. Fasel. (1992) “A Gentle Introduction to Haskell.” ACM SIGPLAN Notices, Vol. 27, No. 5, May 1992, pp. T1–T53.

107. Hughes, J. (1989) “Why Functional Programming Matters.” The Computer Journal, Vol. 32, No. 2, pp. 98–107.

108. Huskey, H. K., R. Love, and N. Wirth. (1963) “A Syntactic Description of BC NELIAC.” Commun. ACM, Vol. 6, No. 7, pp. 367–375.

109. IBM. (1954) “Preliminary Report, Specifications for the IBM Mathematical FORmula TRANslating System, FORTRAN.” IBM Corporation, New York.

110. IBM. (1956) “Programmer’s Reference Manual, The FORTRAN Automatic Coding System for the IBM 704 EDPM.” IBM Corporation, New York.

111. IBM. (1964) The New Programming Language. IBM UK Laboratories, Hursley, England.

12. Ichbiah, J. D., J. C. Heliard, O. Roubine, J. G. P. Barnes, B. Krieg-Brueckner, and B. A. Wichmann. (1979) “Part B: Rationale for the Design of the Ada Programming Language.” ACM SIGPLAN Notices, Vol. 14, No. 6.

13. IEEE. (1985) “Binary Floating-Point Arithmetic.” IEEE Standard 754, IEEE, New York.

14. INCITS/ISO/IEC. (1997) 1539-1-1997, Information Technology—Programming Languages—FORTRAN, Part 1: Base Language. American National Standards Institute, New York.

15. Ingerman, P. Z. (1967). “Panini-Backus Form Suggested.” Commun. ACM, Vol. 10, No. 3, p. 137.

16. ISO. (1998) ISO14882-1, ISO/IEC Standard – Information Technology —Programming Language—C++. International Organization for Standardization, Geneva, Switzerland.

17. ISO. (1999) ISO/IEC 9899:1999, Programming Language C. American National Standards Institute, New York.

18. ISO/IEC. (1996) 14977:1996, Information Technology—Syntactic Metalanguage—Extended BNF. International Organization for Standardization, Geneva, Switzerland.

19. ISO/IEC. (2002) 1989:2002, Information Technology—Programming Languages—COBOL. American National Standards Institute, New York.

20. ISO/IEC. (2010) 1539-1, Information Technology—Programming Languages—Fortran. American National Standards Institute, New York.

21. ISO/IEC. (2014) 8652/2012(E), Ada 2012 Reference Manual. Springer-Verlag, New York.

22. Iverson, K. E. (1962) A Programming Language. John Wiley, New York.

23. Jensen, K., and N. Wirth. (1974) Pascal Users Manual and Report. Springer-Verlag, Berlin.

24. Johnson, S. C. (1975) “Yacc: Yet Another Compiler-Compiler.” Computing Science Report 32. AT&T Bell Laboratories, Murray Hill, NJ.

25. Jones, N. D. (ed.). (1980) Semantic-Directed Compiler Generation. Lecture Notes in Computer Science, Vol. 94. Springer-Verlag, Heidelberg, FRG.

26. Kay, A. (1969) The Reactive Engine. PhD Thesis. University of Utah, September.

27. Kernighan, B. W., and D. M. Ritchie. (1978) The C Programming Language. Prentice Hall, Englewood Cliffs, NJ.

28. Knuth, D. E. (1965) “On the Translation of Languages from Left to Right.” Information & Control, Vol. 8, No. 6, pp. 607–639.

29. Knuth, D. E. (1967) “The Remaining Trouble Spots in ALGOL 60.” Commun. ACM, Vol. 10, No. 10, pp. 611–618.

30. Knuth, D. E. (1968) “Semantics of Context-Free Languages.” Mathematical Systems Theory, Vol. 2, No. 2, pp. 127–146.

31. Knuth, D. E. (1974) “Structured Programming with GOTO Statements.” ACM Computing Surveys, Vol. 6, No. 4, pp. 261–301.

32. Knuth, D. E. (1981) The Art of Computer Programming, Vol. II, 2e. Addison-Wesley, Reading, MA.

33. Knuth, D. E., and L. T. Pardo. (1977) “Early Development of Programming Languages.” In Encyclopedia of Computer Science and Technology, G. Holzman and A. Kent (eds.). Vol. 7. Dekker, New York, pp. 419–493.

34. Kowalski, R. A. (1979) Logic for Problem Solving. Artificial Intelligence Series, Vol. 7. Elsevier-North Holland, New York.

135. Laning, J. H., Jr., and N. Zierler. (1954) “A Program for Translation of Mathematical Equations for Whirlwind I.” Engineering memorandum E-364. Instrumentation Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

136. Ledgard, H. F., and M. Marcotty. (1975) “A Genealogy of Control Structures.” Commun. ACM, Vol. 18, No. 11, pp. 629–639.

137. Lippman, S. B., and J. Lajoie. (2012) C++ Primer, 5e. Addison-Wesley, Upper Saddle River, NJ.

138. Lischner, R. (2000) Delphi in a Nutshell. O’Reilly Media, Sebastopol, CA.

139. Liskov, B., R. L. Atkinson, T. Bloom, J. E. B. Moss, C. Scheffert, R. Scheifler, and A. Snyder. (1981) CLU Reference Manual. Springer, New York.

140. Lomet, D. (1975) “Scheme for Invalidating References to Freed Storage.” IBM Journal of Research and Development, Vol. 19, pp. 26–35.

141. Lutz, M. (2013) Learning Python, 5e. O’Reilly Media, Sebastopol, CA.

142. MacLaren, M. D. (1977) “Exception Handling in PL/I.” ACM SIGPLAN Notices, Vol. 12, No. 3, pp. 101–104.

143. Marcotty, M., H. F. Ledgard, and G. V. Bochmann. (1976) “A Sampler of Formal Definitions.” ACM Computing Surveys, Vol. 8, No. 2, pp. 191–276.

144. Mather, D. G., and S. V. Waite (eds.). (1971) BASIC, 6e. University Press of New England, Hanover, NH.

145. McCarthy, J. (1960) “Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I.” Commun. ACM, Vol. 3, No. 4, pp. 184–195.

146. McCarthy, J., P. W. Abrahams, D. J. Edwards, T. P. Hart, and M. Levin. (1965) LISP 1.5 Programmer's Manual, 2e. MIT Press, Cambridge, MA.

147. McCracken, D. (1970) "Whither APL." Datamation, September 15, pp. 53–57.

148. Metcalf, M., J. Reid, and M. Cohen. (2004) Fortran 95/2003 Explained, 3e. Oxford University Press, Oxford, England.

149. Meyer, B. (1990) Introduction to the Theory of Programming Languages. Prentice Hall, Englewood Cliffs, NJ.

150. Milner, R., R. Harper, and M. Tofle. (1997) The Definition of Standard ML-Revised. MIT Press, Cambridge, MA.

151. Milos, D., U. Pleban, and G. Loegel. (1984) "Direct Implementation of Compiler Specifications." POPL '84 Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Programming Languages, pp. 196–202.

152. Mitchell, J. G., W. Maybury, and R. Sweet. (1979) Mesa Language Manual, Version 5.0, CSL-79-3. Xerox Research Center, Palo Alto, CA.

153. Moss, C. (1994) Prolog++: The Power of Object-Oriented and Logic Programming. Addison-Wesley, Reading, MA.

154. Moto-oka, T. (1981) "Challenge for Knowledge Information Processing Systems." Proceedings of the International Conference on Fifth Generation Computing Systems. Japan Information Processing Development Center, Tokyo. Republished (1982) by North-Holland Publishing, Amsterdam.

155. Naur, P. (ed.). (1960) "Report on the Algorithmic Language ALGOL 60." Commun. ACM, Vol. 3, No. 5, pp. 299–314.

156. Newell, A., and H. A. Simon. (1956) "The Logic Theory Machine—A Complex Information Processing System." IRE Transactions on Information Theory, Vol. IT-2, No. 3, pp. 61–79.

157. Newell, A., and F. M. Tonge. (1960) “An Introduction to Information Processing Language V.” Commun. ACM, Vol. 3, No. 4, pp. 205–211.

158. Nilsson, N. J. (1971) Problem Solving Methods in Artificial Intelligence. McGraw-Hill, New York.

159. Pagan, F. G. (1981) Formal Specifications of Programming Languages. Prentice Hall, Englewood Cliffs, NJ.

160. Papert, S. (1980) MindStorms: Children, Computers and Powerful Ideas. Basic Books, New York.

161. Perlis, A., and K. Samelson. (1958) “Preliminary Report—International Algebraic Language.” Commun. ACM, Vol. 1, No. 12, pp. 8–22.

162. Peyton Jones, S. L. (1987) The Implementation of Functional Programming Languages. Prentice Hall, Englewood Cliffs, NJ.

163. Pratt, T. W., and M. V. Zelkowitz. (2001) Programming Languages: Design and Implementation, 4e. Prentice Hall, Englewood Cliffs, NJ.

164. Remington-Rand. (1952) “UNIVAC Short Code.” Unpublished collection of dittoed notes. Preface by A. B. Tonik, dated October 25, 1955 (1 p.); Preface by J. R. Logan, undated but apparently from 1952 (1 p.); Preliminary exposition, 1952? (22 pp., in which pp. 20–22 appear to be a later replacement); Short code supplementary information, topic one (7 pp.); Addenda #1, 2, 3, 4 (9 pp.).

165. Reppy, J. H. (1999) Concurrent Programming in ML. Cambridge University Press, New York.

166. Richards, M. (1969) “BCPL: A Tool for Compiler Writing and Systems Programming.” Proc. AFIPS SJCC, Vol. 34, pp. 557–566.

167. Robbins, A. (2005) Unix in a Nutshell, 4e. O’Reilly Media, Sebastopol, CA.

168. Robinson, J. A. (1965) “A Machine-Oriented Logic Based on the Resolution Principle.” Journal of the ACM, Vol. 12, pp. 23–41.

169. Roussel, P. (1975) “PROLOG: Manual de Reference et D’utilisation.” Research Report. Artificial Intelligence Group, University of Aix-Marseille, Luming, France.

170. Rubin, F. (1987) “‘GOTO Statement Considered Harmful’ considered harmful” (letter to editor). Commun. ACM, Vol. 30, No. 3, pp. 195–196.

171. Rutishauser, H. (1967) Description of ALGOL 60. Springer-Verlag, New York.

172. Sammet, J. E. (1969) Programming Languages: History and Fundamentals. Prentice Hall, Englewood Cliffs, NJ.

173. Sammet, J. E. (1976) “Roster of Programming Languages for 1974–75.” Commun. ACM, Vol. 19, No. 12, pp. 655–669.

174. Schorr, H., and W. Waite. (1967) “An Efficient Machine Independent Procedure for Garbage Collection in Various List Structures.” Commun. ACM, Vol. 10, No. 8, pp. 501–506.

175. Scott, D. S., and C. Strachey. (1971) “Towards a Mathematical Semantics for Computer Language.” In Proceedings, Symposium on Computers and Automation, J. Fox (ed.). Polytechnic Institute of Brooklyn Press, New York, pp. 19–46.

176. Scott, M. (2009) Programming Language Pragmatics, 3e. Morgan Kaufman, San Francisco, CA.

177. Sebesta, R. W. (1991) VAX Structured Assembly Language Programming, 2e. Benjamin/Cummings, Redwood City, CA.

178. Sergot, M. J. (1983) “A Query-the-User Facility for Logic Programming.” In Integrated Interactive Computer Systems, P. Degano and E. Sandewall (eds.). North-Holland Publishing, Amsterdam.

179. Shaw, C. J. (1963) “A Specification of JOVIAL.” Commun. ACM, Vol. 6, No. 12, pp. 721–736.

180. Smith, J. B. (2006) Practical OCaml. Apress, Springer-Verlag, New York.

181. Sommerville, I. (2010) Software Engineering, 9e. Addison-Wesley, Reading, MA.

182. Steele, G. L., Jr. (1990) Common LISP The Language, 2e. Digital Press, Burlington, MA.

183. Stoy, J. E. (1977) Denotational Semantics: The Scott–Strachey Approach to Programming Language Semantics. MIT Press, Cambridge, MA.

184. Stroustrup, B. (1983) “Adding Classes to C: An Exercise in Language Evolution.” Software—Practice and Experience, Vol. 13, pp. 139–161.

185. Stroustrup, B. (1984) “Data Abstraction in C.” AT&T Bell Laboratories Technical Journal, Vol. 63, No. 8, pp. 1701–1732.

186. Stroustrup, B. (1986) The C++ Programming Language. Addison-Wesley, Reading, MA.

187. Stroustrup, B. (1988) “What Is Object-Oriented Programming?” IEEE Software, May 1988, pp. 10–20.

188. Stroustrup, B. (1994) The Design and Evolution of C++. Addison-Wesley, Reading, MA.

189. Stroustrup, B. (1997) The C++ Programming Language, 3e. Addison-Wesley, Reading, MA.

190. Sussman, G. J., and G. L. Steele, Jr. (1975) “Scheme: An Interpreter for Extended Lambda Calculus.” MIT AI Memo No. 349 (December 1975).

191. Suzuki, N. (1982) “Analysis of Pointer ‘Rotation’.” Commun. ACM, Vol. 25, No. 5, pp. 330–335.

192. Syme, D., A. Granicz, and A. Cisternino. (2010) Expert F# 2.0. Apress, Springer-Verlag, New York.

193. Tatroe, K., P. MacIntyre, and R. Lerdorf. (2013) Programming PHP, 3e. O'Reilly Media, Sebastopol, CA.

194. Tanenbaum, A. S. (2005) Structured Computer Organization, 5e. Prentice Hall, Englewood Cliffs, NJ.

195. Teitelbaum, T., and T. Reps. (1981) "The Cornell Program Synthesizer: A Syntax-Directed Programming Environment." Commun. ACM, Vol. 24, No. 9, pp. 563–573.

196. Tenenbaum, A. M., Y. Langsam, and M. J. Augenstein. (1990) Data Structures Using C. Prentice Hall, Englewood Cliffs, NJ.

197. Thomas, D., A. Hunt, and C. Fowler. (2013) Programming Ruby 1.9 & 2.0: The Pragmatic Programmers Guide (The Facets of Ruby). The Pragmatic Bookshelf, Raleigh, NC.

198. Thompson, S. (1999) Haskell: The Craft of Functional Programming, 2e. Addison-Wesley, Reading, MA.

199. Turner, D. (1986) "An Overview of Miranda." ACM SIGPLAN Notices, Vol. 21, No. 12, pp. 158–166.

200. Ullman, J. D. (1998) Elements of ML Programming. ML97 edition. Prentice Hall, Englewood Cliffs, NJ.

201. van Emden, M. H. (1980) "McDermott on Prolog: A Rejoinder." SIGART Newsletter, No. 72, August, pp. 19–20.

202. van Wijngaarden, A., B. J. Mailloux, J. E. L. Peck, and C. H. A. Koster. (1969) "Report on the Algorithmic Language ALGOL 68." Numerische Mathematik, Vol. 14, No. 2, pp. 79–218.

203. Wadler, P. (1998) "Why No One Uses Functional Languages." ACM SIGPLAN Notices, Vol. 33, No. 2, February 1998, pp. 25–30.

204. Warren, D. H. D., L. M. Pereira, and F. C. N. Pereira. (1979) "User's Guide to DEC System-10 Prolog." Occasional Paper 15. Department of Artificial Intelligence, University of Edinburgh, Scotland.

205. Watt, D. A. (1979) “An Extended Attribute Grammar for Pascal.” ACM SIGPLAN Notices, Vol. 14, No. 2, pp. 60–74.

206. Wegner, P. (1972) “The Vienna Definition Language.” ACM Computing Surveys, Vol. 4, No. 1, pp. 5–63.

207. Weissman, C. (1967) LISP 1.5 Primer. Dickenson Press, Belmont, CA.

208. Wexelblat, R. L. (ed.). (1981) History of Programming Languages. Academic Press, New York.

209. Wheeler, D. J. (1950) “Programme Organization and Initial Orders for the EDSAC.” Proc. R. Soc. London, Ser. A, Vol. 202, pp. 573–589.

210. Wilkes, M. V. (1952) “Pure and Applied Programming.” In Proceedings of the ACM National Conference, Vol. 2. Toronto, pp. 121–124.

211. Wilkes, M. V., D. J. Wheeler, and S. Gill. (1951) The Preparation of Programs for an Electronic Digital Computer, with Special Reference to the EDSAC and the Use of a Library of Subroutines. Addison-Wesley, Reading, MA.

212. Wilkes, M. V., D. J. Wheeler, and S. Gill. (1957) The Preparation of Programs for an Electronic Digital Computer, 2e. Addison-Wesley, Reading, MA.

213. Wilson, P. R. (2005) “Uniprocessor Garbage Collection Techniques.” Available at http://www.cs.utexas.edu/users/oops/papers.htm#bigsurv.

214. Wirth, N. (1971) “The Programming Language Pascal.” Acta Informatica, Vol. 1, No. 1, pp. 35–63.

215. Wirth, N. (1973) Systematic Programming: An Introduction. Prentice Hall, Englewood Cliffs, NJ.

216. Wirth, N. (1975) “On the Design of Programming Languages.” Information Processing 74 (Proceedings of IFIP Congress 74). North Holland, Amsterdam, pp. 386–393.

217. Wirth, N. (1977) “Modula: A Language for Modular Multi-Programming.” Software—Practice and Experience, Vol. 7, pp. 3–35.

218. Wirth, N., and C. A. R. Hoare. (1966) “A Contribution to the Development of ALGOL.” Commun. ACM, Vol. 9, No. 6, pp. 413–431.

219. Zuse, K. (1972) “Der Plankalkül.” Manuscript prepared in 1945, published in Berichte der Gesellschaft für Mathematik und Datenverarbeitung, No. 63 (Bonn, 1972); Part 3, 285 pp. English translation of all but pp. 176–196 in No. 106 (Bonn, 1976), pp. 42–244.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Contents

List of Illustrations

List of Tables

Landmarks

40. 16
41. 17
42. 18
43. 19
44. 20
45. 21
46. 22
47. 23
48. 24
49. 25
50. 26
51. 27
52. 28
53. 29
54. 30
55. 31
56. 32
57. 33
58. 34
59. 35
60. 36
61. 37
62. 38
63. 39
64. 40
65. 41
66. 42
67. 43
68. 44
69. 45
70. 46
71. 47
72. 48
73. 49
74. 50
75. 51
76. 52

77. 53
78. 54
79. 55
80. 56
81. 57
82. 58
83. 59
84. 60
85. 61
86. 62
87. 63
88. 64
89. 65
90. 66
91. 67
92. 68
93. 69
94. 70
95. 71
96. 72
97. 73
98. 74
99. 75
100. 76
101. 77
102. 78
103. 79
104. 80
105. 81
106. 82
107. 83
108. 84
109. 85
110. 86
111. 87
112. 88
113. 89

14. 90
15. 91
16. 92
17. 93
18. 94
19. 95
20. 96
21. 97
22. 98
23. 99
24. 100
25. 101
26. 102
27. 103
28. 104
29. 105
30. 106
31. 107
32. 108
33. 109
34. 110
35. 111
36. 112
37. 113
38. 114
39. 115
40. 116
41. 117
42. 118
43. 119
44. 120
45. 121
46. 122
47. 123
48. 124
49. 125
50. 126

225. 201
226. 202
227. 203
228. 204
229. 205
230. 206
231. 207
232. 208
233. 209
234. 210
235. 211
236. 212
237. 213
238. 214
239. 215
240. 216
241. 217
242. 218
243. 219
244. 220
245. 221
246. 222
247. 223
248. 224
249. 225
250. 226
251. 227
252. 228
253. 229
254. 230
255. 231
256. 232
257. 233
258. 234
259. 235
260. 236
261. 237

521. 497
522. 498
523. 499
524. 500
525. 501
526. 502
527. 503
528. 504
529. 505
530. 506
531. 507
532. 508
533. 509
534. 510
535. 511
536. 512
537. 513
538. 514
539. 515
540. 516
541. 517
542. 518
543. 519
544. 520
545. 521
546. 522
547. 523
548. 524
549. 525
550. 526
551. 527
552. 528
553. 529
554. 530
555. 531
556. 532
557. 533

595. 571
596. 572
597. 573
598. 574
599. 575
600. 576
601. 577
602. 578
603. 579
604. 580
605. 581
606. 582
607. 583
608. 584
609. 585
610. 586
611. 587
612. 588
613. 589
614. 590
615. 591
616. 592
617. 593
618. 594
619. 595
620. 596
621. 597
622. 598
623. 599
624. 600
625. 601
626. 602
627. 603
628. 604
629. 605
630. 606
631. 607

706. 682
707. 683
708. 684
709. 685
710. 686
711. 687
712. 688
713. 689
714. 690
715. 691
716. 692
717. 693
718. 694
719. 695
720. 696
721. 697
722. 698
723. 699
724. 700
725. 701
726. 702
727. 703
728. 704
729. 705
730. 706
731. 707
732. 708
733. 709
734. 710
735. 711
736. 712
737. 713
738. 714
739. 715
740. 716
741. 717
742. 718

743. 719
744. 720
745. 721
746. 722
747. 723
748. 724
749. 725
750. 726
751. 727
752. 728
753. 729
754. 730
755. 731
756. 732
757. 733
758. 734
759. 735
760. 736
761. 737
762. 738
763. 739
764. 740
765. 741
766. 742
767. 743
768. 744
769. 745
770. 746
771. 747
772. 748
773. 749
774. 750
775. 751
776. 752
777. 753
778. 754
779. 755

The architecture consists of two units, the memory unit and the central processing unit. The central processing consists of the Arithmetic and logic unit and the control unit. The control unit sends information to the arthmetic and logic unit. The input and output devices are connected to this unit. The result of the operations in the C P U is given as input to the memory unit. This unit stores both instructions and data and passes on the output to the C P U.

The first layer is the bare machine. The second layer is the macro instruction. The third layer is the operating system. The fourth layer is divided into 8 sections: A d a compiler labeled virtual a d a computer; Java virtual machine labeled virtual java computer; C compiler labeled virtual C computer; dot NET common language run time; Scheme interpreter labeled virtual scheme computer; operating system command interpreter; assembler labeled virtual assembly language computer; and an incomplete section which indicates additional items in this layer. The fifth incomplete layer includes; the java virtual computer consists of the java compiler and java compiler; the dot NET common language run time unit consists of V B dot NET compiler which is in a virtual v b dot net computer and C hash compiler labeled Virtual C hash computer.

The source program is fed into the lexical analyzer. These lexical units are passed to the syntax analyzer. The lexical analyzer and the syntax analyzer are also given as input to the symbol table. The parse trees from the syntax analyzer and the output from the symbol table is fed into the intermediate code generator and semantic analyzer. The output of this semantic analyzer is sent for optional optimization and passed to the code generator. The intermediate code generated and the output from the symbol table is given to the code generator. The code machine language generated by the code generator and input data are given to the computer which produces the results.

The source program is fed into the lexical analyzer. These lexical units are passed to the syntax analyzer. The parse trees from the analyzer are fed into the intermediate code generator. The intermediate code and input data are passed into the interpreter. The output from the interpreter is the result.

In 19 57, Fortran 1 was created which lead to the frotran continuations as well as inspiring the ALGOL programing. In 19 58, Fortran 2 was created which lead to FORTRAN 4 in 1962. In 1978, Fortran 77. In 19 90 Fortran 90. In 19 95 Fortran 95. In 2003 Fortran 2003. In 2008 Fortran 2008. In 20 15 Fortran 2015.In 19 58, ALGOL 58 was inspired by Fortran 1. And continued in it's series in 1960 with ALGOL 60. In 1966, ALGOL W which developed into pascal in 19 71, pascal in turn inspired 3 products, MODULA 2, M L, and Ada 83. On a different branch in 1968: ALGOL 68 was developed from ALGOL 60. C was partially inspired by ALGOL 68 in 19 71. ALGOL 60 inspired Basic in 19 64 which lead to quick basic in 19 88 then visual basic in 19 90 and finally visual basic dot NET in 2001. From Pascal in 19 71 there are 3 product lines, MODULA 2, M L, and Ada 83. MODULA 2 lead to oberon in 19 88. Modula 2 lead to Modula 3 in 19 88, Modela 3 partially inspired python in 19 92. In 1963, SIMULA 1 was created, inspired by ALGOL 60. In 1967, SIMULA 67. In 19 80 small talk 80. Small talk partially inspired both objective c in 19 84,which lead to swift in 20 14, and ruby in 19 94. Simula 67 along with Ada 83 both inspried Eiffel in 19 90. In 19 57 Flow matic was created which lead to COBOL in 19 60 then P L slash I in 19 64. In 19 62 C P L was created and lead to B C P L in 19 69. In 19 70 B was created which lead to C in 19 71. The line then splits into two branches. The first branch starts with A N S I, C or C 89 in 19 89 and goes on to Python in 19 91 then python 2 point 0 in 2000 and Python 3 point 0 in 2007. A N S I, C also leads to C 99 in 19 99. The second branch from C goes to C + + and Java in 19 94. Java lead to Java 5 point 0 2004, Java 6 point 0 in 2006, Java 7 point 0 in 2009, and Java 8 point 0 in 20 14. C + + also leads to C hash 2000 then to C hash 2 point 0 in 2006, C hash 3 point 0 in 2007, C hash 4 point 0 in 2009, C hash 5 point 0 in 20 12. In 19 59 L I S P was created. L I SP then inpired two product lines Scheme and M L. Scheme was made in 1975 and lead to commmon l i s p. M L was created in 19 78 an dlead to Miranda in 19 83 then Haskell 19 88. In 19 63 SNOBOL was created which lead to Icon in 19 82. Snowbol also lead to a w k in 19 78 with lead to Perl in 19 86. Pearl lead to P H P in 19 94 and Javascript in 19 96. Pearl also lead to Ruby in 19 94. Ruby lead to Ruby 1.8 in 2004 which lead to Ruby 1.9 in 2009.

The internal representation of the list A, B, C, D: A pointer points to the head node in the list. Four nodes with data parts A, B, C, and D and the next pointer linked to its successor. The node D does not have a successor and the pointer value is null. The internal representation of the list A, B, C, D, E, F, G: A pointer points to the head node A in the list. Node A is linked to Node B. Node B is linked to node D and Node C. The node C does not have a successor and the pointer value is null. Node D points to a dummy node for E with a null value. This node points to the node E which similarly points to a null node and points to Node F. This node F points to node G. The node G does not have a successor and the pointer value is null.

The first level, assign, has 3 branches from left to right, i d which represents A, equals, expression. The second level, expression, has 3 branches from left to right, i d which represents B, asterisk, expression. The third level, expression, has 3 branches from left to right, left parenthesis, expression, right parenthesis. The fourth level, i d, which represents A, plus, expression. This expression denotes the i d which represents C.

The first tree has 3 levels. The first level, assign, has 3 branches from left to right, i d which represents A, equals, expression. The second level, expression, has 3 branches from left to right, expression which represents the i d with attribute B, asterisk, and expression. The third level, expression, has 3 branches from left to right, expression, which represents the i d with attribute C, asterisk, expression, which represents the i d with attribute A. Similarly the second tree has 3 levels. The first level, assign, has 3 branches from left to right, i d which represents A, equals, expression. The second level, expression, has 3 branches from left to right, expression, asterisk, expression which represents the i d with attribute A. The third level, expression, has 3 branches from left to right, expression, which represents the i d with attribute B, +, expression, which represents the i d with attribute C.

The first level, assign, has 3 branches from left to right, i d which represents A, equals, expression. The second level, expression, has 3 branches from left to right, expression which which leads to the term and factor of i d with attribute B; plus; term. The third level, term, has 3 branches from left to right, term, which leads to the factor of i d with attribute C. the i d with attribute C; asterisk; factor of i d with attribute A.

The first level, assign, has 3 branches from left to right, i d which represents A, equals, expression. The second level, expression, has 3 branches from left to right; plus; term which leads to the factor of i d with attribute A. The third level, expression, has 3 branches from left to right, expression which which leads to the term and factor of i d with attribute B; plus; term, which leads to the factor of i d with attribute C.

First parse tree. The if statement in the first level can be expressed using 5 branches in the second level such as if, logic expression, statement, else, statement. The if statement has another if statement branch in level three which has 3 branches in the fourth level such as if, logic expression, statement. Second parse tree. The if statement in the first level can be expressed using 3 branches in the second level such as if, logic expression, statement. The statement has another if statement branch in level three and has 5 branches in the fourth level such as if, logic expression, statement, else, statement.

The first level, assign, has 3 branches to the second level, from left to right, variable with attribute A; equals; expression. This Expression has 3 branches to the third level, from left to right, variable 2 with attribute A, +, variable 3 with attribute B.

The first level, assign, has 3 branches to the second level, from left to right, variable with attribute A; equals; expression. This Expression has 3 branches to the third level, from left to right, variable 2 with attribute A, +, variable 3 with attribute B. The attribute A flows from the actual type to the expected type. The attributes A and B flow from the actual type towards the expression.

The first level, assign, has 3 branches to the second level, from left to right, variable with attribute A; equals; expression. A note beside variable reads actual type = real type. This Expression has 3 branches to the third level, from left to right, variable 2 with attribute A, +, variable 3 with attribute B. A note beside expression reads expected type = real type, actual type = real type. Notes beside variable 2 and variable 3 reads, actual type = real type and actual type = integer type.

The first level binary underscore number has two branches to the second level, bit 0 and binary number. Binary number has two branches to the third level, bit 1 and binary number. Binary number has a branch bit 1 in the fourth level.

The first level binary underscore number has two branches to the second level, bit 0 and binary number. The object 6 is attached to the first level. Binary number has two branches to the third level, bit 1 and binary number. The object 3 is attached to the second level. Binary number has a branch bit 1 in the fourth level. The object 1 is attached to the third level.

The transitions and their transitional behavior are as follows: Start to i d, Letter, add c h a r, get c h a r; Start to i n t, Digit, add c h a r, get c h a r; unknown to done, t points to look up next c h a r, get c h a r. The state i d has a self transition and its behavior reads, letter or digit, add c h a r, get c h a r. The state i n t has a self transition and its behavior reads, digit, add c h a r, get c h a r. A transition from i d reads return lookup left parenthesis lexeme right parenthesis. A transition from i n t reads, return i n t underscore l i t. Similarly a transition from done state reads, return t.

The first level, expression leads to the second level, term. Term has 3 branches to the third level, from left to right, factor, slash, factor which leads to total. The first Factor has 3 branches to the fourth level, from left to right, left parenthesis, expression and right parenthesis. Expression has three branches from left to right, term to factor to sum, +, term to factor to 47.

L R parser has two components Parser code and parser table. The parser code is connected to the parser stack and the parsing table is connected to the input. The parse stack contains the following elements: S sub 0, X sub 1, S sub 1, incomplete list, X sub m, S sub m. A pointer labeled top points towards S sub m. The input coming from parsing table consists of the following elements a sub i, a sub i + 1, blank, incomplete list, a sub n and dollar sign.

state	action, i d	action, +	action, asterisk	action, left parenthesis	action, right parenthesis	action, dollar sign	go to, E	go to, T	go to, F
0	S 5	blank	blank	S 4	blank	blank	1	2	3
1	blank	S 6	blank	blank	blank	accept	blank	blank	blank
2	blank	R 2	S 7	blank	R 2	R 2	blank	blank	blank
3	blank	R 4	R 4	blank	R 4	R 4	blank	blank	blank
4	S 5	blank	blank	S 4	blank	blank	8	2	3
5	blank	R 6	R 6	blank	R 6	R 6	blank	blank	blank
6	S 5	blank	blank	S 4	blank	blank	blank	9	3
7	S 5	blank	blank	S 4	blank	blank	blank	blank	10
8	blank	S 6	blank	blank	S 11	blank	blank	blank	blank
9	blank	R 1	blank	S 7	R 1	R 1	blank	blank	blank
10	blank	R 3	blank	R 3	R 3	R 3	blank	blank	blank
11	blank	R 5	blank	R 5	R 5	R 5	blank	blank	blank

The single precision format is represented using a 1 by 3 grid where the grids represent the following: sign bit, exponent in 8 bits and fraction in 23 bits.
The double precision format is represented using a 1 by 3 where the grids represent the following: sign bit, exponent in 11 bits and fraction in 52 bits.

The column values of the matrix table are: 0, 1, incomplete, j minus 1, j, incomplete, n minus 1. The row values of the matrix table are: 0, 1, incomplete, i minus 1, i, incomplete, m minus 1. The location of i, j is the cell where the column j and row i meet and is marked using a circle with a x in it.

The fields read: record, name, type, incomplete, , name, type, offset and address. There are two sets called field one and field n. Field 1 includes name, type, and offset. Field n includes name, type, and offset.

The p t r pointer with value 7080 points to the block with reference address 7080. This address block holds the value 206. A note beside this block reads an anonymous dynamic variable. This block points towards the j block and the value 206 is assigned to j.

A pointer from block r points towards action 1. Action 1 leads to action 2 and 7. Action 2 leads to action 3 and 6. Action 3 leads to action 4 and 5. Action 7 leads to action 8 and 10. Action 8 leads to action 9 and action 10 lead to actions 11 and 12.The order of the node marking is represented by creating an outline around the actions using dashed lines with an x at each step.

The value of the caller is sub left parenthesis a, b, c right parenthesis and the value of callee is void sub left parenthesis i n t, x, i n t, y, i n t z right parenthesis. The first semantic model displays the caller parameter a, being passed to the callee parameter x, the arrow is labeled call. This is an in mode transmission. The second semantic model displays the callee parameter y being returned to the caller parameter b, the arrow is labeled return. This is an out mode transmission. The third semantic model displays the caller parameter c being passed to the callee parameter z, the arrow is labeled call and the callee parameter z passed back to the caller parameter c, the arrow is labeled return. This is an in out mode transmission.

Three stacks main, stack and function sub with the following stack values: main, w, x, y, z and code; stack, value of a, value of b, value of c and address d; function sub, r e f to a, assign to b, r e f to c, assign to c and r e f to d; This is together grouped as code. The stack operations for the parameter passing methods are as follows: The main stack values w and y are passed to the stack value of a, and value of c. The address of at start is passed to the address d which is returned back to the main stack. The value of b and value of c are returned back to the main stack values x and y. The function sub parameters r e f to a, assign to b, r e f to c, assign to c and r e f to d are passed on to the stack values value of a, value of b, value of c, value of c and address of d respectively.

Illustration a. Two routines A and B where A receives the input, resume, from master. The routine A consists of sequences, resume B and routine B consists of sequences, resume A. The control transfers from resume B to resume A and back from resume A to resume B. The sequence continues.
Illustration b. Two routines A and B where B receives the input, resume, from master. The routine A consists of sequences, resume B and routine B consists of sequences, resume A. The control transfers from resume B to resume A and back from resume A to resume B. The sequence continues.

Two routines A and B where A receives the input, resume, from master. The routine A consists of sequences, resume B and routine B consists of sequences, resume A. The control transfers from resume B to resume A and back from resume A to resume B. The sequences loop back to the first sequence when it gets completed. The control from resume B to resume A is the first resume and the subsequent resume is the control from resume A to resume B.

Data has 1 cell stack in the main sub section and three cell stacks each in the lettered sections. The main section information reads local variables. Section A reads local variables, parameters, and return address. Section B reads local variables, parameters, and return address. Section C reads local variables, parameters, and return address.

From top to bottom, next to the first step reads sum. Next to the second step reads list 4. Next to the third step reads step 3. Next to the fourth step reads step 2. Next to the fifth step reads step 1. Next to the sixth step reads step 0. Next to the seventh step reads part. Next to the eighth step reads total.

At point one there are 2 sections, A R I which means activation record instance for fun 1 and A R I for main. There are 5 cell stacks in fun 1 reading, local, local, parameter, dynamic link and return to main. There is one cell stack in main which reads local. There are variables notated beside the stacks. Next to the top level is t, next to the second level is s, and next to the third level is r. The main section is labeled with variable p. A line runs from dynamic link to the bottom of the main section or sixth step. At point 2 there are 3 sections, A R I for fun 2, A R I for fun 1, and A R I for main. There are 4 cell stacks in fun 2 which read, local, parameter, dynamic link, return to fun 1. There are 5 cell stacks in fun 1 which read, local, local, parameter, dynamic link and return to main. There is one cell stack in main which reads local. There are variables notated beside the stacks. Next to the top level is y and the one below it x. The fifth level variable is t. Next to the sixth level is s, and next to the seventh level is r. Next to the bottom or tenth level is p. Two lines run from both the third line to the top of the main or bottom level and from the eighth level to the bottom of the main or bottom level. At point 3 there are 4 sections, A R I for fun 3, A R I for fun 2, A R I for fun 1, and A R I for main. There are 2 cell stacks in fun 3 which read, parameter and dynamic link. There are 4 cell stacks in fun 2 which read, local, parameter, dynamic link, return to fun 1. There are 5 cell stacks in fun 1 which read, local, local, parameter, dynamic link and return to main. There is one cell stack in main which reads local. There are variables notated beside the stacks. In section fun 3 variable q is next to the parameter level. In section fun 2 variable y is next to local level and variable x is next to level parameter. In section fun 1 variable t is next to the first local level and variable s is next to the second local level. In section main, the variable p is next to the only level. Three lines run from the dynamic link level in section 3 to the bottom of section fun 2, the dynamic link level in section fun 2 to the top of the main or bottom level, and from the dynamic link level in fun 1 to the bottom of the main or bottom level.

At first call there are 2 sections, First A R I for factorial which means first activation record instance for factorial and A R I for main. There are 4 cell stacks for first A R I for factorial, functional value question mark, parameter 3, dynamic link with line to the bottom of A R I main, and return to main. There is one cell stack for A R I for main which is local question mark. The variable n is next to parameter 3 in the first A R I. The top line of first A R I is pointed to by an arrow labeled top. The word value is next to the main or bottom most level. At second call there are 3 sections, second A R I for factorial, First A R I for factorial which means first activation record instance for factorial and A R I for main. There are 4 cell stacks for second A R I for factorial, functional value question mark, parameter 2, dynamic link with a line to the bottom of first A R I section, and return to factorial. There are 4 cell stacks for first A R I for factorial, functional value question mark, parameter 3, dynamic link with line to the bottom of A R I main, and return to main. There is one cell stack for A R I for main which is, local question mark. The variable n is next to parameter 2 in the second A R I and parameter 3 in the first A R I. The top line of second A R I is pointed to by an arrow labeled top. The word value is next to the main or bottom most level. At third call there are 4 sections, third A R I for factorial, second A R I for factorial, First A R I for factorial which means first activation record instance for factorial and A R I for main. There are 4 cell stacks for third A R I for factorial, functional value question mark, parameter 1, dynamic link with a line to the bottom of the second A R I section, and return to factorial. There are 4 cell stacks for second A R I for factorial, functional value question mark, parameter 2, dynamic link with a line to the bottom of first A R I section, and return to factorial. There are 4 cell stacks for first A R I for factorial, functional value question mark, parameter 3, dynamic link with line to the bottom of A R I main, and return to main. There is one cell stack for A R I for main which is, local question mark. The variable n is next to parameter 1 in the third A R I, parameter 2 in the second A R I, and parameter 3 in the first A R I. The top line of third A R I is pointed to by an arrow labeled top. The word value is next to the main or bottom most level.

At position 2 in factorial third call completed there are 4 sections, third A R I for factorial, second A R I for factorial, First A R I for factorial which means first activation record instance for factorial and A R I for main. There are 4 cell stacks for third A R I for factorial, functional value 1, parameter 1, dynamic link with a line to the bottom of the second A R I section, and return to factorial. There are 4 cell stacks for second A R I for factorial, functional value question mark, parameter 2, dynamic link with a line to the bottom of first A R I section, and return to factorial. There are 4 cell stacks for first A R I for factorial, functional value question mark, parameter 3, dynamic link with line to the bottom of A R I main, and return to main. There is one cell stack for A R I for main which is, local question mark. The variable n is next to parameter 1 in the third A R I, parameter 2 in the second A R I, and parameter 3 in the first A R I. The top line of third A R I is pointed to by an arrow labeled top. The word value is next to the main or bottom most level. At position 2 in factorial second call completed call there are 3 sections, second A R I for factorial, First A R I for factorial which means first activation record instance for factorial and A R I for main. There are 4 cell stacks for second A R I for factorial, functional value 2, parameter 2, dynamic link with a line to the bottom of first A R I section, and return to factorial. There are 4 cell stacks for first A R I for factorial, functional value question mark, parameter 3, dynamic link with line to the bottom of A R I main, and return to main. There is one cell stack for A R I for main which is, local question mark. The variable n is next to parameter 2 in the second A R I and parameter 3 in the first A R I. The top line of second A R I is pointed to by an arrow labeled top. The word value is next to the main or bottom most level. At position 2 in factorial first call completed, there are 2 sections, First A R I for factorial and A R I for main. There are 4 cell stacks for first A R I for factorial, functional value 6, parameter 3, dynamic link with line to the bottom of A R I main, and return to main. There is one cell stack for A R I for main which is local question mark. The variable n is next to parameter 3 in the first A R I. The top line of first A R I is pointed to by an arrow labeled top. The word value is next to the main or bottom most level. In position 3 in main final results. There is one cell stack for A R I for main which is local question mark. The top line of first A R I is pointed to by an arrow labeled top. The word value is next to the main or bottom most level.

There are 5 sections labeled A R I for sub 1, A R I for sub 3, A R I for sub 2, A R I for big sub, A R I for main underscore 2. There is a label at the top reading top. Section sub 1 has 5 cell stacks, local, local, dynamic link which has a line that goes to the bottom of sub 3, static link which has a dotted line that goes to the bottom of big sub, and return to sub 3. The variable d is next to the top local and the variable a is next to the second local. Section sub 3 has 5 cell stacks, local, local, dynamic link which has a line that goes to the bottom of sub 2, static link which has a dotted line that goes to the bottom of big sub, and return to sub 2. The variable e is next to the top local and the variable c is next to the second local. Section sub 2 has 5 cell stacks, local, local, dynamic link which has a line that goes to the bottom of sub big sub, static link which has a dotted line that goes to the bottom of big sub, and return to sub big sub. The variable e is next to the top local, the variable b is next to the second local, and the variable x is next to parameter. Section big sub has 6 cell stacks, local, local, local, dynamic link which has a line that goes to the bottom of main, static link, and return to main. The variable c is next to the top local, the variable b is next to the second local, and the variable a is next to the third local. Section main has 1 cell stacks, local. The variable x is next to the local.

Three unlabeled levels proceed the block variables levels of e, d, c, b and g, and a and f. Locals levels read, z, y, and x. The bottom most level is a large shaded black that reads, activation record instance for main.

There are 5 sections labeled A R I for sub 3, A R I for sub 2, A R I for sub 1, A R I for sub 1, A R I for main. Section sub 3 has 4 cell stacks, local, local, dynamic link which has a line that goes to the bottom of sub 2, and return to sub 2. The variable z is next to the top local and the variable x is next to the second local. Section sub 2 has 4 cell stacks, local, local, dynamic link which has a line that goes to the bottom of sub 1, and return to sub 1. The variable w is next to the top local and the variable v is next to the second local. Section sub 1 has 4 cell stacks, local, local, dynamic link which has a line that goes to the bottom of sub 1, and return to main. The variable w is next to the top local and the variable v is next to the second local. Section main has 2 cell stacks, local and local. The variable u is next to the top local and the variable v is next to the second local.

There are 5 stacks that are labeled, u, v, x, z, and w. The names in the stack cells indicate the program units of the variable declaration. U has one cell stack labeled main. V has three cell stacks labeled, sub 1, sub 1, main. X has two cell stacks labeled, sub 3 and sub 2. Z has one cell stack labeled sub 3. W has three cell stacks labeled, sub 2, sub 1, sub 1.

The first block is the Public class A consists of a method draw and an incomplete line of code. The second block is the Public class B extends A consists of a draw method and an incomplete line of code. This block has an upward arrow pointing towards the first block. The client block consists of the following code. Line 1. A my A = new A (); Line 2. my A period draw (); Line 3. Incomplete line of code.

A stack with value variables b 1 and a 1 pointing towards its corresponding objects. variable b 1 points towards its objects X and Y. This is labeled data area. Variable a 1 points towards its object X. This is labeled data area.

The first illustration has three classes Shape, circle and rectangle with values virtual void draw left parenthesis right parenthesis = 0, void draw left parenthesis right parenthesis and void draw left parenthesis right parenthesis. The classes circle and shape are inherited from the class shape. Similarly class square can also be inherited. The second illustration displays the type of the pointers and it’s binding to objects. The shape asterisk pointer is represented as a rectangular block with reference p t r underscore shape binded towards its object square with value void draw left parenthesis right parenthesis. Similarly, The rectangle asterisk pointer is represented as a rectangular block with reference r e c t binded towards its object rectangle with value void draw left parenthesis right parenthesis. The binding is represented using a continuous arrow pointing towards the block. The square asterisk pointer is represented using a dot.

The class instance record for A is represented using a virtual method table with 3 fields, v table pointer, a and b pointing towards a v table for A with two fields. The two fields have pointers towards, code for A’s draw and code for A’s area. Similarly, the class instance record for B is represented using a virtual method table with 5 fields, v table pointer, a, b, c and d pointing towards a v table for B with three fields. The three fields have pointers towards, code for A’s area, code for B’s draw and code for B’sift.

The class instance record for C is represented using a virtual method table with 5 fields, v table pointer, a, v table pointer, b and c. The first v table pointer points to C’s v table for C and A part with 3 fields. The three fields have pointers towards, code for C’s in it, code for C’s fun and code for C’s d u d. The second v table pointer points to C’s v table for B part with one field that points towards code for B’s sum.

The value of total 3 is represented by a continuous line. Values 4 and 6 are written on the line. Task A is presented by a continuous line with markings Fetch TOTAL, Add 1 and Store TOTAL. Similarly, Task B is represented by a continuous line with markings Fetch TOTAL, Multiply by 2 and Store TOTAL. Time is represented by a continuous arrow pointing towards the right.

The flow of the states is as follows: A new state is created; The state then becomes ready with a task; The scheduled task begins running The task with time slice expiration returns back to the ready state; The completed task is sent to the dead state; The input or output from the running state is sent to the blocked state which, when completed goes back to the ready state.

The processes SUB 1 and SUB 3 are given to the insertion block for inserting data and the removed data is fed back to the processes. Insert sends information to buffer and remove received information from buffer.

The first timeline diagram depicts the occurrence, task example waits for sender. The timeline is represented using a continuous arrow pointing towards the right. The task example consists of the following steps in the time line: Wait at accept, Accept and wait at accept. Wait at accept is represented by dashed lines. The sender consists of the following steps, sends message, rendezvous and continue execution. The rendezvous section is represented in dashed lines. The second timeline diagram depicts the occurrence, sender waits for task example. The timeline is represented using a continuous arrow pointing towards the right. The task example consists of the following steps in the time line: Busy, Accept and wait at accept. Wait at accept is represented by dashed lines. The sender consists of the following steps, sends message and is suspended, rendezvous and continue execution. The rendezvous section is represented in dashed lines.

Task A has two jobs 1 and 2 and task B has two jobs 3 and 4, labeled, accept clauses. Both tasks consist of a task body connected to each other by a bidirectional arrow. A note beside reads, B period Job 3 value.

The flow transfers from the executing code to the exception handlers. The executing code consists of the following: Line 1, incomplete. Line 2, begin. Line 3, incomplete. Line 4, some statement. Line 5, incomplete. Line 6, end semicolon. Line 7, incomplete. The exception handlers consist of the following: Line 1, when incomplete line of code. Line 2. begin. Line 3.incomplete. Line 4, end semicolon. The exception is raised in, some statement. The control exception to handler binding, flows from some statement to the when statements. The when then continues back to the some statement in the executing code, the end, two incomplete sections, and a termination on the side.

The internal representation of the list A, B, C, D: A pointer points to the head node in the list. Four nodes with data parts A, B C and D and the next pointer linked to its successor. The node D does not have a successor and the pointer value is null. The internal representation of the list A left parenthesis B C right parenthesis D left parenthesis E left parenthesis F G right parenthesis right parenthesis: A pointer points to the head node A in the list. Node A is linked to Node B. Node B is linked to node D and Node C. The node C does not have a successor and the pointer value is null. Node D points to a dummy node for E with a null value. This node points to the node E which similarly points to a null node and points to Node F. This node F points to node G. The node G does not have a successor and the pointer value is null.

The internal representation of lisp list C O N S single quote A single quote left parenthesis right parenthesis which after construct operation becomes A. A head node with data part value A and null pointer value. The internal representation of lisp list C O N S single quote A single quote left parenthesis B C right parenthesis which after construct operation becomes A B C. Three nodes with data parts A, B and C and the next pointer linked to its successor. The node C does not have a successor and the pointer value is null. The internal representation of lisp list C O N S single quote left parenthesis right parenthesis single quote left parenthesis A Bright parenthesis which after construct operation becomes left parenthesis right parenthesis A B. A head node with data part value pointing towards another node with data part NIL. The head node points to a node with data part A and this node points to a node with data part B with a null pointer. The internal representation of lisp list C O N S single quote A B single quote left parenthesis C D right parenthesis which after construct operation becomes left parenthesis A B right parenthesis C D. The head node points to two nodes with data parts A and C. Node C points to a node with data part D and with a null pointer value. Node A points to a node with data part B and with a null pointer value.

Each goal is depicted by a rectangular box with four ports call, fail, exit and redo. The control enters the block through the call port or the redo port. If the goal succeeds, the control leaves through the exit port. If the goal fails, the control leaves through the fail port. If two sections are connected as in the example the redo and fail sections connect and the exit and call sections connect.

wishful thinking

Time Management

for Creative People

Manage the mundane - create the extraordinary

Mark McGuinness
www.wishfulthinking.co.uk/blog

First published on www.businessofdesignonline.com

Some rights reserved

Web: www.wishfulthinking.co.uk

Blog: www.wishfulthinking.co.uk/blog

E-mail: wish@wishfulthinking.co.uk

Important note about the illustrations

The images in this e-book are licensed from www.istockphoto.com for use within this document. If you wish to use them elswhere you should obtain a licence from www.istockphoto.com

Image credits:

p.1 www.istockphoto.com/ldf

p.4 www.istockphoto.com/demarco-media

p.7 www.istockphoto.com/foued42

p.10 www.istockphoto.com/kativ

p.14 www.istockphoto.com/claylib

p.17 www.istockphoto.com/tmcnem

p.21 www.istockphoto.com/ju-lee

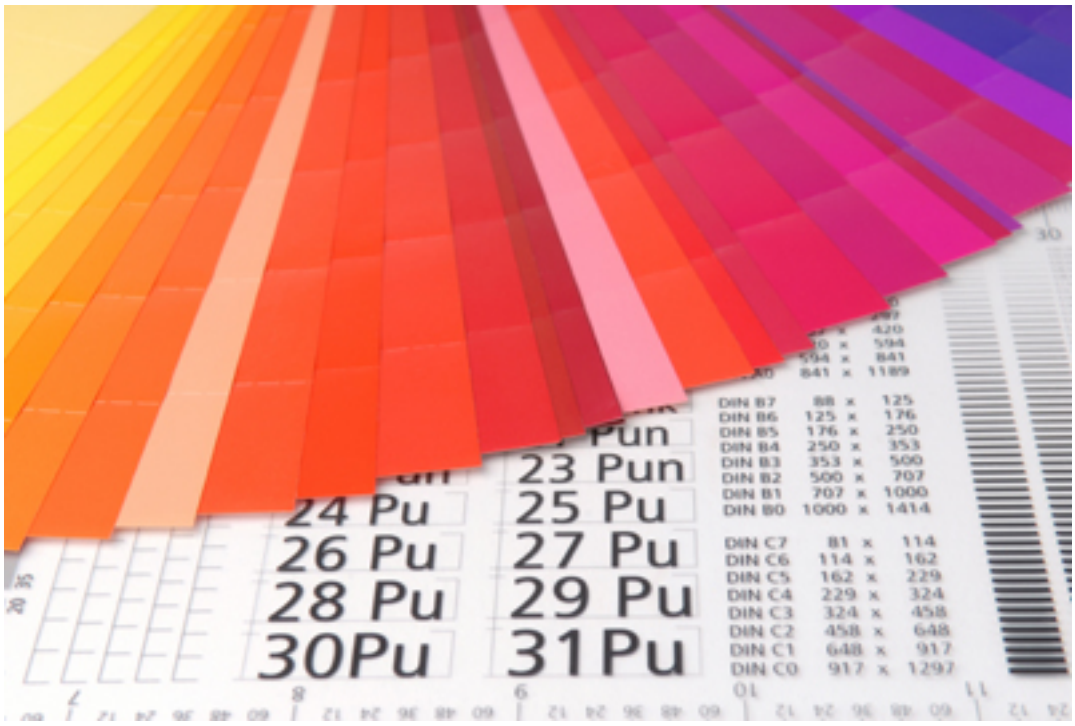
p.25 www.istockphoto.com/leggnet

p.28 www.istockphoto.com/arlindo71

This e-book contains no affiliate links.

Contents

1. Why you need to be organised to be creative



"Be regular and orderly in your life so that you may be violent and original in your work."
Gustave Flaubert

So you start the day full of enthusiasm. You're excited about a new piece of creative work and itching to put your ideas into action. Firing up your computer, the familiar stream of e-mails pours into your inbox, burying the ones you didn't get round to replying to yesterday. Scanning through the list, your heart sinks – two of them look as though they require urgent action. You hit 'reply' and start typing a response to one of them… 20 minutes later you 'come round' and realise you've got sucked into the e-mail zone and have been sidetracked by interesting links sent by friends, as well as writing replies about issues that aren't a priority for you. You minimize the e-mail window and get back to your project…

After 15 minutes you're really enjoying yourself, getting into your creative flow – when the phone rings. Somebody wants something from you. Something to do with a meeting last week. You rummage through the papers on your desk, searching for your notes. You can't find them. Suddenly your heart leaps as you lift up a folder and find an important letter you'd forgotten about – it needed an urgent response, several days ago. 'Hang on, I'll get back to you' you tell the person on the phone, 'I'll ring you back when I've found it'. You put the phone down and pick up

the letter – this needs sorting immediately, but you remember why you put it off – it involves several phone calls and hunting through your files for documents you're not sure you even kept. By now, you've only got half an hour before your first meeting and you've promised to ring that person back. . Your design stares at you reproachfully. The e-mail inbox is pinging away as it fills up – already there are more messages than before you started answering them. Your enthusiasm has nosedived and the day has hardly begun. Creative work seems like a distant dream.

. . .

Is this a familiar scenario for you? Swap the design software for a wordprocessor and I've been there a hundred times. In an ideal world we'd be putting all our time and energy into creative work, but the realities of modern work often seem to be conspiring against us. And in lots of ways the scenario is getting worse. The wonderful thing about modern technology is the amount of communication and information-sharing it facilitates. And the awful thing about modern technology is the amount of communication and information-sharing it facilitates. We are deluged with new information and connections, via telephones, webcams, instant messengers, e-mail, websites, blogs, newsletters, wikis, and social networking technology. The list gets longer every year. And with Blackberry and the mobile internet you can have data and demands coming at you 24/7. No wonder people are starting to run workshops on 'digital stress'.

All of which is bad enough whatever your line of work. But if you're a professional artist or creative, it's even more damaging. Concentration is essential for creative work - certain stages of the creative process require single-minded focus on the task in hand. When we're really in the zone, we experience 'creative flow' – the 'almost automatic, effortless, yet highly focused state of consciousness' that psychologist Mihaly Csikszentmihalyi has identified as characteristic of high-level creative performance. Interruptions, multi-tasking and the anxiety that comes from trying to juggle multiple commitments – these are in danger of eroding the focused concentration that is vital for your creativity.

If you're worried about the effect of all those interruptions, frustrations and distractions on your creative work, this e-book is for you. Over the next seven chapters I will offer you some principles and practical methods for maintaining your creative focus under pressure, and for managing the stream of information and demands so that it informs and stimulates your creativity instead of drowning it out.

And that means being organised.

There, I've said it. Organisation, structure, discipline and habit – these often seen as threats to creativity. Not to mention corporate-sounding phrases such as 'time management' or 'workflow'. We like to think of creativity as a space for untrammelled imagination, free from all constraints. Yet while freedom, rule-breaking and inspiration are undoubtedly essential to the creative process, the popular image of creativity overlooks another aspect: examine the life of any great artist and you will find evidence of hard work, discipline and a hard-won knowledge of the rules and conventions of their medium. Choreographer Twyla Tharp, who directed the opera and dance scenes for the film *Amadeus*, has this to say about the film's portrait of Mozart:

> The film *Amadeus* dramatizes and romanticizes the divine origins of creative genius. Antonio Salieri, representing the talented hack, is cursed to live in the time of Mozart, the

> gifted and undisciplined genius who writes as though touched by the hand of God… Of course this is hogwash. There are no 'natural' geniuses… No-one worked harder than Mozart. By the time he was twenty-eight years old, his hands were deformed because of all the hours he had spent practicing, performing, and gripping a quill pen to compose… As Mozart himself wrote to a friend, "People err who think my art comes easily to me. I assure you, dear friend, nobody has devoted so much time and thought to composition as I. There is not a famous master whose music I have not industriously studied through many times."

This passage is taken from Tharp's excellent book *The Creative Habit: Learn it and Use it for Life*, in which she argues that 'routine is as much a part of the creative process as the lightning bolt of inspiration, maybe more'. It's an inspiring, challenging and very practical book that deserves a space on the shelf of anyone who takes their creative work seriously.

I'm not suggesting that all artists and creatives need to be 'organised' in a way that would satisfy a corporate boss. You might get up at noon and work at home in your dressing gown, in a pigsty of a living room. You might check into a different hotel room every day and work on the bed. Your creative process and working habits might look like total chaos to an outsider, but if they work for you, that's all that matters. And there will be some method in the madness – patterns in your daily activities that are vital to your creativity. These are the things you need to do to keep your imagination alive – whether it's sitting at a desk by 6am, using the same pen, notebook or make of computer, hitch-hiking across America, putting rotten apples in your desk so that the scent wafts into your nostrils as you work, or sitting in your favourite café with a glass of absinthe.

In this e-book, I will offer some suggestions for keeping the tide of external demands at bay and helping you to develop a truly creative routine and rhythm to your working day. I won't offer you a rigid system or any 'best practice' nonsense – just some principles and suggestions for you to try out and adapt as you see fit. As well as drawing on my own experience and study of the creative process, I'll refer to some well-known time- management systems and suggest what I think they have to offer creative professionals.

The next seven chapters will look at specific elements of personal organisation and time management, while the final chapter will be a list of further resources.

Questions

What is your attitude to organising your creative work? Do you see organisation as soulless, uncreative routine or as a necessary, helpful part of your creative process?

What effect does feeling muddled and disorganised have on your creativity?

Which areas of your work would you like to be more organised about?

What do you like about chaos? Where in your work do you want to give chaos and randomness free rein?

2. Prioritise 'important but not urgent' work

A couple of years ago, I was facing a brick wall. I was in the second year of a part-time Master's degree that was essential for my business. I was invited to edit an issue of Magma, one of the top poetry magazines in the UK – as a poet, this was a chance I couldn't turn down. I was also getting married, which took a fair amount of preparation too – and that was one opportunity I was definitely not turning down! Meanwhile, I somehow had to keep my business going, keep my clients happy and fund all these extra-curricular activities.

As if that weren't enough, I discovered this new phenomenon called blogging – or rather, discovered that people were using it to spread their ideas and promote their businesses, rather than just to write about their cat's breakfast menu. It looked like a perfect medium for me – I loved writing, I had ideas I wanted to get into circulation, and I loved connecting with new people. But where was I going to find the time?

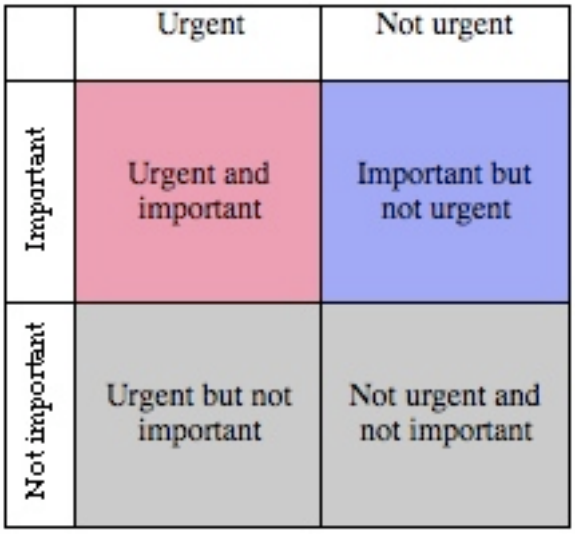
I'd already made a reluctant deal with myself to put my poetry-writing 'on-hold' until the end of the MA (on condition that I resumed afterwards, which I'm now doing with pleasure). But I was still faced with the seemingly impossible task of finding quality, focused time, away from interruptions, to write my essays, read poetry submissions with the care they deserved, and start a blog. After scanning my diary and surveying the tasks in hand, I was faced with a depressing conclusion.

I was going to have to get up early.

There was simply no other time in my schedule – or not the quiet, uninterrupted time I needed for my work, without the intrusion of phone calls, e-mails, meetings and classes. I had never considered myself one of nature’s early risers, and working from home much of the time had allowed me the luxury of avoiding early starts for commuting. On a good day I’d be up by 7.30, on a bad day it was closer to 8.30. Still time to get a reasonable amount of work done by starting at 9.00 – but I was faced with an unreasonable amount of work, so drastic action was called for.

My new start time became 6.30am. If you want to know how I managed this, read Steve Pavlina’s excellent post How to become an early riser. Here, I’m more concerned with the effect – since making the change, I’ve edited a postbag of 2,500 poems into Magma issue 34, achieved a distinction in my Master’s, and created the Wishful Thinking blog which has transformed my business and opened up many new creative avenues for me to explore. I’ve also written some poems I’m pleased with (at the moment, anyway) and which are gradually making it into publication. Most importantly of all, I made it to the wedding on time!

I’m not listing the above to blow my own trumpet, but to illustrate the value of ring-fencing time for your own creative work, in the midst of more urgent demands. It would have been easy for me to justify turning down the poetry magazine because I was too busy. It would have been even easier to put off starting the blog until I had more time. I could even have reasoned my way into stopping or deferring the Master’s. But the thing is, there will always be something ‘urgent’ taking my attention away from my own creative initiatives. Yet when I look back over the last couple of years, the time when I’ve created most value, for myself and my clients, has been those first hours of the day I’ve spent writing blog posts, essays, seminars and poems. It’s the creative wellspring that feeds into all the coaching, training, presenting and consulting I do when I’m face-to-face with clients.



Enough about me. How can **you** find time to achieve your creative ambitions?

Prioritise work that is ‘important but not urgent’

In his book The Seven Habits of Highly Effective People, Stephen Covey classifies work tasks according to whether they are important or urgent.

Covey points out that many of us spend too much time on tasks that are **urgent and important** (the red square in the diagram) – in other words, staving off emergencies by rushing around to solve problems or responding to others’ demands at short notice. Sometimes this is unavoidable – ‘deadline magic’ can spur us on to feats of creative production we wouldn’t otherwise attempt. This can be an exciting and productive experience – but it’s up to you whether you want to work like this most of the time. The example of the computer games industry – where extended ‘crunch’ times can mean endless overtime to meet a deadline – suggests that prolonged deadline magic can turn into deadline misery, with a significant impact on morale and efficiency.

Covey’s solution is to prioritise work that is **important but not urgent** (the blue square in the diagram). Though this is hard to do on any given day, it is the only way to ensure you are **making progress towards your own goals and dreams**, instead of merely reacting to what other people throw at you. And over time, the more you are dealing with important things before they become urgent, the fewer ‘urgent and important’ tasks you will have to deal with.

The most obvious way to do this is to work on your own projects first every day, even if it’s only for half an hour. Whatever interruptions come along later, you will at least have the satisfaction of having made some progress towards your own goals.

It’s obviously not just a question of time – you also need to **ring-fence your attention** so that you can devote your full attention to your creative work, without being knocked off course by distractions. The next chapter will look at how some highly creative people have achieved this, and what you can learn from them.

Questions

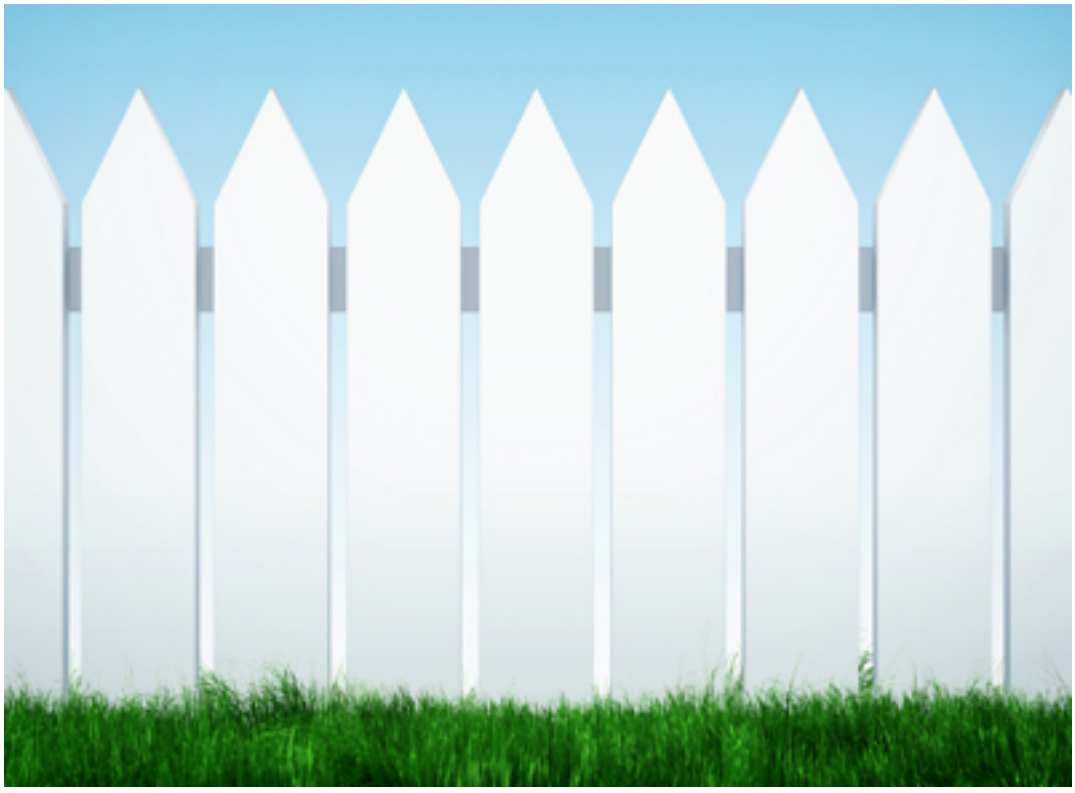
Think of the achievements you are most proud of, and that have added most value to your life and work. When you were working on them, how many of them fell into the ‘important but not urgent’ category?

How do you feel at the end of a day where you have made even a little progress towards a cherished goal?

How do you feel at the end of a day that has been totally swamped by others’ demands and urgent tasks?

What difference would it make to your life if you devoted more of your time to ‘important but not urgent’ work?

3. Ring-fence your most creative time



We've looked at the importance of prioritising 'urgent but not important' work. But how do you actually do this, and maintain the laser-like focus required for concentrated creative work, in the midst of all the demands and distractions of your working life?

Pick your most creative time of day

In my last chapter, I talked about my decision to get up early to write, before the onslaught of phone calls and other distractions. Apart from the lack of external interruptions, I write first thing in the morning because (once I'm up) that's the time of day when I'm most focused and alert. I experience a greater mental clarity in the first couple of hours of the working day than at any other time. As a writer, that quality of attention is my most valuable asset, so I've learned to guard it carefully. If I start ploughing into e-mails, reading blog feeds or doing mundane tasks such as accounts, then I'm squandering my most precious resource.

Victorian novelist Anthony Trollope puts me to shame with his habit of getting up at 5.30 to write his novels before breakfast. But early morning isn't the most creative time for everyone. We all have our own daily rhythms of alertness and rest. ProBlogger Darren Rowse says that 10am to 12pm are his "golden hours" for "thinking creatively and getting things done". Fellow blogger and

author of *The Four Hour Work Week* Tim Ferriss writes blog posts in two phases, at different times of day:

> Separate brainstorming (idea generation) from synthesis (putting it all into a flowing post). I generally note down 10-15 potential points for a post between 10-10:30am with a double espresso, select 4-5 I like and put them in a tentative order from 10:30-10:45am, then I'll let them marinate until 12am-4am, when I'll drink yerba mate tea, craft a few examples to match the points, then start composing. It's important to identify your ideal circadian schedule and pre-writing warm-up for consistent and reliable results.
> The Four Hour Work Week Blog

Writer Maya Angelou makes a similar distinction between time for writing the first draft and for revising it:

> I get up about five… I get in my car and drive off to a hotel room: I can't write in my house, I take a hotel room and ask them to take everything off the walls so there's me, the Bible, Roget's Thesaurus and some good, dry sherry and I'm at work by 6.30. I write on the bed lying down – one elbow is darker than the other, really black from leaning on it – and I write in longhand on yellow pads. Once into it, all disbelief is suspended, it's beautiful…
>
> After dinner I re-read what I've written… if April is the cruellest month, then eight o'clock at night is the cruellest hour because that's when I start to edit and all that pretty stuff I've written gets axed out.
>
> (From *Creators on Creating*, Ed. Frank Barron, Alfonso Montuori, Anthea Barron)

Ring-fence your attention – get yourself in the right state of mind

Maya Angelou's writing habits might seem eccentric, but they actually make complete sense. Writing in a hotel room effectively separates her writing time from the rest of her life, eliminating distractions and ensuring that she can enter a highly creative state of mind whenever she wants to.

My first professional training was in hypnotherapy, which taught me how sensitive the nervous system is to 'triggers' in our environment. The more intense the original emotion and the more unique the trigger, the stronger the emotional reaction will be. For example, if you think of a song you used to play with your first boyfriend/girlfriend, you will probably start to feel some of the emotions you felt with him/her without even hearing the recording. Supposing the song were Bowie's 'Moonage Daydream', you might get something of the same nostalgic/romantic feeling by listening to other glam rock songs, but not as strongly as whenever you hear the opening chords of 'Moonage Daydream' itself.

Looking at Angelou's account, she obviously experiences strong emotions whenever she is writing. And by confining her writing to a special place and time, she has trained her nervous system to associate her creative state with unique combination of different triggers – the early morning drive, the hotel room, blank walls, the Bible, *Roget's Thesaurus*, dry sherry, lying on the bed and yellow paper. No wonder she is in the 'zone' soon after entering the room!

Looked at from this perspective, many of the supposed eccentricities of creative people seem perfectly logical and reasonable. Here is Stephen Spender describing the working habits of himself and his fellow poets:

> Schiller liked to have a smell of rotten apples, concealed beneath his desk, under his nose when he was composing poetry. Walter de la Mare has told me that he must smoke when writing. Auden drinks endless cups of tea. Coffee is my own addiction, besides smoking a great deal, which I hardly ever do except when I am writing.
>
> (from *Creativity*, ed. P.E. Vernon)

Can you see how each of the poets is using a particular stimulus to trigger his creative state?

Maybe you have a special place you go to for focused creative work – a secluded office, a particular chair, a seat in your favourite café. Or you may have a favourite notebook, pen, software application or make of computer – using other tools doesn't feel quite right. Once you get into the habit of using these triggers, they form a kind of ritual, or process of self-hypnosis if you like, to help you reach that state of focused absorption Mihaly Csikszentmihalyi calls 'creative flow'.

I don't go as far as Maya Angelou, but I do have a morning ritual to help me get into the right frame of mind for writing. A cup of tea first, then filter coffee. Tea goes in the blue china 'Maneki Neko' mug. Coffee in a different cup, covered in Japanese calligraphy, from Kyoto (a city with wonderful associations for me). If I'm drafting poetry, the Mac is banished from the desk. I write on sheets of A4 with a black 0.5mm Muji pen. For other writing, I use the Mac and switch on Isolator to black out the whole screen except the window I am writing in.

Even if you have to work in the middle of an office, there are things you can do to minimise distractions and interruptions. Switch off your mobile phone. Put the landline on answerphone. Close your e-mail application. If the office noise is distracting, try listening to music on your headphones. Set up a signal (e.g. a 'Do Not Disturb' sign on your desk) to let your colleagues know they will interrupt you at their peril. Keep a notepad open and write down any tasks to do with other projects that occur to you while you are working. You can then consult the pad and get on with them after you have finished. (Writing them down will get them off your mind and leave you free to focus – more on this in Chapter 6.)

Finally, if you find yourself starting to procrastinate, here's an excellent tip from Mark Forster. Say to yourself: 'I'm not really going to start working on this piece, I'll just open up the file and look at it…'

Questions

When is your most creative time, when you are most alert and find it easy to focus?

If you could arrange your ideal schedule, what time would you ring-fence for focused creative work?

How close to your ideal schedule can you get within the constraints of your current situation?

Do you have a special place for creative work?

What physical triggers (such as pens, paper, computer hardware or software), rituals or routines do you use to get yourself in the right state of mind?

4. Avoid the 'Sisyphus effect' of endless to-do lists

Imprisoned in the ancient Greek underworld as punishment for his earthly crimes, Sisyphus was famously tortured by a never-ending task. He was condemned to roll a huge rock up a hill – only to watch it roll back down and have to start all over again.

Sound familiar?

Let's have another look at the scenario from the first chapter:

> So you start the day full of enthusiasm. You're excited about a new piece of creative work and itching to put your ideas into action. Firing up your computer, the familiar stream of e-mails pours into your inbox, burying the ones you didn't get round to replying to yesterday. Scanning through the list, your heart sinks – 2 of them look as though they require urgent action. You hit 'reply' and start typing a response to one of them… 20 minutes later you 'come round' and realise you've got sucked into the e-mail zone, and have been sidetracked by interesting links sent by friends, and getting involved in issues that aren't a priority for you. You minimize the e-mail window and get back to your project…
>
> After 15 minutes you're really enjoying yourself, getting into your creative flow – when the phone rings. Somebody wants something from you. Something to do with a meeting last week. You rummage through the papers on your desk, searching for your notes. You can't

find them. Suddenly your heart leaps as you lift up a folder and find an important letter you'd forgotten about – it needed an urgent response, several days ago. 'Hang on, I'll get back to you' you tell the person on the phone, 'I'll ring you back when I've found it'. You put the phone down and pick up the letter – this needs sorting immediately, but you remember why you put it off – it involves several phone calls and hunting through your files for documents you're not sure you even kept. By now, you've only got half an hour before your first meeting and you've promised to ring that person back. Your design stares at you reproachfully. The e-mail inbox is pinging away as it fills up – already there are more messages than before you started answering them. Your enthusiasm has nosedived and the day has hardly begun. Creative work seems like a distant dream.

There are two big problems with this way of working:

1. You are at the mercy of interruptions

Whenever you sit down to focus on your own work, you never know when your concentration will be broken – by an e-mail, a phone call, a request from a colleague, or even by yourself, when you suddenly remember something important that you've forgotten to do. Almost as bad is as the interruptions is anticipating them – you can never really relax and focus, because you know you could be derailed at any moment.

When I trained in hypnosis one of the things I learned was that if you want to create amnesia, you should keep interrupting people and/or change the subject. The hypnotic explanation is that memory is dependent on your state of mind, so when you change your focus, you're changing your state of mind – which makes it hard to remember what you were thinking about before. Think of a time when you were chatting to a friend in a restaurant or at a party and someone came over and interrupted you with a question – when they left, you both turned to each other and asked 'What were we talking about?'.

The bottom line is that **interruptions destroy your concentration**. And loss of concentration = loss of creative work. If you're not careful, you can end up in permanent 'reactive mode' – spending your time responding to others' demands and all the things you have to do instead of the one thing you really **wanted** to do today.

Let's face it, the interruptions aren't going away anytime soon. If they did, it would be a bad thing – it would mean you had no clients, colleagues, customers or collaborators. Obvious short-term fixes are to close your e-mail application, switch your phone onto answerphone mode and put a 'do not disturb' notice on your office door – as recommended in my previous chapter on how to **Ring-fence your most creative time**. But you can't do this all the time – and it doesn't solve your second big problem…

2. The 'Sisyphus effect' – a never-ending to-do list

The 'Sisyphus effect' is the result of endless to-do lists, which in turn are created by a constant stream of incoming demands. We start the day full of enthusiasm, but by the end of it, we've taken on so many new commitments that the to-do list is longer than when we started.

Productivity expert Mark Forster makes an excellent and (to me) surprising point about motivation:

> What is the best way to motivate yourself for your daily work? Obviously, enjoying your work and having a clear vision are very important, but I don't believe they are the most important things for keeping going during the daily grind. On the contrary, I believe that what gives us the most energy is the feeling of being totally on top of our work. If you are totally on top of something you have the energy to do it even if you don't particularly like the work.

I hadn't thought about it like that before, but on examining my own experience, I think he's right – when I'm faced with a limited number of things to do in a day, even if they aren't all particularly exciting, I generally feel motivated enough to get through them. But if I'm faced with a vaguely-defined, open-ended list of tasks, I can feel a sense of hopelessness and my energy drops. I can even find myself paralysed by inaction when faced with 3 or 4 really exciting pieces of work, if I don't think there's time to do them all.

You need to give yourself room to breathe!

Faced with the twin problems of unpredictable interruptions and the Sisyphus effect of never-ending tasks, you need to give yourself room to breathe, keep a clear head and stay focused on what you want to achieve. In short, **you need to install a buffer between others' demands and your response**. Otherwise you'll end up in permanently anxious and unproductive 'reaction mode'.

On the other hand, you need to find a way of fulfilling your commitments and giving others what they need from you within a reasonable timescale. Otherwise you'll quickly gain a reputation for unreliability and pay the penalty.

How can you manage this? I'll present one solution in the next chapter – a surprising and counter-intuitive idea that has transformed my own working life…

Questions

What effect do interruptions have on your creativity?

Do you recognise the Sisyphus effect? What does it do to your motivation levels?

What difference would it make to your working life if being derailed by others' demands was the exception rather than the rule?

What difference would this make to your creativity?

5. Get things done by putting them off till tomorrow

In the last chapter, I described the problems created by a never-ending stream of incoming demands: on the one hand, the constant interruptions can destroy the concentration required for creative work; on the other, endless to-do lists create the 'Sisyphus effect' – a feeling of hopelessness and demotivation.

In his excellent book *Do It Tomorrow*, Mark Forster provides a provocative and elegant solution to these problems, which transformed my working life. He suggests we create a buffer between incoming demands and our response – by making 'do it tomorrow' our default response to all requests. Not 'tomorrow' as in 'tomorrow never comes', but 'tomorrow' as in 'tomorrow'. Not today or the day after tomorrow, but tomorrow.

For example, here's Mark's solution to the never-ending stream of e-mail. In this system, on a typical day you only have to deal with one day's worth of e-mails – i.e. those that arrived yesterday:

1. Supposing you received 40 e-mails yesterday (once you've weeded out all the spam) – the first thing you do is move these 40 e-mails into a folder marked 'action'. These are the only e-mails you are going to deal with today.
2. Sit down and answer them all in one batch. Or at most, two or three concentrated bursts of effort.
3. Any e-mails that arrive in your inbox are collecting there for tomorrow – whatever you do, **don't** get caught up in responding to them, or you will find yourself back in Sisyphus' shoes, facing an endless task!

Of course there will be exceptions – sometimes you will receive an e-mail that has to be answered today – e.g. from your boss, demanding an urgent document by 5pm. But these should be the exceptions, rather than the general rule. Mark argues that most tasks are not nearly as urgent as we think they are – ask yourself 'Will there be a disaster if I don't answer this until tomorrow?' and the answer is usually 'no'.

'Doing it tomorrow' has several benefits:

- **Dealing with e-mails in one batch is more efficient**. You can get into 'e-mail mode' and zip through them in one go.
- **It's more motivating to deal with a finite number of e-mails than an ever-expanding inbox**. In other words, it cuts out the Sisyphus effect and presents you with a manageable task instead of a never-ending one.
- **Today's e-mails can't interrupt you** – because you're not going to respond to them today. I experience a feeling of relief each time I look at an e-mail containing a request and then 'let go' of it and return to the task in hand – confident that I will deal with it tomorrow.
- **You answer e-mails in a better state of mind** – so you're much less likely to take on unnecessary commitments by agreeing to something in order to get rid of the e-mail. You are also likely to make a more thoughtful and helpful response.
- **It doesn't really matter how often you check your e-mail**. Personally I can see the benefit of only checking e-mail once a day, but I'm not disciplined enough to resist, especially if I'm waiting for something important. This way, I can check my e-mail as often as I like without getting caught up in responding to it.
- **You deal with the difficult e-mails**. Most of us have a few 'tricky' e-mails that we put off answering for various reasons. But this system means you answer all the e-mails that came in yesterday – so you end up clearing out the difficult ones and getting them off your mind.
- **You know when you're finished for the day!** Once you've answered yesterday's e-mail, you're finished with e-mail today – how good will that feel?

The same principles apply to other communication channels: post, phone calls, text messages, commitments you take on at meetings. They all go into the in-tray for tomorrow. So at the start of every day, you know exactly how much you have to do to keep abreast of your commitments –

once you've dealt with a day's worth of e-mail, post, phone messages and verbal requests, you're free to get on with more interesting things. Like that design you've been itching to get back to.

N.B. This only applies to the reactive side of your work, i.e. requests coming in from others. Work initiated by you is a different matter – see the chapter on **Ring-fencing your most creative time**. Mark Forster suggests that you prioritise your own goals by devoting the first part of the day to a 'current initiative' of your own. But be wary of putting all your ideas for new initiatives into the in-tray for tomorrow, especially if you are the type of person who has a lot of ideas – I tried doing this when I first read Mark's book (not carefully enough) and put my back out by trying to do an absurd amount of work each day!

Yes but…

- **My in-tray already has hundreds of e-mails in it** – so did mine. Mark Forster suggests that you take all these e-mails and put them in a folder labelled 'Backlog'. Voila – an empty inbox! You can now implement the system by dealing with one day's worth of e-mail at a time. You should also set aside dedicated time to work through the backlog – because you have limited the size of the backlog, it can only get smaller, so every e-mail you deal with brings you closer to a cleared backlog.
- **People expect me to respond to them today**. Then manage their expectations. Sometimes it's a case of 'training' others to learn not to expect an instant response. On average, you're actually more likely to get back to them quicker using this system, since you're not overloading yourself by trying to answer everything as it comes in.
- **My boss expects me to respond today!** This is trickier. If you're lucky, your boss will listen to reason – you can explain your new system and s/he will be impressed by your efficiency and agree to wait until tomorrow unless it's really urgent. If not, then you can at least apply the system to everyone else you deal with.
- **I've got too much coming in!** Mark Forster is pretty blunt about this one – if you've got too many demands coming in on a regular basis then you need to scale down your commitments by saying 'no' and/or delegating more. It's not easy, but it's easier than carrying on with an impossible task.
- **I've got too many other things to do!** Beware of stuffing your diary so full of meetings and client appointments that you don't have time to do the rest of your work. And you don't need to keep up every single day. If I'm running a seminar all day I certainly won't be processing all my e-mails and post when I get home! They can definitely wait until tomorrow.

Mark covers all these objections (and more) in his book, *Do It Tomorrow* – he also offers many more invaluable suggestions, so if you're intrigued by the idea of 'doing it tomorrow', I highly recommend you get hold of a copy.

Do it tomorrow – or next week?

Mark Forster's 'do it tomorrow' system works for me, but it may not be right for you. Your work might follow different rhythms. 'Do it next week' might work better for you. Another productivity guru, Tim Ferriss says it's possible to manage by only checking e-mail once a week!!!

The key principle is to create a 'buffer' between the information and demands that are coming at you, and your response. That way you can get out of reactive mode, avoid the Sisyphus effect and spend more time on the kind of work that really inspires you.

Questions

What difference would it make to your work if you knew every morning how much work you had to get through that day?

Apart from 'do it tomorrow' how else could you create 'buffers' between incoming demands and your response?

6. Get Things Off Your Mind

So you're sitting at your desk, trying to focus on a piece of creative work – but it's hard to concentrate, there's something nagging at your attention. Suddenly it pops into your mind – you've forgotten something urgent! Or even worse, you get a phone call or an e-mail out of the blue demanding to know why you haven't delivered on a promise. Or you notice a post-it on the floor, which has fallen off your monitor, containing a reminder to DO SOMETHING VERY IMPORTANT… by yesterday.

If this happens to you often enough, you get used to living with a constant low-level anxiety – scanning your memory, your desk, your e-mails, you post-its, your scattered to-do lists – as you worry that you've forgotten something important. When you agree to do something, you may write it down – but can you be 100% sure you'll notice the note in time to do it? Or if you're out and about and make a commitment, how can you be sure you'll remember to put it on your to-do list when you get back to the office? Wherever you go, whatever you're doing, somewhere at the back of your mind you're wondering whether you've forgotten something vital that could blow up in your face at any moment.

How about this for an alternative?

> What if you could dedicate fully 100 percent of your attention to whatever was at hand, at your own choosing, with no distraction?

> It's a condition of working, doing, and being in which the mind is clear…
>
> Most people give either more or less attention to things than they deserve, simply because they don't operate with a "mind like water".

No, it's not an ancient Zen text – these words are lifted from David Allen's best-selling book on productivity, *Getting Things Done*. When I read this section of the book, I grasped the true value of having a system for managing your workload – not merely to be more productive but to **reclaim your own mind by clearing out unnecessary mental clutter** caused by trying to keep track of all your work commitments.

Before discovering the Getting Things Done system, I would typically have several to-do lists on the go at once, on different sheets of paper, not to mention the post-it notes stuck to my monitor. But I wasn't in the habit of writing **everything** down, so there were always several items I had to remember at any one time. I was vaguely aware that the effort to remember – and anxiety about forgetting – was taking up valuable mental energy and clouding my mind. I resented this all the more, because I had experienced the opposite. I had been on retreats where I had experienced a wonderful mental clarity and peace of mind after several days of silent meditation. But each time the retreat ended, I was frustrated when this clarity was eroded by the demands of everyday life.

When I read David Allen's book, I saw the possibility of experiencing the clarity of a 'mind like water' in the midst of my daily work. Apart from the obvious emotional benefits, I could see that it would help my creativity – the 'mind like water' state sounds very similar to creative flow as described by psychologist Mihaly Csikszentmihalyi: 'an almost automatic, effortless, yet highly focused state of consciousness'.

So how does David Allen suggest we can achieve this state of mind while dealing with the pressures of work?

Set up 'buckets' to capture your commitments

'Buckets' are physical or virtual containers where you capture important information, demands and commitments so that they can't 'leak' away and be forgotten. You should have as few of these as possible, but as many of them as you need.

Here are my buckets:

- **A plastic in-tray** for incoming letters, business cards, papers, meeting notes, scribbled to-do lists, etc.
- **My e-mail inbox**
- **The inbox on iGTD** – the software I use to manage my to-do lists
- **My two answerphones** (one mobile, one landline)
- **The 'Drafts' folder on my mobile phone** – I always carry my phone with me, so if I have an idea or make a commitment when I'm out and about, I write a text message to myself and save it in the drafts folder.

Important:

- **Put ALL your commitments into these buckets**. Even if I think I can remember a task easily, it will take up valuable mental space – if I put into one of these buckets, I will get it off my mind.
- **NEVER put a commitment anywhere but in your buckets**. If I don't put it in one of the above places, I have to assume it won't happen. So I've trained myself to do it. This was a bit odd at first, now it's almost automatic and I feel a slight sense of relief each time I get something off my mind and into a bucket.

Benefits

- **When you get things off your mind you can forget about them and give your full attention to whatever you're doing in the moment** – such as your creative work.
- **You'll stop forgetting important things** – the number of commitments I've forgotten has dropped dramatically since using this system.
- **You'll stop worrying about forgetting things** – see above.
- **You can easily review your commitments** – so you're less likely to take on more than you can manage.

So am I now living in a constant state of blissful peace and clarity? Not quite. If that's your goal then it's hard to beat the monastic routine. But I've definitely removed one big source of stress from my life – the effort of remembering important commitments and the danger of forgetting them. I've been using the 'buckets' system long enough to know that once I put a task in a bucket, I won't forget it. So once I've made a note, I can stop thinking about it and concentrate on whatever I'm doing right now.

Yes but…

- **I don't like the idea of having to write down all my commitments** – Neither did I. But once I tried the system, I found the benefits easily outweighed the effort. Now it's become a habit and I hardly notice it.
- **It's all very well capturing all this stuff in buckets, but how do I know I'll do anything about it?** That's what the next chapter is about…

Questions

What difference would it make to your life if you knew you would never forget another important commitment?

What would it be like if you could get your commitments off your mind and stay focused in the present?

What difference would it make to know that you could review all your current commitments by looking in 5 or 6 convenient ‘buckets’?

What buckets do you / could you use to capture your commitments?

7. Review your commitments

So you're ring-fencing your creativity, avoiding the Sisyphus effect of endless to-do lists and getting all your commitments off your mind and into buckets. So what happens next?

Obviously, there's no point capturing all those to-do items unless you're going to do something about them. Which means regularly 'emptying the buckets', reviewing your commitments and deciding what to do. How you do this and how often is up to you, but here are a few principles to bear in mind.

Why should you review?

1. First, and most obviously, to make sure you actually **do** the tasks on your to-do lists!
2. If you don't review the lists regularly, you'll soon stop trusting them and won't be able to use them to get things off your mind.
3. To think about how you're going to approach your work. It's tempting to 'get going' first thing in the morning, so you feel like you're getting things done – but whenever I do this, my day is always less productive and more stressful than on days where I take 10 minutes to review my commitments and decide how I'm going to tackle them.
4. It helps you step back and see the 'big picture' of your work, weigh up priorities and make decisions about your next steps.

5. Whenever you review your upcoming work and are confident you can get it all done, it will be a weight of your mind and your energy level will rise. If you review and find that you are **not** confident of getting it all done, then the review will be even more valuable – better to find out now than later on!

When should you review?

In his book *Getting Things Done*, David Allen suggests that you review your to-do lists as often as you need to in order to feel on top of things. I do a mini-review every morning when I look through my e-mails and other in-trays (from yesterday of course).

A larger-scale weekly review is one of the cornerstones of the Getting Things Done system. David Allen describes the weekly review as a time to:

- Gather and process all your 'stuff'
- Review your system
- Update your lists
- Get clean, clear, current and complete

I'll be honest and say I don't do the review every week. Some weeks simply feel too busy, other weeks I'm so caught up in what I'm doing that stopping to review seems like an unnecessary interruption. But whenever I do make time for it, I always feel better – the review gives me a clear sense of where I am and what I'm doing. I nearly always find something important that was in danger of slipping through the cracks. After finishing the review, I'm full of renewed enthusiasm for my work. So maybe I'll do it this week after all…

How should you review?

In his book, David Allen gives detailed instructions on performing a weekly review. But it's really up to you how you do it – the review is about doing whatever you need so that you feel on top of your work.

Here's what I usually do:

1. Empty all my 'buckets' (For a definition of buckets see the previous chapter, **Get things off your mind**.)
2. Review my diary.
3. Review my to-do lists, deleting anything I've done or am not going to do.
4. Decide on my priorities – which projects do I really want to move forward in the next week? How will I find time for them?
5. Backup my computer and blogs.

It's important to empty your buckets by **making sure you have a record of each task in a place where you will find it when you need to**. It's up to you how you manage your to-do lists – you might like to have one big list or several, on paper or in digital format. David Allen suggests you have different lists for different contexts – e.g. a list of phone calls to make by the phone, a separate list of things to do when you're in town etc. . .

I use iGTD to manage most of my lists – it's designed for the Getting Things Done system and allows me to assign tasks to both projects (e.g. 'Blog ideas') and contexts (e.g. phone calls or e-mails). When I empty my buckets, I transfer any tasks from meetings, answerphone messages, notes etc. to iGTD. There's no need to do this for e-mails, as the e-mails themselves serve as reminders of the tasks – I'm not finished until the inbox is empty.

Reading through that last paragraph, I realise how geeky I must sound! Well, I'll let my friends be the judge of that. The system probably sounds like a lot of work, but I hardly notice it any more. It took a while to get used to this way of working, but now it works so well for me, I take it for granted. Dealing with tasks in this way has almost become automatic, leaving my mind free to think about more interesting things.

On the subject of geekiness, it will probably come as no surprise to learn that Getting Things Done and similar systems can become an obsession with some enthusiasts. If you're not careful you can spend so long reviewing and tweaking your system that you never get round to actually doing the things on your list… Having said that, I've found the time I've invested in investigating these systems and changing my working habits has been repaid many times over. I hope this e-book helps you make your working life more productive, enjoyable and creative.

In my final chapter I'll point to some useful books, software and websites to help you fine-tune your own personal organisation system.

Questions

How often do you review your commitments? Daily? Weekly? Never?

What difference does it make when you make time to review?

What difference does it make when you're 'too busy' to review?

How do you review? Any tips you'd like to share?

8. Resources to help you get things done

In this e-book I've given you my take on time management and how it can help or hinder creative work. In doing so, I've taken elements from different systems, having assimilated them over time and adapted them to my own needs. If you are keen to investigate these systems, please make sure you try them one at a time! Otherwise you'll end up confused. It's worth devoting some time to working with a system until you know it really well, before deciding whether you need to add to it. The following are all resources I've used myself.

My 'GTD' delicious bookmark

My GTD del.icio.us bookmark is where I bookmark any web pages I find with useful material about time and workflow management. ('GTD' stands for 'Getting Things Done'.) If you subscribe to the RSS feed for this bookmark then you'll receive new recommendations as I find them.

Books about the creative process

The Creative Habit – Learn it and Use it for Life, by Twyla Tharp
Very down-to-earth, very practical, very inspiring. The famous choreographer shares her working

routine and argues passionately that inspiration doesn't come without a lot of perspiration, discipline and hard work. Highly recommended for anyone who takes their creativity seriously.

Creativity – Flow and the Psychology of Discovery and Invention, by Mihaly Csikszentmihalyi
Classic study of the creative process, based on the idea that peak creative performance is characterised by the state of creative flow, in which distractions are tuned out and there is a single-minded focus on the work. Achieving more creative flow is one of the main reasons for a creative person to learn about time management.

Creators on Creating, edited by Frank Barron, Alfonso Montuori, Anthea Barron
A rich collection of first-hand accounts of the creative process, including some fascinating descriptions of creators' working habits. Contributors include Leonardo da Vinci, Brian Eno, Ingmar Bergman, Isadora Duncan, Richard Feynman, Rainer Maria Rilke and Frank Zappa.

'Do it tomorrow' – Mark Forster's approach

Do It Tomorrow and Other Secrets of Time Management, book by Mark Forster
If you're pushed for time and want a book on time management that will deliver results fast, this is the one I recommend. The book is deliberately provocative, prompting you to reconsider working habits you take for granted and full of non-obvious suggestions that make complete sense once Mark has explained his reasoning. Not only that, the ideas work and can be applied almost immediately.

Mark's system is not as complex as David Allen's (see below) but that doesn't mean it's less powerful, it's just different. It may well be all you need.

Mark also writes a lively and useful blog, Get Everything Done and provides a selection of Time Management Articles.

'Getting Things Done' – David Allen's system

Getting Things Done - How to Achieve Stress-Free Productivity, book by David Allen
David Allen's 'Getting Things Done' is a powerful system with legions of devoted fans. It's not for the faint-hearted however - once you've read the book it takes 2-3 full days of work to capture all of your commitments and set up the system. Having done this, I would definitely say it's been worth the time and effort, even if it means sacrificing a weekend.

David Allen's website - lots of resources, including free articles, a blog and forums.

Productive Talk Podcast - Merlin Mann Interviews David Allen about his system.

Getting Things Done Guru David Allen and His Cult of Hyperefficiency - Wired Magazine feature on the origins of Getting Things Done.

Getting Design Done - excellent piece about the Getting Things Done system and Creativity, from D. Keith Robinson at Graphic Define.

Implementing GTD for Creative Work? Merlin Mann of 43 Folders responds to Keith Robinson's article and asks whether GTD and creativity are compatible - prompting an interesting discussion in the comments.

Software

If you're a Mac user, Isolator will help you concentrate on your work by blacking out everything on your screen except the menu bar and the application you are currently working in. I gather Dropcloth does the same for Windows.

I use iGTD to organise my projects and 'to do' lists. It's flexible and reasonably user-friendly.

When I had a PC I used Thinking Rock which has similar functionality.

Blogs

43 Folders is a hugely popular GTD blog, run by Merlin Mann. A vast archive of tips and techniques for improving your productivity. I'm tempted to say I don't know how he finds the time, but I'll resist.

Lifehacker is a constant stream of potentially useful productivity tips, many of them tech-oriented, e.g. Manage a to-do list with your iPod touch. There are lots of posts published every day, but it doesn't take long to read - I just skim through the headlines until I find something useful.

Lifehack.org is another deservedly busy and popular productivity blog, run by Leon Ho and his team.

The Four Hour Work Week - As the title of his book *The Four Hour Work Week* suggests, Tim Ferriss offers a radical and provocative approach to rethinking your work and life. It goes far beyond time management and is a stimulating read for anyone who feels overloaded with work and wants to spend their time doing more interesting things. I first heard about Tim's ideas through this interview with Darren Rowse of Problogger.

Stationery

The Behance team specialise in helping creative professionals 'make ideas happen'. They espouse a philosophy of 'productive creativity', embodied in their Action Method and in their range of products for capturing and processing creative ideas. I blogged about their Action Pad for creative meetings a while ago.

There's an argument that having good quality tools will encourage you to use them more. Since buying some Moleskine notebooks earlier this year I've definitely noticed my fingers itching to scribble down more creative ideas and lines of verse.

9. If you found this e-book helpful...

Share it with a friend or colleague

If you know someone else who might find this e-book helpful, feel free to send it to them. Under the terms of the Creative Commons licence, you are free to copy and share this document, as long as you do not sell it, and as long as you keep it intact and credit me as the author.

Subscribe to the Wishful Thinking blog

I write the Wishful Thinking blog to share tips and inspiration for creative professionals, based on my experience of coaching artists and creatives since 1996.

You can have the latest blog posts delivered to you (free, of course) via RSS or E-mail. (If you're new to RSS, have a look at my What is RSS? page.)

Join the discussion on the original blog series

The chapters of this e-book were originally published on Business of Design Online, which offers a wealth of resources from specialist authors, about the business side of working in the design industry. Click on the headings read the posts on BoDo and join the discussion in the comments.

1. Why you need to be organised to be creative
2. Prioritise work that is 'important but not urgent'
3. Ring-fence your most creative time
4. Avoid the 'Sisyphus effect' of endless to-do lists
5. Get things done by putting them off till tomorrow
6. Get things off your mind
7. Review your commitments
8. Resources to help you get things done

Subscribe to Business of Design Online

Subscribe to Business of Design Online to receive more information, tips and advice on the business of graphic design, via RSS or E-mail.

10. The author and publisher

Mark McGuinness - Author

Mark is a poet and a business coach who runs Wishful Thinking - a specialist consultancy for creative professionals and agencies, studios and other companies in the creative industries sector. He writes the Wishful Thinking blog to provide practical tips and inspiration for creative professionals of all descriptions.

Mark originally qualified as a psychotherapist and has an MA in Creative & Media Enterprises from the University of Warwick, UK. His poetry blog is Mark McGuinness|poetry. E-mail Mark.

Business of Design Online - Original Publisher

The text in this e-book was originally published as a series on Business of Design Online, which provides a wealth of information, tools and techniques for successfully managing and marketing a design practice.

BoDo was conceived and created by three seasoned professionals, hailing from three continents – Catherine Morley, based in Bangkok, Thailand, Jeanette (Jay) Wickham, based in Melbourne, Australia and Neil Tortorella, based in North Canton, Ohio, USA. Along with their various additional skills, each is a communication designer who've run their own businesses for many years.

On BoDo you'll find continuously updated resources for running a design shop, including select e-books, business forms, excellent articles and more. The team contributes regular blog posts along with guest authors who will share their business experiences.

Morley and Tortorella were also instrumental in the development of Creative Latitude (CL), a popular resource site for designers, writers and other creatives. BoDo and CL will be sister sites, complimenting each other in terms of audience and content. While BoDo is targeted to those who are just setting up their business, CL is targeted toward those with a a few years experience.

STUDY PROGRAMME ACCREDITATION MATERIAL:

INDUSTRIAL ENGINEERING / ENGINEERING MANAGEMENT

DOCTORAL ACADEMIC STUDIES

Novi Sad

2012.

Prevod sa srpskog jezika:

Jelisaveta Šafranj

Ivana Mirović

Marina Katić

Vesna Bodganović

Dragana Gak

Ličen Branislava

Content

Content

Content

Content

Content

Content

Programme name	Industrial Engineering / Engineering Management
Independent higher education institution where the programme is being executed	University of Novi Sad
Higher education institution where the programme is being executed	Faculty of Technical Sciences
Educational-scientific/educational-art field	Technical-Technological Science
Scientific, proffesional or art field	Industrial Engineering and Management
Type of studies	Doctoral Academic Studies
Study scope, expressed in ECTS	180
Academic degree, abbreviation	Doctor of Science - Industrial Engineering and Engineering Management, Ph.D.Ind.Eng.Managem.
Study length	3
Programme implementation starting year	2005
Future course implementation starting year (for new programme)	
Number of students attending this programme	37
Planned number of students to be enrolled in this programme	120
Programme approval date (state the approval issuer)	14.11.2012 - Science Education Council 29.11.2012 - University of Novi Sad Senate
Programme language	Serbian, English
Programme accreditation year	2008
Web address containing programme information	http://www.ftn.uns.ac.rs

Standard 00. Higher Education Institution Competence for the Implementation of PhD Studies

The Faculty is fully prepared in terms of academic staff, classroom capacity and other facilities for administering doctoral studies in all the fields studied at the Faculty based on indicators related to scientific and research work. The Faculty has a short-term and long-term plan and is accredited as a scientific and research institution, as required by law.

The ability of the Faculty to administer doctoral studies can be indicated by the following criteria:

•the number of Ph.D. and Master theses defended at the higher education institution which are in the area for which the study programme is accredited, in terms of the ratio of the doctoral and master theses and the number of students who have graduated from the programme and the number of professors.

•the ratio between the number of professors and the number of professors involved in scientific and research projects.

•the ratio between publications in the Ministry of Science acclaimed international journals in the last 10 years and the number of professors.

•cooperation with institutions in the country and abroad.

The capability of the Faculty to administer doctoral studies is obvious from the references which are enclosed with the accreditation material.

Standard 01. Programme Structure

The name of the Doctoral Study Programme is Industrial Engineering and Engineering Management. The acquired academic degree is a Doctor of Philosophy in Industrial Engineering and Management. The outcome of the learning process is the knowledge that enables students to become capable of independent scientific research.
Doctoral studies in Industrial Engineering and Engineering Management last three years and they are worth at least 180 ECTS. It is 90 ECTS obtained through examination of the subjects, 30 ECTS laying theoretical basis for doctoral dissertations, and 60 ECTS are acquired by the drafting and defense of the doctoral dissertation.
Doctoral studies last at least three years (six semesters) and no longer than ten academic years.
Research study on theoretical grounds is a doctoral dissertation qualifying exam for the preparation of a doctoral thesis in which students demonstrate that they mastered necessary theoretical knowledge in the scientific areas of interest. Theoretical foundations are laid as examination (written and / or oral) in certain fields of study (issues) from at least three courses defined in the study programme.
Studies on doctoral studies are organized through lectures, research work, scientific research, development and defense of the doctoral dissertation.
Student`s research interest is profiled by selecting the course which will be studied and taken; and thus, contribute in-depth knowledge and understanding of areas (themes) of his doctoral dissertation. Optional courses are selected from the group of proposed subjects of study programme, but the students have the opportunity to choose a number of courses, with the consent of the mentor, from a set of subjects for Doctoral Studies at Faculty of Technical Sciences, University of Novi Sad, or any other university in the country or abroad. At the same time the conditions prescribed for lecture attendance in selected cases have to be fulfilled.
Teaching activity for the courses (compulsory or optional) is a group or individual (mentoring) activity. Group classes are held when the course was chosen by five or more students or when this type of training is necessary to organize due to the nature (character) of the subject-matter. The decision on the type of instruction and optional courses that are taught is taken by the Head of Doctoral Studies with the consent of the Manager of Doctoral Studies at the Faculty.

Standard 02. Programme Objectives

The purpose of this Study Programme is education of students capable of high quality and independent scientific research in accordance with the needs of our society. On the other hand, educating staff who are trained to critically evaluate research work and independently carry out original and scientifically relevant research enables the development of new technologies and procedures that contribute to the overall development of society. In addition, the purpose of this Doctoral Study Programme is a contribution to our national science as well as the application of new scientific solutions in the industry and in broader areas of energy, telecommunications, electronics and computing.
Study Programme at Doctoral Studies in Industrial Engineering/Engineering Management is designed to provide acquisition of skills that are socially justified and useful. The Faculty of Technical Sciences defined tasks and goals for educating highly competent personnel in the field of technology. The purpose of this Study Programme is completely in line with high objectives and goals of the Faculty of Technical Sciences and at the level of strict standards of education in the Ph.D. in Industrial Engineering/Engineering Management.

Standard 03. Programme Goals

The objective of the study program is to achieve student`s scientific competencies and academic skills in the field of Industrial Engineering/ Engineering Management. Besides, this includes the development of creative abilities of considering the problems and the ability of critical thinking, development of teamwork skills and mastering specific practical skills necessary to perform the profession.
The objective of the study program is to educate an expert who has sufficient extended knowledge consistent with contemporary directions of development of science in the world.
One of the specific objectives which is in accordance with educational aims of experts at the Faculty of Technical Sciences is to develop students` awareness of the need for a personal contribution to the development of society in general and environmental protection. The objective of the study program is also the education of experts in the field of teamwork, and development of technical capacity for communication and presentation of their original results to scientific public.

Standard 04. Graduates' Competencies

PhD graduates of the academic study programme in Industrial Engineering/Engineering Management are competent to conduct research and solve problems in real life practice activities. Competencies include, above all, the development of critical thinking skills, the problem analysis capabilities, the synthesis solution, predicting the behaviour of selected solutions with a clear representation of what the advantages and disadvantages of the selected solution.
Qualifications that indicate the completion of doctoral academic studies are gained by students:
•who have demonstrated systematic knowledge and understanding in the field of Industrial Engineering/Engineering Management which is the basis for developing critical thinking and application of knowledge;
•who have mastered the skills and methods of research in the field of Industrial Engineering/Engineering Management;
•who have shown the ability of making concepts, design, construction and application of the selected solution;
•who have shown the ability to adapt the research process with the necessary level of academic integrity;
•who have performed original research work, extending the existing boundaries of knowledge, which is verified by publishing papers in the appropriate scientific journal and by the references at national and international levels;
•who are capable of critical analysis, evaluation and synthesis of new and complex ideas;
•who are capable of knowledge and ideas transfer to their colleagues, wider academic community and society in general
•who are capable of promoting technological, social and cultural progress in the academic and professional environment
After graduation, PhD programme allows students to have the knowledge, skills, developed abilities and competencies to :
•independently solve practical and theoretical problems and organize and realize developing activities and research;
•be involved in international scientific projects
•be able to implement the development of new technologies and procedures in the field of electrical and computer engineering and to understand and use modern knowledge;
•think critically, work creatively and independently;
•respect the code of ethics and principles of good scientific practice;
•be capable to present scientific research results at scientific conferences and publish in scientific journals, verifying them through patents and new technical solutions;
•contribute to the development of scientific disciplines in science generally.
After this study programme completion, the student obtains the following subject-specific competences:
•thorough knowledge and understanding of the disciplines that are the subject of their involvement;
•ability to solve problems using scientific methods and procedures;
•linking basic knowledge in various fields and their application;
•ability of follow modern developments in the field of profession;
•necessary skills and ability in applying knowledge in the field of industrial engineering/engineering management;
•the use of information and communication technologies
Students are trained to design, organize and manage production. During training the student acquires the ability to independently perform experiments, statistical analysis of results and to formulate and adopt the appropriate conclusions.
Students who complete their doctoral studies in Industrial Engineering / Engineering Management gain knowledge on how to economically use the natural resources of the Republic of Serbia in accordance with the principles of sustainable development.
Particular attention has been focused on developing the capacity for teamwork and the development of professional ethics.

Standard 05. Curriculum

The curriculum of the Doctoral Academic Study Programme Industrial Engineering/Engineering Management is made to meet the set goals. The structure of the study programme enables the students to choose optional courses which will be worth at least 70%.
During the course of the doctoral academic studies students are encouraged to specialize in the specific field of study they are most interested in. Through optional courses they are able to take further interest in the scientific and research areas studied during the course of their graduate academic studies.
All courses last one semester and are worth a certain number of ECTS credits.
The curriculum defines every course of the study programme which states the following: the course name, type, the year and semester when the course is lectured, the number of ECTS credits, the name of the lecturer, the course objective with the expected outcome, the knowledge and competences the student will acquire, the prerequisites for taking the course, the course content, the recommended literature, the methods of lecturing, the knowledge tests and evaluation and other relevant data. Each course is designed in such a way to provide about half of the class load in the form of lectures and half of class load in the form of research. The research is an independent work of the PhD student, who has been doing the detailed study in the field of the selected course, as agreed with the course teacher.
The study programme is consistent with European standards regarding enrolment requirements, duration of study, terms of enrolling into the next year of studies, the acquisition of a diploma and mode of study.
The curriculum enables students to attend 7 courses during the first three semesters. In the first semester three compulsory courses are taught, namely: The research method and Selected topics in industrial engineering/engineering management and one optional course. In the second and third semester (each contains two optional courses), students elect optional courses after consulting their mentor, being available to every student of doctoral studies.
Lectures in teaching subjects are performed as a group or individual (mentor) classes. Group classes are held when there are five or more students studying particular subject, or when this kind of teaching is necessary to organize because of the nature (character) of the teaching subject.
The Manager of doctoral studies with the consent of the Chief of doctoral studies at the Faculty takes the decision on the type of instruction and elective courses.

Table 5.2 Course specification

Course:	Scientific Research Method
Course id: DZ001	
Number of ECTS: 5	
Teachers:	Atanacković M. Teodor, Folić J. Radomir
Course status:	Mandatory

Number of active teaching classes (weekly)

Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
0	0	0	3	0

Precondition courses: None

1. Educational goal:

To enable students for successful writing of scientific papers and doctoral dissertations.

2. Educational outcomes (acquired knowledge):

- Ability of understanding varius scientific metods witch was used in scientific literature
- Ability of successful managing in proffesonal literature
- Ability of successful writing of scientific paper in area of of interests
- Ability of successful creating and ending of doctoral dissertation

3. Course content/structure:

Definition of science. Development of science through history.
Scientific methodology.
General and special scientific methods.
Structure of a scientific paper. Types of scientific results.
Writing and publishing scientific papers.
Writing the doctoral dissertation.
Evaluating scientific results.

4. Teaching methods:

Lectures. Consultations with students. Seminar paper.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	30.00	Oral part of the exam	Yes	70.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Karl Poper	Logika naučnog otkrića	Nolit, Beograd	1973

Table 5.2 Course specification

Course:		Science of Industrial Engineering and Management
Course id:	IMDR0	
Number of ECTS:	11	
Teachers:		Beker A. Ivan, Borocki V. Jelena, Borovac A. Branislav, Ćosić I. Đorđe, Ćosić P. Ilija, Dobromirov P. Dušan, Grubić-Nešić S. Leposava, Katalinić -. Branko, Krsmanović B. Cvijan, Lazarević M. Milovan, Maksimović M. Rado, Nikolić T. Slavka, Ostojić M. Gordana, Radaković J. Nikola, Stankovski V. Stevan, Šešlija D. Dragan, Tešić M. Zdravko
Course status:		Mandatory

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The educational goal is to introduce students to doctoral studies in Industrial Engineering and Management, to learn about the history and development of the field and learn the general settings that apply in this area.

2. Educational outcomes (acquired knowledge):

The student is competent to apply global settings of this area in further education in vocational subjects.

3. Course content/structure:

A review of research in the fields of organization and business management, innovation and entrepreneurship, project management, investment management, information management, quality management and logistics, risk management and insurance management, industrial engineering, marketing and media, human resource management, energy management, planning, organization and systems management, automation, and information-management systems and kominikacionih quality and logistics.

4. Teaching methods:

Mentor together with the student selects one or more modules depending on the scope of the module. Consultation. Lectures are delivered in combination. Theoretical part is followed by the examples that clarify the theoretical part of the curriculum. In addition to lectures, consultations are held regularly. Student independently deepens the subject-matter learnt at lectures through his research work while studying scientific journals and other literature. In addition to working with the teacher, students are trained to write their own scientific papers.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Stankovski, S., Lazarević, M., Ostojić, G., Ćosić, I., Purić, R.	RFID Technology in Product/Part Tracking During the Whole Life Cycle	Assembly Automation, Elsavier	2009
2,	Maksimović, R., Lalic, B.	Flexibility and Complexity of Effective Enterprises	Strojniški vestnik - Journal of Mechanical Engineering	2008
3,	Gajić G., Stankovski S., Ostojić G., Tešić Z., Miladinović Lj.	Method of evaluating the impact of ERP implementation critical success factors–a case study in oil and gas industries	Enterprise Information Systems	2012
4,	Maksimović R., Stankovski S., Ostojić G., Petrović S., Ratković Ž.	Complexity and Flexibility of Production Structures	Journal of Scientific and Industrial Research	2010
5,	Blagojevic, V., Šešlija, D., Stojiljkovic, M.	Cost effectiveness of restoring energy in execution part of pneumatic system	Journal of Scientific and Industrial Research (JSIR),	2011
6,	Ćosić, I., Šešlija, D., Ignjatović, I,	Razvoj obrazovanja industrijskih inženjera	Ekonomski institut	2011
7,	Ignjatović, I., Komenda, T., Šešlija, D., Mališa, V.	Optimisation of compressed air and electricity consumption in a complex robotic cell	Robotics and Computer-integrated Manufacturing	2012
8,	Grubić-Nešić L., Duđak Lj	Ljudski resursi i razvoj industrijskog inženjerstva,	Ekonomski institut	2011

Table 5.2 Course specification

Course:		Selected Chapters in Mathematics
Course id:	DZ01M	
Number of ECTS:	12	
Teachers:		Adžić Z. Nevenka, Doroslovački D. Rade, Gilezan K. Silvia, Grbić P. Tatjana, Kostić Z. Marko, Kovačević M. Ilija, Mihailović P. Biljana, Pantović B. Jovanka, Pilipović R. Stevan, Rajković R. Milan, Ralević M. Nebojša, Sladoje Matić I. Nataša, Stojaković M. Mila, Teofanov Đ. Ljiljana, Uzelac S. Zorica
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	3	0

Precondition courses: None

1. Educational goal:

To acquire knowledge which can be used in professional subjects and practical work, develop and solve mathematical models for engineering courses using the knowledge gained through selected chapters in mathematics.

2. Educational outcomes (acquired knowledge):

Student will have been competent enough to develop and solve mathematical models in further professional education.

3. Course content/structure:

Student can choose in consultation with programme supervisor, one of the suggested modules: 1. Numerical Mathematics, 2. Optimization. 3. Pattern Recognition. 4. Partial Differential Equations, 5. Nonlinear Equations. 6. Computational geometry. 7. Elements of Functional Analysis. 8. Combinatorics. 9. Graph Theory.10.Operational Research- Linear Programming. 11. Probability 12. Statistics .13.Stochastic Processes. 14. Vector analysis. 15. Complex Analysis. 16. Linear Algebra. 17. Differential and Difference Equations. 18. Euclidean and Non-Euclidean Geometry. 19. Fractional Calculus,Differential Equations . 20. Operational Research-Quiuing theory. 21. Logic in Computing. 22. Discrete Mathematics. 23. Higher order Logic. 24. Theory of Mobile Processes. 25. Numerical Methods of Linear Algebra. 26. Fuzzy Sets. 27. Economic and Financial Mathematics. 28. Groups and Algebras Li. 29. Formal Languages and Automata Theory. 30. Process Algebras. 31. History of Mathematics. Part of the course is in the form of independent research and study in the field of mathematics. Study and research work is based on primary scientific sources, organization and conduction of experiments and statistical data analysis, numerical simulations, and possible paper in the field of mathematics.

4. Teaching methods:

Lectures. (The student can choose in consultation with supervisor, one or more modules depending on module scope). Consultations. Lectures are organized in combined form. The presentation of the theoretical part is followed by the corresponding examples which contribute to better understanding of the theoretical part. In addition to lectures there are regular consultations. Through research and study work the student will, on the bases of scientific journals and other relevant literature that has been studied independently, develop further understanding of the material covered in lectures. Working with the course teacher the student develops the ability to independently work on a scientific paper.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Alexander Mood,...	Introduction to the theory of statistics	McGraw Hill	2005
2,	Athanasios Papoulis	Probability, random variables and stochastic processes	McGraw Hill	2002
3,	I. Kovačević, N. Ralević	Funkcionalna analiza	FTN (edicija tehničke nauke-udžbenici), Novi Sad	2004
4,	N.Ralević,I.Kovačević	Zbirka rešenih zadataka iz Funkcionalne analize	FTN (edicija tehničke nauke-udžbenici), Novi Sad	2004
5,	M.Stojaković	Slučajni procesi	FTN, Novi Sad	1999
6,	V.Jevremović,J.Mališić	Statističke metode u metorologiji i inženjerstvu	Savezni hidrometeorološki zavod, Beograd	2002
7,	Zeidler E.	Nonlinear Functional Analysis and Aplications	Springer-Verlag, New York-Berlin-Heidelberg-Tokyo	1985
8,	Zlobec S., Petrić J	Nelinearno programiranje	Naučna knjiga, Beograd	1989
9,	Dauxois, M. Peyrard	Physics of Solitons	Cambridge University Press, Cambridge, New York	2006
10,	Saaty, T. L	Modern Nonlinear Equations	Dover Publications, Inc., New York	1981
11,	N. Ralević, S.Medić	Matematika 1 - drugi deo	FTN, Novi Sad	2002
12,	Heinz-Otto Peitgen, H. Juergens, D. Saupe	Chaos and Fractals	Springer Verlag, New York	2004

Literature

Ord.	Author	Title	Publisher	Year
13,	Mileva Prvanović	Osnovi geometrije	Građevinska knjiga, Beograd	1990

Table 5.2 Course specification

Course:		Selected Chapters in Physics			
Course id:	DZ01F				
Number of ECTS:	12				
Teachers:		Budinski-Petković M. Ljuba, Kozmidis-Luburić F. Uranija, Kozmidis-Petrović F. Ana, Satarić V. Miljko, Vučinić-Vasić T. Milica			
Course status:		Elective			
Number of active teaching classes (weekly)					
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:	
5	0	0	3	0	
Precondition courses		None			

1. Educational goal:

To acquire the knowledge of physics which is applied in modern engineering.

2. Educational outcomes (acquired knowledge):

The students will have acquired the knowledge which enables them to develop models for solving problems in practical professional work as well as evolvement in science and research work in the corresponding areas.

3. Course content/structure:

Student can choose in consultation with programme supervisor, one of the suggested modules: 1. Lasers, their applications in engineering, 2. Quantum tunnelling effect and applications, 3. Quantum dots, wires and tubes, Applications in nanotechnologies, 4. New materials, amorphous materials, spin glass, 5. Natural and artificial polymers and their application in nanotechnologies, 6. Numerical method of statistics physics, random number generator. Monte Carlo simulation.

4. Teaching methods:

Lectures. (The student can choose in consultation with co-mentor, one or more modules depending on module scope). Consultations. Lectures are organized in combined form. The presentation of the theoretical part is followed by the corresponding examples. In addition to lectures there are regular consultations. Through research and study work the student will, on the bases of scientific journals and other relevant literature that has been studied independently, develop further understanding of the material covered in lectures. Working with the course teacher the student develops the ability to independently work on a scientific paper.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature				
Ord.	Author	Title	Publisher	Year
1,	K. Binder, D.W. Heermann	Monte Carlo Simulation in Statistical Physics	Springer-Verlag	1988

Table 5.2 Course specification

Course:		Current State in the Field
Course id:	SID04	
Number of ECTS:	2	
Teachers:		Atanacković M. Teodor, Katić A. Vladimir, Kulić J. Filip, Vilotić Ž. Dragiša
Course status:		Mandatory

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
0	0	0	2	0

Precondition courses: None

1. Educational goal:

Introducing students to the current research directions and manners in solving problems from the wider study field.

2. Educational outcomes (acquired knowledge):

Knowledge on the current research directions worldwide in the field, based on lectures by prominent professors from the universities in Europe or prominent experts from the well-known companies abroad.

3. Course content/structure:

Contemporary topics in the field of research, presented by prominent professors and experts on lectures on invitation. Students select topics or attend lectures as they wish or as they find the topic interesting.

4. Teaching methods:

Survey on solving contemporary problems by theoretical methods and multimedia presentations.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	30.00	Oral part of the exam	Yes	70.00

Literature				
Ord.	Author	Title	Publisher	Year
1,	Razni	Časopisi sa SCI liste	IEEE Publishing, i dr.	2008

Table 5.2 Course specification

Course:		Selected Chapters in Non-Industrial Robotics			
Course id:	HDOK-2				
Number of ECTS:	14				
Teacher:		Borovac A. Branislav			
Course status:		Elective			
Number of active teaching classes (weekly)					
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:	
5	0	0	4	0	
Precondition courses		None			

1. Educational goal:

The course goal is to make students, having in mind their previous knowledge and interests, familiar with the new topics in the field of Non-Industrial Robotics, which is a field that is becoming increasingly more important, and to introduce them to research study.

2. Educational outcomes (acquired knowledge):

The expected educational outcomes of this course are the student`s knowledge and ability to fully understand the topics and issues related to Non-Industrial Robotics and his/her involvement in research work in this field of study.

3. Course content/structure:

In accordance with the student`s interests, some of the following topics will be further studied: applications for service robots (in a household, on a building site, in a hazardous environment, inspection robots, life saving robots, etc.), autonomous robots, control and regulation in biological systems, the comparison of the `control architecture` of biological systems and autonomous robots, types of autonomous robots depending on the way in which they move (wheels and tracks, jumping robots, snake-like robots, flying robots, multiple-legged and two-legged robot locomotion, etc.), robot learning, “behaviour-based robotics” which represents a new way in which we control robots in an unstructured environment like ours, grasping and manipulation of objects, humanoid robots. A part of the course work is conducted through independent individual study and research work in the field of Non-Industrial Robotics.The research study requires the student`s active and constant interest in and reading of the primary scientific resources, the organization and conducting of experiments and statistical processing of data, numerical simulations, writing a paper in the specific scientific field relevant to the doctoral dissertation

4. Teaching methods:

Depending on the number of students the course can be carried out either through lectures, or by working with a mentor (tutorial work). Modes of teaching depend on the number of students and the chosen chapters (topics). Students are involved in the research study work.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	George A. Bekey	Autonomous robots – From biological inspiration to implementation and control	The MIT Press, ISBN 0-262-02578-7	2005
2,	Rodney A. Brooks	Cambrian Intelligence – The Early History of the New AI	A Bradford Book, The MIT Press	1999
3,	Ronald Arkin	Behavior-based Robotics	The MIT Press, ISBN 0-262-01165-4	1998
4,	Vukobratović M., Borovac B., Surla D., Stokić D.	BIPED LOCOMOTION -Dynamics, Stability, Control and Application	Springer, ISBN 0-540-17456-7, ISBN 0-387-1745	1990

Table 5.2 Course specification

Course:		Selected Chapters in Production Process Automation
Course id:	HDOK-4	
Number of ECTS:	14	
Teachers:		Buchmeister S. Borut, Čuš -. Franci, Katalinić -. Branko, Palčič -. Iztok, Šešlija D. Dragan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The objective of the course is to obtain actual knowledge in the field of working process automation which is used in production and service systems and to introduce research problems.

2. Educational outcomes (acquired knowledge):

The outcome of the course is to obtain knowledge that enables students to systematically carry out working process automation in modern production and service systems as well as the knowledge and students` ability for independent and group research and research in this area.

3. Course content/structure:

Pneumatic, hydraulic and electrical systems automation. Energy efficiency of pneumatic systems. The quality of compressed air. Correlation requirements for air pressure and implementation methods. Effective filtration of compressed air. Automation filtering. Vacuum technology in automation.

4. Teaching methods:

Teaching activity is conducted through lectures and consultations. Preparation and defense of the scheduled project and passing the final examination. Prerequisite for taking the final examination is to complete and defend the project successfully. The final examination is written and refers to theoretical issues.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project defence	Yes	70.00	Theoretical part of the exam	Yes	30.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Groover P. Mikkell	Automation Production Systems and Computer Integrated Manufacturing	Prentice Hall	2003
2,	M. Stojiljkovć	Logička sinteza pneumatskog upravljanja	Mašinski fakultet, Niš	2002
3,	Šešlija, D., Lagod, B.	Stanje pneumatskih sistema u industriji Srbije sa aspekta energetske efikasnosti	Centar za automatizaciju i mehatroniku, Novi Sad	2006
4,	Šešlija D, Ignjatović I, Dudić S	Increasing the Energy Efficiency in Compressed Air Systems	InTech	2012
5,	Dudić S, Ignjatović I, Šešlija D, Blagojević V, Stojiljković M	Leakage quantification of compressed air using ultrasound and infrared thermography	Elsevier	2012

Table 5.2 Course specification

Course:		Selected Approach in Production Process Management
Course id:	IMDR14	
Number of ECTS:	14	
Teacher:		Tešić M. Zdravko
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The objective of the course is students` understanding of the state-of-the-art approach in the development of fundamental managerial field of study and study research.

2. Educational outcomes (acquired knowledge):

The outcome of the course is students` obtained knowledge for independent and group research and research work in basic areas of management.

3. Course content/structure:

- DZ-08 Access to working process management
- Just-In-Time, Lean Producton
- Virtual enterprise
- Agile manufacturing
- Management of business processes
- Intelligent enterprising

4. Teaching methods:

Lectures: (Mentor and student select one or more modules depending on their volume). Consultation. Lectures are conducted in combination. Presentation of the theoretical part is followed by the examples that clarify the theoretical part of the curriculum. In addition to lectures, consultations are held regularly. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Brown j., Harhen J., Shirnan J,	Production management systems	Addison-Wesley	1988
2,	Scheer AW., Krippke H., Kidermann H.	Agility by ARIS	Springer	2006

Table 5.2 Course specification

Course:		Effective Production and Service Systems
Course id:	IMDR31	
Number of ECTS:	14	
Teachers:		Ćosić P. Ilija, Katalinić -. Branko, Maksimović M. Rado, Šormaz N. Dušan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal of the course is to enable students to understand the latest approaches in the development of production and service technologies, the structure of production and service systems, their organization and management in accordance with their prior knowledge and interests, as well as to introduce research work in this particular field of study.

2. Educational outcomes (acquired knowledge):

The outcome of the subject is the knowledge and student`s ability to understand the issues of effective production and service systems and engage in research work in this field.

3. Course content/structure:

- Changes in the manufacturing and service systems.
- Contributions in development of production and service systems: CIM, Lean production, effective production systems.
- The principles in the development of production and service systems.
- Characteristics of production and service systems.
- Development of effective structures of production and service systems.
- Grouping on the basis of the classification system.
- Grouping based on similarity of procedures.
- The spatial structure and location system.
- Automation of processes of designing the structure of effective production and service systems.
- Simulation of production and service systems.
- Organization technology of effective production and service systems.

4. Teaching methods:

Lectures: (Mentor and student select one or more modules depending on their volume). Consultation. Lectures are conducted in combination. Presentation of theoretical part is followed by the examples that clarify the theoretical part of the curriculum. In addition to lectures, consultations are held regularly. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Zelenović, D.	Tehnologija organizacije industrijskih sistema - preduzeća	Univerzitet u Novom Sadu - Fakultet tehničkih nauka	2011
2,	Kay, J., Surresh, A.	Group Technology & Cellular Management - A state of-The-Art Synthesis of Research & Practice	Cluwer Pres, Buffalo - New York	1998

Table 5.2 Course specification

Course:	Structures of Modern Information and Communication Systems
Course id: IMDR33	
Number of ECTS: 14	
Teachers:	Krsmanović B. Cvijan, Ristić M. Sonja, Stefanović M. Darko
Course status:	Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Development of the awareness of the need for a multidisciplinary point of view and multimethodological approach to the research of modern information and communication systems. Overview and analysis of the different architectures of modern information systems, point out possible directions for their development. To enable students to participate in the development of new models and concepts of the development of information and communication systems.

2. Educational outcomes (acquired knowledge):

Students gain knowledge about the architecture of modern information systems; learn new and alternative approaches to research and design of information and communication systems.

3. Course content/structure:

The architecture of information systems. Distributed systems, hardware and software concepts. Client-server model. Service-oriented business models and information technology. Review of current networking technologies. The communication software and protocols. Internet services: traditional, contemporary and developmental trends. Web technologies to support new business models. Interoperability of information systems. The integration of data from different sources. Incomplete information systems with structured data. Mobile information systems and services.

4. Teaching methods:

Teaching activity depends on the number of students, i.e. mentor or frontal approach. During the course students are required to develop and defend a research paper.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Stallings W.	Data & Computer Communications	Prentice Hall, Inc.	2000
2,	Tanenbaum A, Maarten van Steen	Distributed Systems – Principles and Paradigms	Prentice Hall, Inc.	2002
3,	Douglas E. Comer	Internetworking With TCP/IP Volume 1: Principles Protocols, and Architecture, 5th edition	Prentice Hall, Inc.	2006
4,	Clements P., Kazman R., Klein M.	Evaluating Software Architectures - Methodes and Case Studies	Addison-Wesley	2006
5,	Clements P., Bachmann P., Bass L.	Documenting Software Architectures: Views and Beyond	Addison-Wesley	2002
6,	Taylor, R. N., Medvidovic N., Dashofy N.	Software Architecture: Foundations, Theory, and Practice	John Wiley&Sons	2010
7,	Silver Bruce	BPMN Method and Style, 2nd Edition, with BPMN Implementer's Guide: A structured approach for business process modeling and implementation using BPMN 2.0	Cody-Cassidy Press	2011

Table 5.2 Course specification

Course:		Behavioral Corporate Finance
Course id:	IMDR47	
Number of ECTS:	14	
Teacher:		Dobromirov P. Dušan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Teaching activity enables students to master the concept of bihevioral approach in finance, with strategic choices of measures and analysis of operational implications. The most important educational objectives are: 1) definition of action and the importance of psychological factors in decision making in finance, 2) introduction to the key psychological factors that occur in different areas of corporate finance, 3) understanding the errors that occur in making decisions due to psychological factors.

2. Educational outcomes (acquired knowledge):

Students will gain knowledge in the field of behavioral corporate finance and learn about the latest trends in finance.

3. Course content/structure:

1) The definition of behavioral finance
2) Determining the value of projects
3) Capital budgeting
4) Risk
5) Inefficient markets and corporate decisions
6) Capital Structure
7) Dividend Policy
8) Conflict of Interest and Corporate Governance
9) Group Processes
10) Mergers and Acquisitions

4. Teaching methods:

Lectures. Consultations. Seminar paper.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Lecture attendance	Yes	20.00	Oral part of the exam	Yes	40.00
Term paper	Yes	40.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Hersch Shefrin	Bihejviorističke korporativne finansije	McGraw-Hill	2007
2,	Dobromirov Dušan; Radišić Mladen i Aleksandar Kupusinac	Emerging Markets Arbitrages' Perception:Risk vs. Growth Potential	African Journal of Business ManagementVol. in press (AJBM-10-060Dobromirov et al)	2010

Table 5.2 Course specification

Course:		Media Systems
Course id:	IMDR49	
Number of ECTS:	14	
Teacher:		Radenković B. Vladimir
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Acquiring the necessary knowledge regarding the functioning of media systems. Students will gain insight into the significance and strength necessary to use a media system for each organization.

2. Educational outcomes (acquired knowledge):

Engineer of management will be able to adequately apply acquired knowledge in their research. You will have the requisite skills and competencies to create new areas of application, media systems in order to increase the effectiveness of business organizations.

3. Course content/structure:

Media Policy, Media Strategy; Media in the function of the integrated economy; Links between media systems, social systems and the audience; Media in Education; Effects of media; Corporate social responsibility of media; Public service; Media regulation; Media convergence; Media Sustainability; Social Media; International decisions, documents, organizations; Impact of new technologies on media; Media positioning; Distribution of media content.

4. Teaching methods:

The method of oral presentation, method calls, work with individuals. Teaching includes lectures and exercises. Evaluation of knowledge is done through an oral exam and seminar work as prerequisites given.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	60.00	Oral part of the exam	Yes	40.00

Literature

Ord.	Author	Title	Publisher	Year
1,	-	Television in Europe: regulations, policy, independence	Open Societe Institute (OSI)	2006
2,	Lowe, G. H. and Bardoel, J.	From Public Service Broadcasting to Publice Service Media	Nordicom, Gőteborg, Sweden.	2007
3,	Radenković, V.	Business practices in corporations of radio and television cable distribution programmes in Serbia	Journal for East European Management Studies (JEEMS)	2010
4,	Radenković, V., Radenković, M., Engus, K.	Media and Social Responsible Business-Serbian Model	African Journal of Business Management	2010

Table 5.2 Course specification

Course:		Strategic Development of Human Resources
Course id:	IMDR52	
Number of ECTS:	14	
Teacher:		Duđak D. Ljubica
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal of course is to master the necessary skills to strategically place the development of human resources in an organization, that is, recognizing the link between the success and development of contemporary organizations and the development of its human resources.

2. Educational outcomes (acquired knowledge):

Students will (1) be trained to recognize the importance of strategic human resource development in contemporary organizations, (2) be familiar with the needs and possibilities of the development of different strategies that organizations can define the process of acquiring adequate knowledge of human resources and the development of competitive knowledge, (3) be familiar with the characteristics of the concept of "learning organizations" in modern business and development opportunities and building a "learning organization", (4) able to develop an effective plan for the development of human resources in an organization, and (5) familiar with the operational aspects of the development process, that is, training workers.

3. Course content/structure:

Context of human resource development, human resource management versus human resource management - the debate and the implications for human resource development, strategic basis for the concept of human resource development, strategic human resource development and human resource development strategies, interventions from training to teaching staff as a way of life - Analysis of organizational culture for the development of an effective learning environment, organizational dimensions of human resource development, concept of "learning organization" and the application of modern business, transformational change management from the perspective of human resource development, human resource development role in creating synergies of the organization, developing human resources, building organizational values ??(commitment, business ethics, diversity management), process development and training of employees - operational aspects

4. Teaching methods:

Teaching is done through lectures, study research and consultation during the preparation of the project. The essence of the approach to the teaching of subjects Strategic human resource development in the use and application of theoretical knowledge in the analysis of case studies from real organizations.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Beardwell, I., Holden, L., Claydon, T.	Human Resource Management	Prentice Hall, Harlow, England	2004
2,	Becker, B.E., Huselid, M.A., Ulrich, D.	The HR Scorecard – Linking People, Strategy and Performance	Harvard Business School Press, Boston	2001
3,	Kearns, P	HR Strategy – Business focused, individually centred	Butterworth Heinemann - Elsevier, London	2003
4,	Reid, M.A., Barrington, H., Brown, M.	Human Resource Development	CIPD House, London	2004
5,	Walton, J.	Strategic Human Resource Development	Prentice Hall, Pearson Education, Harlow, England	1999
6,	Ivancevich, J.M.	Human Resource Management	McGraw-Hill Irvin, New York	2007
7,	Hristić, D., Grubić Nešić, L., Duđak, Lj.,	The Differences in Approaching Management by Managers of Different Gender – an Example from Serbia	African Journal of Business Management,	2011

Table 5.2 Course specification

Course:		Data Research
Course id:	IMDR55	
Number of ECTS:	14	
Teachers:		Ćulibrk R. Dubravko, Mirković R. Milan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Obtaining advanced knowledge in the field of data mining.

2. Educational outcomes (acquired knowledge):

Students will obtain the knowledge and skills that enable them to effectively use applied techniques of artificial intelligence and machine learning for data mining. They will be familiar with various aspects of computers as tools for data mining, detection of structural scheme of the data, presentation and use of discovered knowledge.

3. Course content/structure:

The course will cover the following areas: review of main concepts of data mining, the typical sources and data preparation, decision trees, neural networks, support vector machines, clustering of data, analysis and presentation of data that have temporal and spatial dimension. Theoretical instruction will be accompanied by training in practical use of open source solutions for data mining.

4. Teaching methods:

Auditory and laboratory, seminar paper and oral examination.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Ian H. Witten & Eibe Frank	Data Mining - Practical Machine Learning Tools	The Morgan Kaufmann	2005
2,	Fosca Gianotti & Dino Pedreschi Eds.	Mobility, data mining, and privacy: geographic knowledge discovery	Springer-Verlag	2008
3,	Culibrk, D., Marques, O., Socek, D., Kalva, H., Furht, B.	Neural Network Approach to Background Modeling for Video Object Segmentation	IEEE Transactions on Neural Networks	2007
4,	D Culibrk, M Mirkovic, V Zlokolica, M Pokric, V Crnojevic, D Kukolj	Salient Motion Features for Video Quality Assessment	IEEE transactions on image processing	2010

Table 5.2 Course specification

Course:		Selected Chapters in Hydraulic Systems
Course id:	IMDR58	
Number of ECTS:	14	
Teacher:		Jocanović T. Mitar
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Knowledge gained from the subject is used in practice addressing present issues related to the operation and exploitation of hydraulic systems and filtering.

2. Educational outcomes (acquired knowledge):

The student is competent to apply the acquired knowledge to solve problems related to hydraulic systems in practice, and the acquisition of practical skills for independent and team work as well as in scientific research in fields related to hydraulics.

3. Course content/structure:

1.Hydraulic automation systems, 2 Energy efficiency of hydraulic systems. 3.Selected chapters in logical components in hydraulic, 4. Selected chapters in hydraulic power steering, 5 Selected chapters in proportional hydraulics, 6 Impact of variability working regime of physical and chemical properties of fluid, 7 Impact of changes in operation modes to the work of hydraulic components and systems, 8 The issue of exploitation of lubricants in hydraulic systems, 9 Obliteration of fluid power systems, 10. The presence of contaminants in hydraulic system and their impact on performance and service life of components and systems, 11. Problem of filtering, 12. Recycling and the problem of processing used lubricants in the field of hydraulics.

4. Teaching methods:

Lectures: (Co-mentor and student select one or more topics depending on the scope and problems of thematic areas). Consultation. Lectures are delivered in combination with active participation of students. Delivering the theoretical part is followed by the examples to clarify the theoretical part of the curriculum. Part of the teaching activity is carried out through an independent study research in the field of hydraulics. Student`s research work includes active monitoring of primary scientific sources, organization and experiments as well as statistical data processing, numerical simulations, writing a paper about an issue regarding the scientific area of doctoral dissertations.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project task	Yes	40.00	Written part of the exam - tasks and theory	Yes	60.00

Literature

Ord.	Author	Title	Publisher	Year
1,	I.T.Hong, K. Izawa, T. Ito	Evaluation of Cilinder, Solenoid valve and Servovalve Contaminant Sensitivity	Fluid Power Reseach Center Oklahoma State University	1984
2,	V.Savić, D. Knežević, D.Lovrec, M.Jocanović, V.Karanović	Determination of Pressure Losses in Hydraulic Pipeline Systems by Considering Temperature and Pressure	Strojniški Vestnik-Journal of Mechanical Engineering	2009
3,	G. E. Totten, D.K. Wills, D.G.Feldmann	Hydraulic Failure Analysis: Fluids, Components, and System Effects	ASTM, West Conshohocken	2001
4,	Wolfgang Bock	Hydraulik-Fluide als Konstruktionselement	Vereinigte Fachverlage, Mainz	2007
5,	T.Christopher Dickenson	Filters and Filtration Handbook	Elsevier	1979
6,	E.C.Fitch,	Fluid Contamination Control	Fluid Power Reseach Center Oklahoma	1988

Table 5.2 Course specification

Course:		Selected chapters of enterprise's management and control
Course id:	IMDR69	
Number of ECTS:	16	
Teachers:		Maksimović M. Rado, Tešić M. Zdravko
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal of course is to master the basic concepts and approaches that allow the definition of the global architecture of the system, the consistency of decision making across the business system, monitoring the flow models whose activities go beyond the limits of functions, business process management and real-time basis for the definition of enterprise business process improvement.

2. Educational outcomes (acquired knowledge):

Students will be able to participate in the creation of different types of organizational and management model of enterprises with the aim of building a complete representation of enterprises, which consists of the definition of the mission, strategy, key performance indicators (KPI), business processes and competencies and their relations to improve synergies within the company and fulfill the mission and vision of an effective and efficient manner. In addition, students will be able to use tools that allow companies to share key information / knowledge to achieve business process coordination and cooperative decision-making, and achieve enterprises integration.

3. Course content/structure:

Structure of enterprises. The functional approach to the organization of business processes. Process approach in the organization of business processes. Methodological approaches to modeling business processes. Business Process Reengineering. Architectures for enterprise integration. CIMOSA and GRAI concepts. ARIS modeling approach and the integration of business processes. Specifics of modeling service organizations and public sector enterprises. PLM as a concept of enterprise integration. Enterprises Interoperability - the basic framework. Integration of information technology in enterprises. Enterprise systems and their integration (ERP, SCM, BPMS). Key performance indicators. Measuring the performance of business processes. Practical examples of the organization, management and integration processes in the enterprise.

4. Teaching methods:

To achieve the set goals of education in the learning process are using a combination of lectures with presentation software solutions and case studies supported by applicative systems for analysis and modeling organizational structures and business processes. Case studies are used to lay the practical basis and show students how to analyze, model and improve business processes in real-life situations

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Burbidge, J.L.	Production Flow Analysis	Clarendon Press, Oxford	1989
2,	Zelenović, D., Ćosić. I., Maksimović, R.	Design and Reenginering of Production Systems: Yugoslavian (IISE) Approaches, Vol. I6 in Monograph "Group Technology and Cellular Manufacturing", State of-The-Art Synthesis of Research and Practice	Kluwer Academic Publishers, Massachusetts	1998
3,	Zelenović, D.	Tehnologija organizacije industrijskih preduzeća	Fakultet tehničkih nauka	2005
4,	Tešić, Z., Lalić,D.,Ćosić, I., Mitrović, V.	Integration of information for manufacturing shop control	University of Ljubljana	2010
5,	Waldman D., Jensen E.	Industrial Organization	Prentice Hall	2012
6,	Hammer, M., Champy, J.	Reengineering the corporation	Harper Business	2001

Table 5.2 Course specification

Course:		Advanced risk assessment methods
Course id:	IMDR72	
Number of ECTS:	14	
Teacher:		Sakulski M. Dušan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

To gain a knowledge and understanding regarding advanced disaster risk assessment methods

2. Educational outcomes (acquired knowledge):

Students will be capable to apply a contemporary mathematical and statistical disaster risk assessment tools regarding various natural and human induced hazards

3. Course content/structure:

This course will implement the advanced risk assessment methods. Students will focus on the advanced assessment of the basic risk parameters such as hazard, vulnerability, exposure and resilience. Special attention will be on the probabilistic risk assessment methods. After the course completion students will be able to apply knowledge gained.

4. Teaching methods:

Lectures, computer based exercises and consultation.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Exercise attendance	Yes	5.00	Written part of the exam - tasks and theory	Yes	50.00
Lecture attendance	Yes	5.00			
Term paper	Yes	15.00			
Test	Yes	25.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Tim Bedford and Roger Cooke	Probabilistic Risk Analysis: Foundations and Methods	Cambridge	2001
2,	Patrizia Grossi	Catastrophe Modeling: A New Approach to Managing Risk	Springer	2005

Table 5.2 Course specification

Course:		Industrial eco-marketing management
Course id:	IMDR82	
Number of ECTS:	14	
Teachers:		Bunčić M. Sonja, Nikolić T. Slavka, Vojinović-Miloradov B. Mirjana
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Understanding of eco-products / brands as a modern phenomenon and sustainable ecological imperatives of sustainable development, health security, economic production and environmental improvement. Decision-making in the field of eco-marketing in terms of ecological development of the economy, industry and social development.

2. Educational outcomes (acquired knowledge):

The ability of the optimal management of eco-marketing in terms of environmental engineering, sustainable development and eco-marketing success in all spheres.

3. Course content/structure:

Challenges, strategies and new approaches to eco-marketing. Fitting between traditional and eco / green marketing. Standards, laws, guidelines and recommendations. BAT (Best Available Techniques) and BEP (Best Environmental Practice). Stockholm and Basel Convention. Eco-marketing and environmental engineering, production eco-modification, eco-packaging. The main segments of the integrated sustainable eco-marketing: eco-design, shape, color, eco-positioning. Promotion of organic products and eco-marketing. The main areas of eco marketing: product and productivity in the function of preventing contamination of the environment and the elimination of existing and potential ecological damage. Price of products focused on environmental packaging and organic production. The financial benefit of eco-marketing, fellowship and mutual funds in the eco-marketing. Eco-law. Friendly and eco-oriented marketing activities, substitution of hazardous products with eco-products. Urban metabolism, productivity, flows of hazardous materials, safety and eco-marketing.

4. Teaching methods:

Lectures (mentor with a student chooses one or more modules, depending on the scope of the module). Consultation. Lectures are conducted in combination. Leaving the theoretical part is followed by examples. In addition to lectures are held regularly and consultation. Through study research student, studying scientific journals and other literature, self deepens the material from the lecture. In addition to working with the teacher, student is trained to write your own scientific work.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Exercise attendance	Yes	5.00	Coloquium exam	No	30.00
Lecture attendance	Yes	5.00	Oral part of the exam	Yes	60.00
Test	Yes	30.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Webster, F. E.	Industrial Marketing Strategy	New York: JohnWiley & Sons.	1991
2,	Ottman, J.A.	Green Marketing Opportunity for Innovation	NTC Business Books, Chucago	1998
3,	Dragan A. Marković, Šimon A. Đarmati, Ivan A. Gržetić et al	Fizičkohemijski osnovi zaštite životne sredine - Izvori zagađivanja, posledice i zaštita, II	Univerzitet u Beogradu	1996
4,	Al Iannuzzi	Greener Products: The Making and Marketing of Sustainable Brands	CRC Press	2011
5,	Nikolić, T.S.; Ćosić, I.; Miletić, A., Pečujlija, M	The Effect of the 'Golden Ratio' on Consumer Behaviour	African Journal of Business Managemen,Vol. 5(20), pp. 8347-8360	2011
6,	Nikolić, S. et al.	Industrijski eko-marketing	FTN - Novi Sad	2013
7,	Wilson, R. M. S. and Gilligan,C.	Strategic Marketing Management: Planning, implementation and control	Elsevier, Amsterdam	2005
8,	Kuhre, W.L.	ISO 14020s Environmental labeling-marketing, efficient and accurate environmental marketing procedures	New York: Prentice Hall PTR	1996
9,	Graedel T.E., and B.R. Allenby	Design for Environment	Prentice Hall, Inc. Simon & Schusters/A Viacom Company Upper Saddle River	1996

Table 5.2 Course specification

Course:		Quality abd organisational performance
Course id:	IMDR83	
Number of ECTS:	14	
Teachers:		Beker A. Ivan, Jocanović T. Mitar, Kamberović L. Bato, Milisavljević M. Stevan, Radlovački S. Vladan, Šević D. Dragoljub
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The course is designed as a base for examining the most important outcomes of the quality management system - increase in organizational performance. Students will be introduced to the approaches in research methods of relations between quality management and organizational performance. By observing the nature of relations between these important aspects, students will be trained for the research which are aimed towards effective improvements of in organization

2. Educational outcomes (acquired knowledge):

Upon passing the exam, students will be able to explore the relations between quality management and organizational performance, and to apply existing knowledge in order to achieve effective organization and quality management system improvements. Course provides fundamental knowledge of relationship between quality management and organizational performance dimensions (elements), which serves as guidance of organizational efforts towards effective improvements.

3. Course content/structure:

Quality management system. Quality dimensions. Organizational performance. Examination of the relationship between quality management and organizational performance. Improvements which are based on studies of relationship between quality management and organizational performance. Performance in unfavorable environment.

4. Teaching methods:

Lectures, research work, consultations. The rating is based on the success of the project and an oral exam.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Radlovački V., Pečujlija M., Kamberović B., Jovanović R., Delić M., Beker I.	SATISFACTION OF HIGH SCHOOL STUDENTS WITH THE APPLICABILITY OF THEIR KNOWLEDGE	TTEM. Tehnics tehnologies education management, 2012, Vol. 7, No 2, pp. 777-785, ISSN 1840-1503	2012
2,	Jovanović R., Radlovački V., Pečujlija M., Kamberović B., Delić M., Grujić J.	Assessment of blood donors' satisfaction and measures to be taken to improve quality in transfusion service establishments	Medicinski glasnik (BiH), 2012, Vol. 9, No 2, pp. 231-237	2012
3,	Radlovački V.	Opšti procesni model i ocenjivanje efikasnosti sistema menadžmenta kvalitetom u skladu sa zahtevima serije standarda ISO 9000	FTN Izdavaštvo, Novi Sad	2011
4,	Grupa autora	Metode i tehnike unapređenja procesa rada	FTN i IS-ITC Novi Sad	2012
5,	Grupa autora	SISTEM MENADŽMENTA KVALITETOM	FTN i IIS-ITC Novi Sad	2012

Table 5.2 Course specification

Course:		Data ACQUISITION, ANALYSIS AND INTERPRETATION 1
Course id:	IMDR84	
Number of ECTS:	14	
Teachers:		Pečujlija D. Mladen, Vrgović D. Petar
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The subject aims to enable students to understand many basic concepts, processes, and issues that arise when performing empirical studies in most psychological and managerial disciplines, and thus create a conceptual basis for later studies in facilities that include this type of knowledge.

2. Educational outcomes (acquired knowledge):

Students are trained in-house research design, data collection, data processing, univariate procedures, interpretation of data and preparation of reports on research conducted using the SPSS software package.

3. Course content/structure:

Preparation of research, design research design, data collection, analysis and interpretation of results, and preparation of the report on the investigation. Uzrorkovanje. Levels of measurement (nominal, ordinal, interval, ratio). Design of research tools. Frequency, correlation and factorial research designs. Student t test. Chi-square analysis. Univarjantna analysis of variance (ANOVA). Multivariate analysis of variance (MANOVA). Regression analysis. Within each of the three groups of drawings appear gradually from simpler to more complex types. At the end of the course describes the structure of a standard written report on the investigation. During the course, for illustration shows a large number of (mostly simplified) examples of research in many areas of management.

4. Teaching methods:

Lectures, computer exercises and consultations.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Computer exercise attendance	Yes	5.00	Oral part of the exam	Yes	30.00
Project	Yes	30.00	Practical part of the exam - tasks	Yes	20.00
Project task	Yes	15.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Dejan Todorović	Osnovi metodologije psiholoških istraživanja	DPS	1994
2,	Nunnally, J.M	Psychometric theory	McGRAW-HILL, INC	1994
3,	Stanislav Fajgelj	Metode istraživanja ponašanja	Centar za primenjenu psihologiju, Beograd	2004
4,	Mladen Pečujlija	Initiating innovation in Serbian companies' organizational cultures	Academic Journals.	2010

Table 5.2 Course specification

Course:		Effective technological and production structures
Course id:	IMDR85	
Number of ECTS:	14	
Teachers:		Ćosić P. Ilija, Lazarević M. Milovan, Maksimović M. Rado, Radaković J. Nikola, Šormaz N. Dušan, Tešić M. Zdravko
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal of the course is to enable students to understand the latest approaches in the development of technological and production structures in accordance with their prior knowledge and interests, as well as to introduce research work in this particular field of study.

2. Educational outcomes (acquired knowledge):

The outcome of the subject is the knowledge and student`s ability to understand the issues of effective technological and production structures and engage in research work in this field.

3. Course content/structure:

Changes in the technological and production structures. Approaches in development of technological and production structures. The principles in the development of technological and production structures. Characteristics of technological and production structures. Automation of processes of designing the technological and production structures. Simulation of technological and production structures.

4. Teaching methods:

Lectures: (Mentor and student select one or more modules depending on their volume). Consultation. Lectures are conducted in combination. Presentation of theoretical part is followed by the examples that clarify the theoretical part of the curriculum. In addition to lectures, consultations are held regularly. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Zelenović, D.	Inteligentno privređivanje	Prometej, Novi Sad	2011
2,	Maksimović, R.	Složenost i fleksibilnost struktura industrijskih sistema	Fakultet tehničkih nauka u Novom Sadu	2003

Table 5.2 Course specification

Course:		Advanced topics on Innovation and Entrepreneurship
Course id:	IMDR70	
Number of ECTS:	16	
Teacher:		Borocki V. Jelena
Course status:		Elective

Number of active teaching classes (weekly)

Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of the selected course is to improve and develop theoretical and empirical topics from innovation and entrepreneurship. The students will be able to (1) assess the changes, trends and impacts of different factors from the innovation and entrepreneurship field, to (2) identify strategies and ways how changes were implemented within the organization (product or service company), and (3) to analyze the impact of changes on level of innovation activities and entrepreneurship in existing enterprises (SMEs, companies - multinational, large, industries, institutions supporting entrepreneurship and innovation, et al.). Also, students should understand the impact of a dynamic business environment in creating innovative corporate strategy and innovation management strategies.

2. Educational outcomes (acquired knowledge):

Students who pass the exam will be able to (1) obtain the understanding of advanced research methodologies and approaches in selected areas, (2) compare and analyze the principles and theoretical approaches within several traditional and modern approaches to innovation and entrepreneurship, (3) demonstrate research skills in a process of critical examination all relations between theoretical explanations, methods, research issues and questions and empirical data in the selected area, (4) apply the knowledge and techniques to analyze specific research in the field.

3. Course content/structure:

The nature of entrepreneurship and opportunities - an introductory elements, based on identifying business opportunities in the region, sources of opportunities, active research and discovery, relations and capabilities of the business concept. Market elements of commercialization opportunities - research techniques, assessment of the size of market opportunities. Business ideas and testing the feasibility of the business idea, promotion, creation of business ideas in organizations. Detection of entrepreneurial opportunities and decision models. The concept of innovation - different methods of research; assessement the use of certain models in the changing conditions of the external environment. Business Models - Innovation processes, entrepreneurship, the development of the organization. Analysis of the results of different studies in the field of innovation, entrepreneurship and technology. Identification and selection of key elements of the research. Analysis of different techniques tools and models to gain a competitive advantage through innovation

4. Teaching methods:

Lectures. Consultations. Student independently deepens the subject-matter learned at lectures through his research work while studying scientific journals and other literature. In addition to working with the teacher, students are trained to write their own scientific papers.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Katic, Penezic, Borocki, Zekic	Entrepreneurship significance in restructuring process	TTEM – Technics, Technologies Education Management	2011
2,	Borocki, J., Cosic, I., Lalic, B., Maksimovic, R.	Analysis of company development factors in manufacturing and service company: a strategic approach	Strojniski vestnik - Journal of Mechanical Engineering, Ljubljana	2010

Table 5.2 Course specification

Course:		Selected topics of project management
Course id:	IMDR71	
Number of ECTS:	16	
Teachers:		Lalić P. Bojan, Morača D. Slobodan, Radaković J. Nikola
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of the course is that students master advanced approach to project management and specific knowledge necessary for the successful implementation of the project. During the teaching process, students will be introduced to modern techniques and tools integration process, time management, cost, quality, communications, risk and supply, as well as the procedures for the development and improvement of existing approaches, tools and techniques of project management.

2. Educational outcomes (acquired knowledge):

After completing this course, students will be able to manage complex projects, using modern approaches, tools and techniques for scientific research in this field.

3. Course content/structure:

The new project management approaches; Modern techniques and tools of project management;
Project management according to internationally recognized standards software packages for project management, Lean Project Management, Change Management, Development tools and techniques of project management, Agile project management methods.

4. Teaching methods:

Lectures, Auditory Practice, Laboratory Practice and Consultations. Lecturing method is based on the multimedia lectures and practice. During lectures problem frame is presented and facts and theoretical approach is analyzed, while the practice is interactive and practical in the form of laboratory practice. Besides lectures and practice, consultations are held on a regular basis. Lecturing method plans for at least 40% of the time to be devoted to the active participation of students, which includes working in the laboratory and visits to production and service organizations.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature				
Ord.	Author	Title	Publisher	Year
1,	Grupa autora	Korpus znanja za upravljanje projektima, četvrto izdanje	FTN	2010

Table 5.2 Course specification

Course:		Selected Chapters in Investment Management
Course id:	IMDR35	
Number of ECTS:	16	
Teacher:		Gradojević J. Nikola
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of this course is to enable students to understand the latest approaches in the narrow area of investment management and to introduce the research in this area.

2. Educational outcomes (acquired knowledge):

The outcome of the course is to obtain skills for independent and group research and research in the field of investment management.

3. Course content/structure:

Financial markets;International Finance; Money and banking; Exchange transactions; E-business; Strategic management; Corporate governance; Corporate finance; Entrepreneurial finance; Management investments

4. Teaching methods:

Lectures: (Mentor and student select one or more modules depending on their volume). Consultation. Lectures are conducted in combination. Presentation of theoretical part is followed by the examples that clarify the theoretical part of the curriculum. In addition to lectures, consultations are held regularly. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Zvi Bodie, Alex Kane, Alan Marcus	Investments	McGraw-Hill/Irwin	2010
2,	Ramo Gençay, Nikola Gradojevic, Faruk Selcukand Brandon Whitcher	Asymmetry of Information Flow between Volatilities Across Time Scales	Quantitative Finance	2010
3,	Ramo Gençay and Nikola Gradojevic	Crash of 87 - Was it Expected? Aggregate Market Fears and Long Range Dependence	Journal of Empirical Finance	2010
4,	Nikola Gradojevic, Ramo Gençay and Dragan Kukolj	Option Pricing with Modular Neural Networks	IEEE Transactions on Neural Networks	2009
5,	Nikola Gradojevic	Non-linear, Hybrid Exchange Rate Modelling and Trading Profitability in the Foreign Exchange Market	Journal of Economic Dynamics and Control	2007

Table 5.2 Course specification

Course:		Selected chapters from Information management
Course id:	IMDR73	
Number of ECTS:	16	
Teachers:		Bošković M. Dragan, Ćulibrk R. Dubravko, Krsmanović B. Cvijan, Mirković R. Milan, Ristić M. Sonja, Stefanović M. Darko
Course status:		Elective

Number of active teaching classes (weekly)

Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Introduction of students in selected field of information management and their preparation for independent research work. Consideration of information technology development perspectives and their applications in engineering management. Studying of actual approaches and methods in research work oriented to advancement of management procedures in business systems and their working processes.

2. Educational outcomes (acquired knowledge):

Introducing of students with modern development trends and approaches in problem solving processes in the field of information management. Students preparation for high-grade and accurate problems recognition and their solving using by scientific and research methods. Development and advancement of students creative component in individual and team work.

3. Course content/structure:

Contemporary information technologies and their development trends. Information technologies as a condition of success in the manager work. Management of information systems development in modern enterprises. Agile approaches in development of software products and systems purposed to support in manager work. Empirical software engineering. Contemporary data base systems and approaches in exploitation of data. Contemporary systems in business resources planning. Fundamentals and development of business intelligence systems. Electronic government systems. Case studies for applications of information technology means in engineering management.

4. Teaching methods:

A student, together with his advisor, select one or more of modules from the subject, dependency of its volume. Lectures are combined (theoretical considerations and analysis of practical examples). Consultations are usual. During of work with professors, a student are preparing himself for writing of scientific articles in selected scientific field.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Clarke, S.	Information systems strategic management	Routledge Information systems Textbooks	2001
2,	Cockburn, A.	Agile Software Development	Addison/Wesley	2001
3,	Hawking, P.	Enterprise resource planning systems in a global environment	IGI Global	2008
4,	Homburg, V.	Understanding e-government: Information systems in public administration	Routledge	2008
5,	Tan, P., Steinbach, M., Kumar, V.	Introduction to data mining	Addison - Wesley	2006
6,	Vercelis, C.	Business intelligence: Data mining and optimization for decision making	Wiley	2009
7,	Juristo, N., Moreno, A.	Basics of software engineering experimentation	Springer - Verlag	2001
8,	Kimball, R., Ross, M.	The data warehouse toolkit: The complete guide to dimensional modeling	John Wiley & Sons	2011
9,	Johnston, T., Weis, R.	Managing time in relational databases: How to design, maintain and query temporal data	Morgan - Kaufmann	2010

Table 5.2 Course specification

Course:		Selected Topics in Quality Management and Logistics
Course id:	IMDR74	
Number of ECTS:	16	
Teachers:		Beker A. Ivan, Filipović V. Jovan, Jocanović T. Mitar, Kamberović L. Bato, Milisavljević M. Stevan, Radlovački S. Vladan, Šević D. Dragoljub
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The course introduces students to research in this area is characterized by an intense and innovative development. Students will become familiar with the development of the area in the past two decades, and the latest research and forecasts about developments in the future. The knowledge acquired will enable students a thorough understanding of the field of quality and logistics, which will form the basis for independent research.

2. Educational outcomes (acquired knowledge):

After completing courses and passing the exam, students will master the existing models developed in the particular area. Students will also gain the ability to create research and to critically analyze existing processes, quality management and logistics.

3. Course content/structure:

Logistics, supply chain management, quality management system, environmental management system, Health and Safety, Continuity of Systems

4. Teaching methods:

Lectures, research work, consultations. The rating is based on the success of the project and an oral exam.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Radlovački V., Beker I., Majstorović V., Pečujlija M., Stanivuković D., Kamberović B.	Quality Managers' Estimates of Quality Management Principles Application in Certified Organisations in Transitional Conditions - Is Serbia Close to TQM	Strojniški vestnik - Journal of Mechanical Engineering, 2011, Vol. 57, No 11, pp. 851-861, ISSN 0039-2480	2011
2,	Hiroyuki Hirano	JIT Implementation Manual - The Complete Guide to Just-in-Time Manufacturing	Volume 1-6, CRC Press	2009
3,	Paul C. Husby and Dan Swartwood	Fix your supply chain : how to create a sustainable lean improvement roadmap	Productivity Press, 2009, ISBN-13: 978-1-56327-381-0	2009

Table 5.2 Course specification

Course:	Selected Topics in Risk Management and Insurance Management
Course id: IMDR75	
Number of ECTS: 16	
Teachers:	Avdalović A. Veselin, Ćosić I. Đorđe
Course status:	Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal of this course is to introduce students to the process of risk management, and technical and technological consequences of the execution risk, as well as contemporary processes of insurance

2. Educational outcomes (acquired knowledge):

After passing the exam, students will be trained in the proper analysis of the risk, its assessment and management methods of the same

3. Course content/structure:

analysis, risk assessment, risk management, risk management cycle, emergency response, reconstruction response, preparedness, mitigation, prevention, risk management current trends, satellite systems, geoinformation technology, satellite images, insurance and reinsurance companies as well as professional carriers rzika, Monte Carlo simulation, CAT-NET Munich Re.

4. Teaching methods:

Lectures, exercises and consultations

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Lecture attendance	Yes	10.00	Oral part of the exam	Yes	50.00
Test	Yes	40.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Avdalović S., Ćosić Đ., Avdalović V.	Osnove osiguranja sa upravljanjem rizikom	FTN	2010
2,	Harrington, Niehaus	Risk management and insurance	The McGraw Hill Companies	2004

Table 5.2 Course specification

Course:		Selected topics in industrial marketing and media engineering
Course id:	IMDR76	
Number of ECTS:	16	
Teachers:		Kamberović L. Bato, Lalić S. Danijela, Nikolić T. Slavka, Radenković B. Vladimir, Radlovački S. Vladan, Ratković-NJegovan M. Biljana
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Educational objective: Gaining insight and understanding the complexities of industrial marketing and media engineering, and the necessity of a multidisciplinary approach for problems solving in a given scientific field.

2. Educational outcomes (acquired knowledge):

Gaining abilities for scientific research in the field.

3. Course content/structure:

Fundamentals and present trends in industrial marketing and media engineering. Tendencies in specific behavior of industrial users. Customer involvement and specific customer needs (Customer Co-Creation) in product design and development. Modern holistic approach in industrial marketing management. Modern media application. Media as a function of industrial systems. Quality management and marketing.

4. Teaching methods:

Lectures (mentor with a student chooses one or more modules, depending on the module scope). Consultations. Combined lectures. Theoretical part is followed by adequate examples. Through research study of scientific journals and other literature, student gains further insights in the field. With given mentorship, student is trained for writing scientific paper.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Michael H. Morrris; Leyland F. Pitt; Earl D. Honeycutt, Jr.	Business-to-Business Marketing: A Strategic Approach	Sage Publications, London	2001
2,	Michael D. Hutt, Thomas W. Speh	Business Marketing Management	South-Western	2007
3,	Nikolić, T.S.; Pečujlija, M.	Customer behavior in the culture of fear and short attention	African Journal of Business Managemen, Vol. 6 (9), pp. 3147-3155	2012
4,	Zdravko Tešić, Vojin Mitrović, Ilija Ćosić, Danijela Lalić	Integration of Information for Manufacturing Shop Control	Strojniski vestnik - Journal of Mechanical Engineering 56 (2010) 3, pp. 217-223	2010
5,	Slavka T. Nikolć, Slobodan Miladinović	'Customized' Consumer and Consumer 'Innovator' in the Light of Social Capital and Dominant Cultural Pattern	5th International Conference on Mass Customization Marketing and Personalization in Central Europe	2012
6,	Danijela Lalić, Slađana Gajić, Valentin Konja	Social Media Influence on Mass Customization and Personalization Process	5th International Conference on Mass Customization Marketing and Personalization in Central Europe	2012
7,	Vladimir Radenković	Business practices in corporations of radio and television cable distribution programmes in Serbia	Journal for East European Management Studies	2010

Table 5.2 Course specification

Course:		Selected Chapters from Human Resource Management
Course id:	IMDR77	
Number of ECTS:	16	
Teachers:		Duđak D. Ljubica, Grubić-Nešić S. Leposava, Katić R. Ivana, Lalić S. Danijela, Pečujlija D. Mladen, Vrgović D. Petar
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal of this course is to introduce students to the basic principles and roles of human resource management in the engineering management.

2. Educational outcomes (acquired knowledge):

Knowledge about the practical implications of human resource management in a framework of organizational management, especially related to engineering management.

3. Course content/structure:

Organization's cultural climate of the organization; Knowledge Economy in the organization; Leadership and changes; Teamwork; Stress and conflict; Engineering Psychology; Motivating employees; Protection of employees.

4. Teaching methods:

Teaching is done interactively, with the active participation of students in the teaching process. Increased number of exercises has the aim to explain the students theoretical and practical approaches.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Homework	Yes	20.00	Theoretical part of the exam	Yes	20.00
Term paper	Yes	20.00	Oral part of the exam	Yes	40.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Desler,H.	Human Resource management	Prentice Hall	2005
2,	Cabrilo, S.; Grubic-Nesic, L.	„ The role of creativity, innovation and invention in knowledge management“, in Buckley, S. and Jakovljevic, M (ed.) Knowledge Management Innovations for Interdisciplinary Education: Organisational Applications,	IGI Global	2012
3,	Gragg,L.,Cassell,J.	Progress in management Engineering	Nova Science Publisher	2009

Table 5.2 Course specification

Course:		Odabrana poglavlja iz energetskog menadžmenta
Course id:	IMDR78	
Number of ECTS:	16	
Teacher:		Gvozdenac D. Dušan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Energy undoubtedly a strong influence on the national and regional economic and social development. This course introduces the chill of the energy flows in the building industry and in order to fully examine the opportunities and needs of their management. Energy efficiency and renewables are a great modern means of which they can make to reduce environmental pollution and production costs. This course covers many areas of energy and gives students the opportunity to get to know the basic tools used to analyze and create efficient energy management at the regional level, business or office building.

2. Educational outcomes (acquired knowledge):

• The acquisition of knowledge in the field of energy menadžmena and training for implementation, implementation and monitoring of the SIST ISO 50001 energy management ENMS.
• Knowledge of the flows of materials and energy production systems and
• Identify solutions for saving energy power systems
• Designing, implementing and monitoring the implementation of energy management.

3. Course content/structure:

Energy in Industry, Energy in Buildings, Energy Efficiency in Energy, Renewable Energy, an integrated policy on energy and the environment; Motivation industry to improve energy efficiency and environmental protection; legal framework for energy management and energy efficiency in the European Union (EU), the concept of energy management, energy management systems, and environmental impacts; Industrial energy systems (industrial steam systems, industrial elektrosistemi, cooling systems, industrial cogeneration); Energy in Buildings (energy requirements of buildings, heat consumption in buildings, thermal protection of buildings, energy infrastructure in buildings, measures to increase energy efficiency in buildings), measurement and verification (M & V) (Determination of energy conservation, monitoring of energy flows, Plan M & V and the summary of the effects Examples of M & V procedures for some parts of the energy system, Manage M & V procedures), design of energy efficient systems and components (effective, optimal and nearly optimal design; concept project, decision matrix, development of the concept; risk in the implementation of energy projects).

4. Teaching methods:

Lectures and workshops. Leaving the theoretical part of the following examples and calculations independent student.
Through study research student in consultation with the supervisor systematically processed following the set subject relevant scientific literature.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	60.00	Oral part of the exam	Yes	40.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Morvay Z K, Gvozdenac D D	Applied Industrial Energy and Environmental Management	John Wiley	2009
2,	Gvozdenac D, Gvozdenac-Urošević B, Morvaj Z	Energetska efikasnost (industrija i zgradarstvo)	FTN	2012
3,	Gvozdenac D, Nakomčić-Smaragdakis B, Gvozdenac-Urošević B	Obnovljivi izvori energije	FTN	2012
4,	-	ISO 50001 EnMS (Energy Management System)	ISO	-

Table 5.2 Course specification

Course:	Selected chapters in enterprise's design, organization and control
Course id: IMDR5	
Number of ECTS: 16	
Teachers:	Ćosić P. Ilija, Maksimović M. Rado, Radaković J. Nikola, Tešić M. Zdravko
Course status:	Elective

Number of active teaching classes (weekly)

Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Acquiring the latest knowledge about design, organization and control methods of enterprises structure, based on group technology, manufacturing cells and the development of production, organizational and control structures with the ability to maintain an independent working existence. Mastering the techniques of applying methods of group approaches in the design, classification and analysis of trends in production and application of these methods and techniques in designing and revitalization of company production, organizational and control structures.

2. Educational outcomes (acquired knowledge):

Necessary knowledge and skills of the students for independent and group research and further research work in the field of design, organizing and control in enterprise. Acquiring skills for project management of construction or rehabilitation of production and organizational structures suitable for management.

3. Course content/structure:

Basics of group technology in manufacturing. Method of design, organization and control based on the classification of objects of production and structures capable of maintaining an independent existence of labor. Methods of design, organization and control based on the FFA, GA, LA, and PFA analysis. Method of design, organization and control based on Lean Principles. Case studies.

4. Teaching methods:

Lectures. Consultations. Seminar paper. Theoretical part of the subject-matter is followed by the examples due to clarify this part of the curriculum. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Burbidge, J.L.	Production Flow Analysis	Clarendon Press, Oxford	1989
2,	Zelenović, D., Ćosić. I., Maksimović, R.	Design and Reenginering of Production Systems: Yugoslavian (IISE) Approaches, , Vol. I6 in Monograph "Group Technology and Cellular Manufacturing", State of-The-Art Synthesis of Research and Practice	Kluwer Academic Publishers, Massachusetts	1998
3,	Shahrukh, A.I.	Handbook of Cellular Manufacturing Systems	John Wiley & Sons	1999

Table 5.2 Course specification

Course:		Selected chapters in automation			
Course id:	IMDR80				
Number of ECTS:	16				
Teachers:		Stankovski V. Stevan, Šešlija D. Dragan, Borovac A. Branislav, Ostojić M. Gordana, Dudić P. Slobodan			
Course status:		Elective			
Number of active teaching classes (weekly)					
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:	
5	0	0	4	0	
Precondition courses		None			

1. Educational goal:

The educational goal is to introduce doctoral students in the chosen area of automation that is used in modern engineering industry.

2. Educational outcomes (acquired knowledge):

Outcomes are the knowledge and skills of students in independent and team research and research in the field of automation in industrial engineering.

3. Course content/structure:

A review of research in the fields of sensor, actuator, control systems, robotic systems, system integration, communication protocols, systems for automatic identification.

4. Teaching methods:

Mentor a student chooses one or more areas, depending on the scope of the field. Consultation. Lectures are conducted in combination. Leaving the theoretical part is followed by examples which serve to clarify the theoretical part of the curriculum. In addition to lectures are held regularly and consultation. Through study research student, studying scientific journals and other literature deepens own curriculum with lectures. In addition to working with the student teacher is trained to write your own scientific work in the chosen field.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Gajić G., Stankovski S., Ostojić G., Tešić Z., Miladinović Lj.	Method of evaluating the impact of ERP implementation critical success factors–a case study in oil and gas industries	Enterprise Information Systems	2012
2,	Stankovski S., Ostojić G., Šenk I., Rakić-Skoković M., Trivunović S., Kučević D.	Dairy cow monitoring by RFID	Scientia Agricola	2012
3,	Dudić, S., Ignjatović, I., Šešlija, D., Blagojević, V., Stojiljković, M.	Leakage quantification of compressed air using ultrasound and infrared thermography	Measurement	2012
4,	Ignjatović, I., Šešlija, D., Tarjan, L., Dudić S.	Wireless sensor system for monitoring of compressed air filters	Journal of Scientific and Industrial Research	2012

Table 5.2 Course specification

Course:		Selected topics in quality engineering and logistics
Course id:	IMDR79	
Number of ECTS:	16	
Teachers:		Beker A. Ivan, Filipović V. Jovan, Jocanović T. Mitar, Kamberović L. Bato, Milisavljević M. Stevan, Radlovački S. Vladan, Šević D. Dragoljub
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The course introduces students to research in this area is characterized by an intense and innovative development. Students will become familiar with the development of the area in the past two decades, and the latest research and forecasts about developments in the future. The knowledge acquired will enable students a thorough understanding of the field of quality and logistics, which will form the basis for independent research.

2. Educational outcomes (acquired knowledge):

After completing courses and passing the exam, students will master the existing models developed in the particular area. Students will also gain the ability to create research and to critically analyze existing processes in the field of quality engineering and logistics.

3. Course content/structure:

Reliability, techniques and technologies in maintenance, logistics, supply chains, measuring and control technology

4. Teaching methods:

Lectures, research work, consultations. The rating is based on the success of the project and an oral exam.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Jocanović M., Šević D., Karanović V., Beker I., Dudić S.	Increased Efficiency of Hydraulic Systems Through Reliability Theory and Monitoring of System Operating Parameters	Strojniški vestnik - Journal of Mechanical Engineering, 2012, Vol. 58, No 4, pp. 281-288, ISSN 0039-2480	2012

Table 5.2 Course specification

Course:	Selected chapters from Information, management and communication systems
Course id: IMDR81	
Number of ECTS: 16	
Teachers:	Bošković M. Dragan, Ćulibrk R. Dubravko, Krsmanović B. Cvijan, Mirković R. Milan, Ristić M. Sonja, Stefanović M. Darko
Course status:	Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Introduction of students in selected field of information, management and communication systems and their preparation for independent research work. Consideration of information technology development perspectives and their applications in industrial systems engineering . Studying of actual approaches and methods in research work oriented to advancement of management procedures in industrial systems and their working processes.

2. Educational outcomes (acquired knowledge):

Introducing of students with modern development trends and approaches in problem solving processes in the field of information, management and communication systems in industry. Students preparation for high-grade and accurate problems recognition and their solving using by scientific and research methods. Development and advancement of students creative component in individual and team work.

3. Course content/structure:

Contemporary information technologies and their development trends. Information technologies as a condition of effectiveness in industrial systems work. Management of information systems development in modern industrial systems. Agile approaches in development of software products and systems purposed to support in manufacturing and production management. Empirical software engineering. Contemporary data base systems and approaches in exploitation of data. Contemporary systems in manufacturing resources planning. Fundamentals and development of business intelligence systems. Case studies for applications of information technology means in industrial systems engineering.

4. Teaching methods:

A student, together with his advisor, select one or more of modules from the subject, dependency of its volume. Lectures are combined (theoretical considerations and analysis of practical examples). Consultations are usual. During of work with professors, a student are preparing himself for writing of scientific articles in selected scientific field.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Clarke, S.	Information Systems Strategic Management	Routledge Information Systems Textbook	2001
2,	Cockburn, A.	Agile Software Development	Addison - Wesley	2001
3,	Warner, T.	Communication Skills for Information Systems	Pearson Education Ltd.	1996
4,	Hawking, P.	Enterprise Resource Planning Systems in a Global Environment	IGI Global	2008
5,	Tan, P. N., Steinbach, M., Kumar, V.	Introduction to Data Mining	Addison - Wesley	2006
6,	Vercelis, C.	Business Intelligence: Data Mining and Optimization for Decision Making	Wiley	2009
7,	Juristo, N., Moreno, A.	Basics of Software Engineering Experimentation	Springer - Verlag	2001
8,	Elmasri, R., Navathe, S.	Database Systems: Models, Languages, Design and Application Programming	Pearson Education Ltd.	2011

Table 5.2 Course specification

Course:	Preparation for the Application of Doctoral Dissertation Topic
Course id: SID05	
Number of ECTS: 2	
Teachers:	
Course status:	Mandatory

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
0	0	0	2	0

Precondition courses: None

1. Educational goal:

Overview of situation in the area of the proposed topic for doctoral dissertation based on the scientific literature analysis – books, monographs, papers in referential journals, papers from conference proceedings, available documentation at websites, etc. The objective is to overview the possibilities of the thesis and scientific potential of the topic.

2. Educational outcomes (acquired knowledge):

Study on the potentials of the proposed doctoral dissertation topic, i.e. the systematized knowledge in the area of the research topic for doctoral dissertation, as well as clear directions in further research on the topic.

3. Course content/structure:

Defining the wider area of the doctoral dissertation topic and key motives for research. Overview of literature on the basis of available scientific books, monographs, papers in referential journals, papers from conference proceedings, available documentation at websites, etc. Study on the potentials of the proposed doctoral dissertation topic.

4. Teaching methods:

Teaching is performed as tutorials.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	70.00	Oral part of the exam	Yes	30.00

Literature				
Ord.	Author	Title	Publisher	Year
1,	Priznati naučnici i stručnjaci iz oblasti teme Dr teze	Razna naučna dela		sve

Table 5.2 Course specification

Course:		Motion controla and application of MEMS
Course id:	HDOL13	
Number of ECTS:	14	
Teachers:		Ivandić I. Željko, Jovanović M. Vukica, Kozak V. Dražen, Ostojić M. Gordana, Stankovski V. Stevan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of this course is to master the knowledge necessary for designing and implementing systems for motion control.

2. Educational outcomes (acquired knowledge):

Outcomes of the course are the knowledge that primarily cover the field of linear motion, and include sensors, actuators and control algorithms used for manipulation devices, machines and systems.

3. Course content/structure:

Exploring the possibilities of application of linear motion system with: servopneumatikom, servohidraulikom, DC motors, AC motors, servo motors. Research applications of sensors: proximity, position, pressure, velocity, flow. Exploring possibilities of MEMS as accelerometers, gyroscopes, pressure sensors.

4. Teaching methods:

Mentor and student select one or more modules depending on their volume. Consultation. Lectures are delivered in combination. Delivering the theoretical part is followed by the examples that clarify the theoretical part of the curriculum. In addition to the lectures, consultations are held regularly. While studying scientific journals and other literature student independently deepens subject-matter delivered at lectures. In addition to working with the teacher, students are trained to write their own scientific work.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature				
Ord.	Author	Title	Publisher	Year
1,	Tan K. K., T. H. Lee and S. Huang	Precision motion control: Design and implementation, 2nd ed.,	London, Springer	2008
2,	Robert H. Bishop	TheMechatronicsHandbook	CRC PRESS	2002
3,	Andrzej Pawlak	Senzors and Actuators in Mechatronics, Design and Applications	Taylor&Francis	2007

Table 5.2 Course specification

Course:		Nonindustrial automation
Course id:	HDOL14	
Number of ECTS:	14	
Teachers:		Ivandić I. Željko, Jovanović M. Vukica, Kozak V. Dražen, Ostojić M. Gordana, Stankovski V. Stevan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of this course is to enable students to understand the modern approach of the application of automation in thermal systems and research in this area.

2. Educational outcomes (acquired knowledge):

Outcomes are student`s knowledge and skills for independent and group research and further research work in this area.

3. Course content/structure:

Automation in residential and commercial buildings. Monitoring energy consumption in buildings. Control. The application of automation in education. A part of teaching activity is accomplished through an independent study research in the field of non-industrial automation. Research work includes active monitoring of primary scientific sources, organizing and conducting experiments and statistical data processing as well as writing a paper regarding a topic in the field of study.

4. Teaching methods:

Lectures: (Mentor with the student selects one or more modules depending on its volume). Consultation. Lectures are conducted in combination. Presentation of the theoretical part is followed by the examples that clarify the theoretical part of the curriculum. In addition to the lectures, consultations are held regularly. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Stankovski, S., Tarjan, L., Škrinjar, D., Ostojić, G., Šenk, I.	Using a Didactic Manipulator in Mechatronics and Industrial Engineering Courses	IEEE Transactions on Education	2010
2,	Ostojić, G., Stankovski, S., Tarjan, L., Šenk, I., Jovanovic, V.	Development and Implementation of Didactic Sets in Mechatronics and Industrial Engineering Courses	International Journal of Engeneering Education	2010
3,	Grupa autora	Odabrani radovi sa SCI liste		2010

Table 5.2 Course specification

Course:		COGNITIVE MANAGEMENT
Course id:	IMDR10	
Number of ECTS:	14	
Teachers:		Pečujlija D. Mladen, Vrgović D. Petar
Course status:		Elective

Number of active teaching classes (weekly)

Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses

1. Educational goal:

Introduce students to basic concepts of cognitive management, raising awareness of the relevance of cognitive management as applied management discipline, developing awareness of and openness to interdisciplinary cooperation touch with scientific disciplines, exploring the scientific and practical aspects of the problem. Introduction, application and development of standard methods (including experimental testing) and research techniques in management. Introduction to the method of application of psychological knowledge, theory and research to solve problems in practical work.

2. Educational outcomes (acquired knowledge):

Mastering the cognitive principles and regularities of human economic behavior and ways of application of psychological knowledge and principles to the design, editing and predicting economic behavior of both individuals and groups, and understanding the nature of human interaction and psychological characteristics of human psychological processes and economic behavior.

3. Course content/structure:

Introductory considerations. Definition of cognitive management. Predictors of cognitive management. Cognitive management as parallel and sequential process. Terror management theory and cognitive management. Cognitive psychology of learning and management. Attitudes and cognitive management. Values and cognitive management. Emotions and cognitive management. Cognitive development and management. Motivation and cognitive management. Cultural, gender and age aspects of economic behavior. Cognitive management in crisis situations. The concept of justice and cognitive management. Cognitive and tax management. Emotional branding. Changing attitudes: the central and lateral strategies. Connotative and denotative meaning. Hemispheric strategies in information processing and decision making. Psychological aspects of adopting a new economic system. Methods and techniques of research

4. Teaching methods:

Lectures, case studies, practical exercises and consultations.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Exercise attendance	Yes	5.00	Oral part of the exam	Yes	30.00
Lecture attendance	Yes	5.00			
Presentation	Yes	10.00			
Project defence	Yes	30.00			
Term paper	Yes	20.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Pecujlija, M. et al	Employees' Attitudes Toward Company Privatization as Possible Predictors of a High-Performance Work System	African Journal for Business and Management	2010
2,	Kirchler, E.	The economic psychology of tax behaviour	Cambridge University Press.	2007
3,	Pecujlija, M. et al	Questionnaire and EFA as Tools for Researching Employee's Assumptions Despite of Scheins Opposite Claims	African Journal for Business and Management	2010
4,	Anand, Stephen Lea	The psychology and behavioural economics of poverty	Journal of Economic Psychology	2011

Table 5.2 Course specification

Course:		Employees' creativity management
Course id:	IMDR11	
Number of ECTS:	14	
Teacher:		Vrgović D. Petar
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Goal of the course is to master the basic and advanced skills necessary to analyze, measure and manage the creative potentials of employees in organizations. The course aims to familiarize students with the rules and principles, which are used to optimally manage creative power of all employees in the organization in order to maximize their potential. On the basis of this course competences for research and active management of the creative forces of the whole working organization will be aquired.

2. Educational outcomes (acquired knowledge):

On the basis of goals achieved in the course, it is expected that students will be able to independently design and conduct research processes aimed at obtaining information on the level of creative potential of employees in the organization, about the factors that affect them, as well as their degree of utilization. The outcome of the course will be the students' competences for proper and optimal use of the creative potential of the employees in organizations through a systematic approach, with emphasis on mastering systems of idea management and creative stimulation.

3. Course content/structure:

Thematic sections: Factors and scientific approaches to the concept of creativity, creative potential measurement, researching stimulating and hindering factors of creativity, managing the creative force of employees,idea management systems, open innovation.

4. Teaching methods:

Teaching is done interactively, with the active participation of students in the teaching process and with organizing empyrical research of the observed phenomena.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	30.00	Written part of the exam - tasks and theory	Yes	30.00
Term paper	Yes	20.00	Oral part of the exam	Yes	20.00

Literature

Ord.	Author	Title	Publisher	Year
1,	DeGraff J., Lawrence K.A.	Creativity at Work - Developing the Right Practices to Make Innovation Happen	John Wiley & Sons, Inc.	2002
2,	von Stamm B.	Managing Innovation, Design and Creativity	John Wiley & Sons Ltd	2003
3,	VanGundy, A.B.	Getting to innovation : how asking the right questions generates the great ideas your company needs	AMACOM	2007
4,	Bilton K.	Menadžment i kreativnost	Clio, Beograd	2010

Table 5.2 Course specification

Course:		Organizational structures			
Course id:	IMDR12				
Number of ECTS:	14				
Teachers:		Borocki V. Jelena, Maksimović M. Rado			
Course status:		Elective			
Number of active teaching classes (weekly)					
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:	
5	0	0	4	0	
Precondition courses		None			

1. Educational goal:

The goal of the course is to enable students to learn, develop and design different organizational structures necessary to plan, organize, lead and control processes.

2. Educational outcomes (acquired knowledge):

Students will obtain knowledge and competencies for analyzing processes, different types of organizational structures and for resolving practical problems in ant type of the company.

3. Course content/structure:

Vision, mission, goals, objectives and politics of the company; stakeholders, processes and their relations inside the company; different flows in the company; types of organizational structures; how to design effective organizational structure; organizational changes

4. Teaching methods:

Lectures are auditory with theoretical treatment of the required number of case studies. Practice include auditory introduction to issues in focus, interactive processing of case studies and specific class projects due to use the acquired knowledge for real organizational structure design. Individual work - case study of one company from environment.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Zelenović, D.	Tehnologija organizacije industrijskih sistema - preduzeća	Fakultet tehničkih nauka u Novom Sadu	2012
2,	Maksimović, R.	Složenost i fleksibilnost struktura industrijskih sistema	Fakultet tehničkih nauka u Novom Sadu	2003

Table 5.2 Course specification

Course:	Raster and Image Processing Technologies in Engineering and Management
Course id: IMDR34	
Number of ECTS: 14	
Teachers:	Ćulibrk R. Dubravko, Krsmanović B. Cvijan, Mirković R. Milan
Course status:	Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The objective of the course is to train students for basic and applied research work in the field of raster technology and image processing and further enhance their prospects for the use of resources and tools based on these technologies in industrial engineering and management.

2. Educational outcomes (acquired knowledge):

Opening new horizons of research in this area and define new areas of application of the underlying technology in industrial engineering and management. Practical means and tools to master the subject area and their application in research.

3. Course content/structure:

Fundamentals and mathematical basis of raster technology. Principles and means of digitization of documents and images. Structure and form of document in raster presentation. Raster presentation - fields and methodology. Recognition based on raster presentations. Cryptology. Processing digital documents and images. Principles and Methods of Image Processing. Pattern vectorization of raster presentation. Entity recognition. Application in media, industry and military technology. Research in the field of machine and robo-vision.

4. Teaching methods:

Teaching activity depends on the number of listeners, i.e. mentor or frontal approach.
During the course, students are required to prepare a seminar paper and its defense.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	60.00	Oral part of the exam	Yes	40.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Gonzalez, R., Woods, R. E.	Digital Image Processing, 3rd Edition	Prentice Hall	2007
2,	Umbaugh, S.	Computer Imaging: Digital Image Analysis and Processing	Prentice Hall, Inc.	2005

Table 5.2 Course specification

Course:		Advanced Data Models and Database Systems
Course id:	IMDR36	
Number of ECTS:	14	
Teacher:		Ristić M. Sonja
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Introducing students to advanced data models and database systems. Students learn to engage in actual projects in the field of databases.

2. Educational outcomes (acquired knowledge):

Understanding the contemporary data models and database systems and acquiring knowledge and skills required for the use of advanced techniques for BP design.

3. Course content/structure:

Contemporary data models and database systems and their development trends. Distributed databases. The integration of data from different sources. Data warehouse systems. XML databases. Spatial databases. Temporal databases. Case studies for the application of contemporary data models and database systems.

4. Teaching methods:

Teaching activity depends on the number of listeners, i.e. mentor or frontal approach.
During the course, students are required to prepare a seminar paper and its defense.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Elmasri R, Navathe S. B,	Fundamentals of Database Systems, 5th Edition	Addison Wesley	2006
2,	Malinowski E., Zimányi E.	Advanced Data Warehouse Design; From Conventional to Spatial and Temporal Applications	Springer	2008
3,	A.K. Elmagarmid; A.P. Sheth	Distributed and Parallel Databases; An International Journal	Springer US	2009
4,	K.-Y. Whang; P.A. Bernstein; C.S. Jensen	The VLDB Journal; The International Journal on Very Large Data Bases	Springer	2009
5,	Kashyap V., Bussler C., Moran M.	The Semantic Web; Semantics for Data and Services on the Web	Springer	2008
6,	Kutsche R-D., Milanovic N.	Model-Based Software and Data Integration; First International WS, MBSDI 2008, Berlin, Germany, April 2008	Springer	2008
7,	Akmal B. Chaudhri Awais Rashid Roberto Zicari	XML Data Management: Native XML and XML-Enabled Database Systems	Addison-Wesley	2003

Table 5.2 Course specification

Course:		CAE/CAD/CAM and CIM Concepts and Systems
Course id:	IMDR37	
Number of ECTS:	14	
Teacher:		Krsmanović B. Cvijan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Development of multicriteria review and systematic approach to using computer supported technologies in developing and designing new processes and reengineering existing products. Introducing listeners in research aimed at developing and implementing effective production processes and procedures based on computer integration of manufacturing, Rapid Prototyping and Rapid Manufacturing concepts.

2. Educational outcomes (acquired knowledge):

Listeners need to acquire certain knowledge and skills in computer aided modeling and redesign of industrial products, engineering analysis based on the digital product model, a highly productive design and modern approaches to documenting and archiving research and development results. As part of the subject, the audience will developed a clear vision of future product development and engineering design, and industrial production as a whole.

3. Course content/structure:

Industrial product as a technical system. Form, structure and metrics as the basic definition of the product. Engineering design and information technology to support development and product design. Computer aided modeling: principles, methods and tools. CSG and B-Rep model of the principles of building components. Sweeping Method. The principles of automated formation of higher-level application. The procedures and methods of computer aided engineering analysis. Construction supported by software tools. Documenting and archiving - concept and functions of digital archives. Design procedures in the production and assembly. Computer integration of manufacturing, fundamentals of CIM. Rapid Prototyping and Rapid Manufacturing methods and processes in modern industrial production.

4. Teaching methods:

Frontal approach in teaching activity; mentor approach in the case of small number of students.
During the course, students are required to prepare a seminar paper and its defense.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Groover, M. P., Zimmers, E. W.	CAD/CAM: Computer Aided Design and Manufacturing	Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632	1984
2,	Magrab, E. B.	Integrated Product and Process Design and Development: The Product Realization Process	CRC Press LLC, 2000 Corporate Blvd., N. W., Boca Raton,	1997
3,	Krsmanović, C.	Automatizacija projektovanja u industrijskom inženjerstvu; knjiga I: Principi i sredstva automatizacije projektovanja pr	Fakultet tehničkih nauka, Novi Sad, Republika Srbija	1997

Table 5.2 Course specification

Course:		Production control structure			
Course id:	IMDR38				
Number of ECTS:	14				
Teacher:		Tešić M. Zdravko			
Course status:		Elective			
Number of active teaching classes (weekly)					
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:	
5	0	0	4	0	
Precondition courses		None			

1. Educational goal:

To achieve the set goals of education in the learning process using a combination of lectures, with presentation software solutions, and case studies supported software products for the implementation of a system for managing work processes. Case studies are used to lay the practical basis and show students how to apply different technologies in realistic industrial enterprises.

2. Educational outcomes (acquired knowledge):

Students will be able to participate in the creation of diverse and non-standard systems management that are incurred as requirements of different types of organizational structures and production companies. In addition, students will be able to apply the theoretical methods and techniques developed in the case studies show that the solutions to achieve business process coordination and cooperative decision-making, and how to manage the process of integration in the enterprise.

3. Course content/structure:

Structure of enterprises. Approached in the organization of business and production processes of enterprises. Specifics of the service organization's system and public sector enterprises Process approach in setting up the organizational structure. Production structure of the company. Access OPT - Optimum production technology. Approach - PBC - Management at equal intervals. Management in terms of group technology. Controls structure in LEAN manufacturing. Running a virtual system products. Information and communication technologies for the management of work processes. Implementation of SAP system in the management and production management. Practical examples of the organization, management and integration processes in the enterprise.

4. Teaching methods:

To achieve the set goals of education in the learning process using a combination of lectures, with presentation software solutions, and case studies supported software products for the implementation of a system for managing work processes. Case studies are used to lay the practical basis and show students how to apply different technologies in realistic industrial enterprises.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Dickersbach J, Keller G	Production planning and control with SAP	SAP PRESS	2010

Table 5.2 Course specification

Course:	Application of Information and Satellite Technologies in Risk Management
Course id: IMDR45	
Number of ECTS: 14	
Teacher:	Popov B. Srđan
Course status:	Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of this course is to enable students to understand the modern approach in the application of information and satellite technology in the field of risk management.

2. Educational outcomes (acquired knowledge):

The course outcomes are student`s knowledge and skills for independent and group research and research work in this area.

3. Course content/structure:

The reasons and the need for application of information and satellite technology. The present situation in the field of satellite technology. The connection between information and satellite technology. Modern software tools for the application of technology in risk management. Examples of application of technology in all phases of the risk management cycle.

4. Teaching methods:

Lectures. Consultation. Lectures are delivered in combination (traditional instruction and distance learning). Delivering the theoretical part is followed by the examples that clarify the theoretical part of the curriculum. Consultations are held regularly in addition to lectures. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Michelle K. Hall , C. Scott Walker , Anne Huth , Robert F. Butler, Larry P. Kendall, Jeff S. Jenness	Exploring the Dynamic Earth: GISInvestigations for the Earth Sciences	ESRI	2009
2,	Michelle K. Hall , C. Scott Walker , Anne Huth , Robert F. Butler, Larry P. Kendall, Jeff S. Jenness	Exploring Tropical Cyclones: GIS Investigations for the Earth Sciences	ESRI	2009
3,	Ćosić Đ., Popov S., Sakulski D., Pavlović A	Geo-Information Technology for Disaster Risk Assessment	Acta Geotechnica Slovenica	2010
4,	Sakulski D.	Web-enabled GIS in Disaster Management	The Global Magazine for Geomatics	2005
5,	Michelle K. Hall , C. Scott Walker , Anne Huth , Robert F. Butler, Larry P. Kendall, Jeff S. Jenness	Exploring Water Resources: GIS Investigations for the Earth Sciences	ESRI	2009

Table 5.2 Course specification

Course:		Advanced Risk Management
Course id:	IMDR48	
Number of ECTS:	14	
Teacher:		Gradojević J. Nikola
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of this course is to enable students to understand the latest theoretical and practical knowledge in the narrow field of financial risks (including financial engineering) and the introduction of the research work in this area.

2. Educational outcomes (acquired knowledge):

The course outcome is student`s knowledge and training for independent and group research and research work in the management of financial risks (including financial engineering).

3. Course content/structure:

Forward (forward) and futures (futures) contracts and hedging (hedging) exposure to financial risks; price estimate options (option pricing) using a binomial and Black-Scholes model of dynamic hedging; indicators of financial risk (Value-at-Risk, Cashflow-at-Riski.td); SWAPS and their use, and financial engineering (exotic derivatives and related financial products).

4. Teaching methods:

Lectures. Consultations. Seminar paper.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Lecture attendance	Yes	20.00	Oral part of the exam	Yes	40.00
Term paper	Yes	40.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	René M. Stulz	RiskManagementandDerivatives	Thomson, South-Western	2003
2,	John C. Hull	Options, Futures and Other Derivatives	Prentice Hall	2008
3,	Ramo Gençay and Nikola Gradojevic	Crash of 87 - Was it Expected? Aggregate Market Fears and Long Range Dependence	ournal of Empirical Finance	2010
4,	Nikola Gradojevic, Ramo Gençay and Dragan Kukolj	Option Pricing with Modular Neural Networks	IEEE Transactions on Neural Networks	2009
5,	Nikola Gradojević	Overnight Interest Rates and Aggregate Market Expectations	Economics Letters	2008

Table 5.2 Course specification

Course:		Media Research
Course id:	IMDR50	
Number of ECTS:	14	
Teacher:		Radenković B. Vladimir
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Acquiring the necessary knowledge in the field of research. Students will become familiar with the strategies and tools for media research, as well as with all relevant conditions in which modern organizations operate, and which have significant implications for media research results that the organization are using to improve the implementation of their communication goals within overall business objectives.

2. Educational outcomes (acquired knowledge):

Application of knowledge in research on filling specific organizational requirements. Engineering Management will be able to apply their research skills in order to get the desired information relevant to the implementation of the communication and media plan in any organization. Also, the improvement of their competence there is a possibility of opening new areas of application of all aspects of media research.

3. Course content/structure:

Investigation of the media and society; profit and objectives of media organizations in the media industry and market, media relations and environment, media and the public; Media changes in the direction of individualization and multiplication of consumer choice and audience fragmentation, modernization, globalization and commercialization of the media, Selected Topics in Media Control; Quantitative and qualitative measurement of diversity, balance, social benefits and production values of media content; Comparative study on the interaction of media, technology and communication, subjective and objective measures to assess the quality of picture and sound.

4. Teaching methods:

Classes are taught through verbal lectures and audio-visual exercises, with the introduction of the theoretical background of media research and learning based on practical examples. Testing knowledge is done through the development of the paper, which is obligation before final exam, and final oral examination.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	60.00	Oral part of the exam	Yes	40.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Picard, R. G.	Assessment of Public Service Broadcasting: Economic and Managerial Performance Criteria, The Public/Javnost, Vol. 10, No. 3, pp. 29–44.		2006
2,	--	Research Report on European Television Stations(2006), http://www.nuns.rs/dosije/19/12.jsp	-	2006
3,	McQuail, D.	Mass Communication Theory	Sage Publications	2005
4,	Radenković, V., Radenković, M., Engus, K.	Media and Social Responsible Business-Serbian Model	African Journal of Business Management	2010
5,	Radenković, V.	Business practices in corporations of radio and television cable distribution programmes in Serbia	Journal for East European Management Studies (JEEMS)	2010

Table 5.2 Course specification

Course:		Organisational Behavior
Course id:	IMDR51	
Number of ECTS:	14	
Teacher:		Grubić-Nešić S. Leposava
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The behavior of the employees is the most important factor of success. The complexity of studying the employees` behavior is conditioned by the fact that organizational behavior as a function of organizational culture, structure, personal characteristics, value and economic context on which the organization is based. Employed as carriers of human capital, their knowledge, skills, motivation, experience, form the basis of the development of the organization. The aim of this course is to introduce students to the basic laws of organizational behavior, and factors that determine it. Another aim is to master knowledge and skills important for directing, managing and developing desired organizational behavior.

2. Educational outcomes (acquired knowledge):

Students learn about the basic laws of organizational behavior, determinants, master the tools to diagnose current and desired behavior, and models of development and improvements in their results.

3. Course content/structure:

1. Organizational design and employees` behavior
2. Organizational culture
3. Staff - abilities, skills, knowledge
4. Organizational Communication
5. Team performance
6. Management Styles
7. Stress
8. Development of organizational behavior

4. Teaching methods:

Lectures and exercises

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	40.00	Oral part of the exam	Yes	60.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Robbins,S.	Organizational Behavior	PrenticeHall	1998
2,	McShane, Von Glinow,	Organizational Behavior - essentials	McGraw- Hill/Irwin	2007
3,	McKenna,E.	Business Psychology and Organisational Behavior,	Psychology	2007
4,	Petkovic,M.,	Organizaciono ponasanje	Press, Ekonomski fakultet, Beograd	2003
5,	Kirin,S.,Grubić-Nešić,L.,Cosic,I.,	Increasing a Large Petrochemical Company Efficiency by Improvement of Decision Making Process	Hemijska industrija ISSN 0367-598X	2010

Table 5.2 Course specification

Course:		Selected Chapters in Life Insurance
Course id:	IMDR53	
Number of ECTS:	14	
Teachers:		Mrkšić Lj. Dragan, Lisov R. Milimir
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of this course is to enable students to master and learn about the latest trends in the life insurance industry with special focus on new products and the European Union directives relating to life insurance and their implementation in practice of insurance companies.

2. Educational outcomes (acquired knowledge):

The reasons and the need for signing a life insurance. State life insurance in our country and the world. The possibility of implementing new life insurance contract and directives of the European Union in Serbia.

3. Course content/structure:

-The new life insurance contracts that are used in most developed countries, and still have no application or application is still expected in the market of life insurance in Serbia
-Contract-saving (savings), unit link, risk life insurance, life insurance with additional risk in case of serious illness
-EU Directive relating to life insurance and that Serbia will be obliged to apply as a condition for joining the European Union
-Contemporary models of life insurance commition as a precondition for the development of life insurance procedure applied in most developed countries-implementation of ISO procedures and instructions in the branch of life insurance
-Bank assurance in the life insurance industry
-New trends in the voluntary pension insurance

4. Teaching methods:

Lectures (traditional and distance learning). Consultation. Introduction to the theoretical concepts of insurance policy in the most developed countries worldwide. Students research work, scientific journals and other literature that enable them to write their own scientific papers with the assistance of their teacher.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Žarković Nebojša, Mrkšić Dragan i Lisov Milimir	Situation and possibilities of improvement of voluntary pension insurance in Serbia as a developing country	African Journal of Business Management	2010
2,	Mrkšić Dragan, Petrović Zdravko	Životna osiguranja	DIS Public, Beograd	2008
3,	Lisov Milimir	Privatno penziono osiguranje	Centar za automatizaciju i mehatroniku, Novi Sad	2006
4,	Ćurković Marjan	Ugovor o životnom osiguranju	CROACIA, Zagreb	2008
5,	Mašić Nikola	Životno osiguranje	Naklada autora, Zagreb	2008

Table 5.2 Course specification

Course:		Computer Vision in Industrial Engineering and Management
Course id:	IMDR54	
Number of ECTS:	14	
Teachers:		Crnojević S. Vladimir, Ćulibrk R. Dubravko
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The acquisition of advanced knowledge in the field of computer vision and extraction of information from multimedia contents (pictures and videos).

2. Educational outcomes (acquired knowledge):

Students will obtain the knowledge and skills that enable them to effectively apply the techniques of using images and video, artificial intelligence and machine learning to extract information from multimedia content. They will be introduced to different problems in the domain computer vision and basic techniques used to solve them.

3. Course content/structure:

The course will cover the following areas: techniques for coding and storing pictures and videos, image segmentation based on texture and color, object detection, classification, texture, detection of moving objects, tracking moving objects, detection of interesting behavior of objects and subjects. Theoretical classes will be complemented by hand-on training in the use of open source computer vision software to solve practical computer-vision problems.

4. Teaching methods:

Auditory and laboratory, seminar paper and oral exam.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	70.00	Oral part of the exam	Yes	30.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Rafael C. González, Richard Eugene Woods	Digital image processing	Pearson/Prentice Hall	2008
2,	Gary Bradski, Adrian Kaehler	Learning OpenCV: Computer Vision with the OpenCV Library	O`Reilly Media	2008
3,	Culibrk, D., Marques, O., Socek, D., Kalva, H., Furht, B.	c Modeling for Video Object Segmentation	EEE Transactions on Neural Networks	2007
4,	D Culibrk, M Mirkovic, V Zlokolica, M Pokric, V Crnojevic, D Kukolj	Salient Motion Features for Video Quality Assessment	IEEE transactions on image processing	2010
5,	Petrovic, N.I., Crnojevic, V.	Universal Impulse Noise Filter Based on Genetic Programming	IEEE transactions on image processing	2008

Table 5.2 Course specification

Course:		Traceability of Product Lifecycle
Course id:	IMDR56	
Number of ECTS:	14	
Teachers:		Herakovič S. Niko, Lazarević M. Milovan, Šormaz N. Dušan, Ćosić P. Ilija
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of the course is obtaining current knowledge and understanding of contemporary approach in the field of traceability and introduction into research issues in this area.

2. Educational outcomes (acquired knowledge):

Outcomes of the course are the knowledge that enables systematic traceability of different types of products and skills for independent and group research and advanced research work in this area.

3. Course content/structure:

Traceability of products. Introduction to the problem. Aspects of traceability. Traceability in the food industry. Environmental aspects - recycling. Modeling process and system of traceability. Standards of traceability. Technologies applied in product traceability. Infrastructure for access to information about the product. Tracking products in real time. Research trends in product traceability. Research of social and legal aspects of traceability. Case studies. Experimental studies in the laboratory.

4. Teaching methods:

Lectures: (Mentor with the student selects one or more modules depending on the volume.) Consultation. Lectures are delivered in combination. Theoretical part of the subject-matter is followed by examples due to clarify this part of the curriculum. Lectures and consultations are held regularly. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature				
Ord.	Author	Title	Publisher	Year
1,	Milovan Lazarević	RAZVOJ MODELA ZA UPRAVLjANjE PROIZVODIMA U TOKU ŽIVOTNOG VEKA PRIMENOM RFID TEHNOLOGIJA	Autorski reprint	2009
2,	Stankovski, S., Lazarević, M., Ostojić, G., Ćosić, I., Purić, R.	RFID Technology in Product/Part Tracking During the Whole Life Cycle	Assembly Automation, Elsavier	2009

Table 5.2 Course specification

Course:		Strategic Planning and Designing Procedures and Systems at the End of Product Lifecycle			
Course id:	IMDR57				
Number of ECTS:	14				
Teachers:		Čuš -. Franci, Katalinić -. Branko, Lazarević M. Milovan, Ćosić P. Ilija			
Course status:		Elective			
Number of active teaching classes (weekly)					
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:	
5	0	0	4	0	
Precondition courses		None			

1. Educational goal:

The aim of the course is to enable students to understand the latest approach in the development of the procedures and systems at end of product life cycle and introduction to research issues in this area.

2. Educational outcomes (acquired knowledge):

Outcomes of the course are the knowledge that enables students to understand issues relating to the procedures at the end of product life cycle and engage in research work in this field.

3. Course content/structure:

The concept of sustainable development. Industrial ecology. Ecological design and sustainable development. Dismantling for installation, maintenance and recycling. Design for Sustainability (DFS). Design for Environment (DfE). Design for disassembly (DfD). Design for Recycling (DfR). Problems of dismantling products. Technology of dismantling. Collection of products for disassembly. Trends in technology dismantling. Toxic materials. Logistics systems for recycling. Recycling technologies. National and European environmental legislation.

4. Teaching methods:

Lectures: (Mentor with the student selects one or more modules depending on the volume.) Consultation. Lectures are delivered in combination. Theoretical part of the subject-matter is followed by the examples due to clarify this part of the curriculum. Lectures and consultations are held regularly. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Vukelić Đ., Ostojić G., Stankovski S., Lazarević M., Tadić B., Hodolič J., Simeunović N.	Machining fixture assembly/disassembly in RFID environment (Article in press, Date of acceptance 23. February 2010)	Assembly Automation	2010
2,	Milovan Lazarević	PRILOG RAZVOJU SISTEMA ZA DEMONTAŽU PROIZVODA U SKLADU SA USVOJENOM STRATEGIJOM ZA UPRAVLjANjE PROIZVODIMA NA KRAJU ŽIVOTNOG VEKA	Autorski reprint	2006
3,	A.J.D. (Fred) Lambert Surendra M. Gupta	Disassembly modeling for Assembly, Maintenance and Recycling	The St. Lucie Press Series on Recource Management	2005
4,	Ian M. Langella	Planning Demand - Driven Dissassembly for Remanufacturing	Gabler edition wissenschaft	2007

Table 5.2 Course specification

Course:		Project Approach in Effective Systems
Course id:	IMDR59	
Number of ECTS:	14	
Teacher:		Palčič -. Iztok
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Acquiring knowledge about (1) project approach and effective systems, (2) theoretical bases in the area of leadership and project management, (3) current state in the subject area of project management, (4) current research and directions of further development and (5) intelligent systems supported through the establishment of Project Management. The aim is to assess the difference between leadership and management in the subject area, as well as understanding the concept of project readiness.

2. Educational outcomes (acquired knowledge):

Based on the philosophy of the project approach students will understand and be capable to improve working processes in the area of the project activity. According to the theoretical basis and situation in the domain of supremacy students will define the research area and contribute the selected trends in the scientific field of interest as well as advance the development of knowledge on managing projects in an unstable situation. Participants will be prepared to influence the development of project approach to effective systems.

3. Course content/structure:

Links between project management leadership (PML) and scientific approaches in the effective system (ES). Philosophy of the project and ES. Theoretical support in PML. Projects as a heritage of mankind. Philosophy of the organization, strategy and project success. Links between project success and project readiness of ES. The organizational aspects of project management. Scientific approach in PML discipline. The situation in the subject area according to the relevant scientific and professional sources. Leading researchers and their work. Areas explored since the PML became a scientific discipline. Approaches in the development and implementation of projects based on scientific knowledge. Current research in the field of PML (2000 - 2010). The project strategy. Dimensions of project success. Comparison between traditional and modern approaches. The importance of interest groups. Office, as a central unit, for project management. Access to "thinking beyond the boundaries of traditional values in project management". Ethics in PML. Beyond the boundaries of traditional PML. The project approach in manufacturing and product life cycle. Intelligent ES and project approach.

4. Teaching methods:

Lectures. Consultations. Seminar paper. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Poli, M.	Project Strategy: The Path to Achieving Competitive Advantage/Value	Stevens Institut of Technology	2006
2,	Maksimović, R., Lalić, B.	Flexibility and Complexity of Effective Enterprises	Journal of Mechanical Engineering, University of Ljubljana	2008
3,	Poli, M., Mithiborwala, .S., Maksimovic, R., Lalic, B.	PROJECT STRATEGY: SELECTING THE BEST PROJECT STRUCTURE,	PICMET; Portland	2009
4,	Turner, R.	The Handbook of Project-Based Management: Leading Strategic Change in Organizations(3rd Edition)	Nalco System	2008
5,	Kerzner, H.	Advanced Project Management: Best Practices on Implementation	Wiley, Hoboken, NJ.	2004
6,	PMI.Preveli Lalić, B., Marjanović, U.	Vodič kroz korpus znanja za upravljanje projektima	Fakultet tehničkih nauka, Novi Sad	2010

Table 5.2 Course specification

Course:		Enterprise Complexity and Flexibility
Course id:	IMDR60	
Number of ECTS:	14	
Teacher:		Maksimović M. Rado
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Acquiring the latest knowledge on the most important characteristics of company structure and their interconnectedness, as well as their impact on the other characteristics. Mastering techniques for the development of enterprise structure of low complexity and high flexibility.

2. Educational outcomes (acquired knowledge):

Student`s basic knowledge and skills for independent and group research and research in the field of organizational structure. Understanding crucial points of the relationship between the elements of company structure. Acquiring skills for project management of company construction or rehabilitation.

3. Course content/structure:

The complexity of manufacturing / service, organizational and management structure of companies, flexibility of production / service, organizational and management structure of companies. Interrelation of characteristics of complexity and flexibility of company structure. Company structure design with the most favorable ratio of complexity and flexibility. Case studies.

4. Teaching methods:

Lectures. Consultations. Seminar paper. Theoretical part of subject-matter is followed by the examples due to clarify this part of the curriculum. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Maksimović, R.	Složenost i fleksibilnost struktura industrijskih sistema	Fakultet tehničkih nauka u Novom Sadu	2003
2,	Maksimović, R., Stankovski, S., Ostojić, G., Petrović, S., Ratković, Ž.	Complexity and Flexibility of Production Structures	Journal of Scientific and Industrial Research (JSIR), Scientific Publishers	2010
3,	Maksimović, R., Lalić, B.	Flexibility and Complexity of Effective Enterprises	Strojniski vestnik -Journal of Mechanical Engineering, University of Ljubljana	2008
4,	Maksimović, R.	Relationship between Complexity and Flexibility of Production Structures	Strojarstvo, Croation Union of Mechanical Engineers and Naval Architects	2010

Table 5.2 Course specification

Course:		Enterprise Innovative Business
Course id:	IMDR61	
Number of ECTS:	14	
Teacher:		Borocki V. Jelena
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Acquiring the latest knowledge about the requirements to create an innovative company and possible differences between production and service companies. Identifying the impact of dynamic business environment to create innovative corporate strategy.

2. Educational outcomes (acquired knowledge):

Learning outcomes are required knowledge and skills of the students for independent and group research and further research work in this area.

3. Course content/structure:

Innovation - the basic concepts, strategies, innovations, innovative activities. Innovative organization - creating the basic preconditions, characteristics of innovative enterprise - management style, organizational structure, process innovation, staff training, creating a climate for encouraging innovative activities, characteristics of the basic models of measuring innovation enterprise, innovation in the region, differences in innovative activities of manufacturing and service companies.

4. Teaching methods:

Lectures. Consultation. Seminar paper. Examples from practice. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. Theoretical part of the subject-matter is followed by the examples due to clarify this part of the curriculum. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Gupta Praveen	Business Innovation In the 21st Century – A Comprehensive Approach to Institutionalize Business Innovation	Accelper Consulting, USA	2007
2,	Bojović V, Šenk V,Rašković V, StančuMirosavljev M, Borocki J,Radovanović J.	Vodič za inovativne preduzetnike	U sklopu projekta PromotingEntrepreneurial Thinking in theHigh-tech Area,EU	2007
3,	J. Tidd, J.Bessant, K.Pavitt	MANAGING INNOVATION –Integrating technological, market and organizational change	John Wiley and Sons	2008
4,	Borocki. J., Maksimović, R.	Determination of Possible Differences in Applying the Strategic Planning Model between Manufacturing and Service Companies	acta Univerisitates: Mechanical Engineering	2009
5,	Borocki, J., Ćosić, I., Lalić, B., Maksimović, R.	Analysis of company development factors in manufacturing and service company: a strategic approach	Strojniski vestnik -Journal of Mechanical Engineering	2010

Table 5.2 Course specification

Course:		Enterprise Business Process Integration
Course id:	IMDR62	
Number of ECTS:	14	
Teachers:		Šormaz N. Dušan, Tešić M. Zdravko
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of this course is acquiring knowledge on different approaches to the integration of business functions in manufacturing and service companies. Mastering the procedures, methods and techniques of business process integration with the aim of business system management - business.

2. Educational outcomes (acquired knowledge):

Gaining knowledge that will enable students to observe the company as a system of integrated business processes. Understanding the core functions of integration and needs of the company. Acquiring knowledge of automated systems to manage business and production processes in business system - company.

3. Course content/structure:

3. Course content/structure:
Enterprise organization and management in the terms of integrated business processes within the company. IIS access to the integration of company`s functions. ERP concept of Integrated Business Management. LEAN concept of company integration. Business Process Management - BPM approach to business process integration. Case studies (SAP, ORACLE, BAAN).

4. Teaching methods:

Compulsory number of lectures with examples from the above approach. Consultations are held regularly during and after the lectures. Seminar paper written on the basis of required reading and at least three papers published in journals from the SCI list. Through the study research work, seminar paper, scientific journals and other literature student deepens the knowledge acquired at the lectures. Application of obtained knowledge on writing scientific paper.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Tešić, Z., Mitrović, V., Ćosić,I., Lalić,D.	Integration of information for manufacturing shop control	Strojniski vestnik= Journal of Mechanical Engineering	2010
2,	Laudon, K., Laudon, J.	Essentials of Management Information Systems	Pearson Education-Prentice Hall	2010
3,	Bell, S.	Lean enterprise systems	Wiley-Interscience	2005
4,	Dickersbach,J., Keller, G., Weihrauch, K	Production Planning and Control with SAP	Gallileo Press	2007
5,	Ćosić, I., i dr.	Analysis of company development factors in manufacturing and service company</eng	Strojniski vestnik= Journal of Mechanical Engineering</eng	2010
6,	Vom Brocke, J., Rosemann, M.</eng	Handbook of Business Process Management</eng	Springer</eng	2010

Table 5.2 Course specification

Course:		Intelligent Organisation
Course id:	IMDR63	
Number of ECTS:	14	
Teachers:		Tešić M. Zdravko, Maksimović M. Rado, Marić B. Branislav
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Acquiring the latest knowledge about the performance of enterprise, its processes and organizational units, and on key indicators of company prerformance. Mastering the methods of organization and management as well as techniques and methodology of balanced management of company performance.

2. Educational outcomes (acquired knowledge):

Studentćs necessary knowledge and skills for independent and group research and further research work in the field of procedures of company organization and management. Understanding the fundamentals of the analysis of corporate performance. Acquiring the ability to run a business.

3. Course content/structure:

Organization and management of modern enterprise. Virtual Enterprise. Performance of company management. Key performance indicators (KPIs), Balanced Scorecards and other methods of balanced performance of company management. Case studies.

4. Teaching methods:

Lectures. Consultations. Seminar paper. Theoretical part of subject-matter is followed by the examples due to clarify this part of the curriculum. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Schwaninger, M.	Intelligent organizations - Powerful Models for Systemic Management	Springer	2006
2,	Thannhuber, M.J.	The Intelligent Enterprise	Springer- Physica-Verlag Heildelberg	2005
3,	Kaplan, R.S., Norton, D.P.P.	The Strategy-Focused Organization: How Balanced Scorecard Companies Thrive in the New Business Environment	Harvard Business School Press, Boston, Massachusetts	2001
4,	Kaplan, R.S., Norton, D.P.P.	The Balanced Scorecard – Measures that drive performance	Harvard Business Review – HBR	1999
5,	Đurić, Ž. , Maksimović, R., Adamović, Ž.	Key performance indicators in a Joint-Stock Company	African Journal of Business Management, Academic Journals	2010

Table 5.2 Course specification

Course:		Entrepreneurship and Organizational Development
Course id:	IMDR65	
Number of ECTS:	14	
Teachers:		Maksimović M. Rado, Borocki V. Jelena
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Acquiring the latest knowledge on key principles and the principles of entrepreneurship in innovative economy and the main characteristics of organizational development. Knowledge of the latest trends and key changes and concepts of organizational development and creating strategic plan of company development.

2. Educational outcomes (acquired knowledge):

Necessary knowledge and skills of the students for independent and group research and further research work in this area. Acquiring the ability to work independently in the company and / or institutions to support innovative enterprises, understand the core technology development, types and importance of certain institutions to support high-tech entrepreneurship.

3. Course content/structure:

Basic concepts and trends in modern business - impact of changes; role of corporate entrepreneurship to achieve higher levels of innovative activity in the enterprise; impact on the level of enterprise, characteristics of an innovative economy, strategic planning and entrepreneurship; stages of organizational development, creation of strategic plan of enterprise development and its application in unstable conditions. Characteristics of high-tech entrepreneurship, "technopreneurship. Problems in organizational development and ways of their solving - organizational development pyramid.

4. Teaching methods:

Lectures. Consultation. Seminar paper. Examples from practice. Through the study research work, scientific journals and other literature student independently broadens the knowledge presented at lectures. Theoretical part of the subject-matter is followed by the examples due to clarify this part of the curriculum. In addition to working with the teacher, students are trained to write their own scientific and research articles.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Davenport, T.H.	Strategic Management in the Innovation Economy – Strategic Approaches and Tools for Dynamic Innovation Capabilities	Publicis Corporate and Wiley-VCH Verlag GmbH&Co. KGaA, Germany	2006
2,	John S.Oakland	Total organizational excellence – Achieving world-class performance	Butterworth-Heinemann, Linnacre House, Oxford	2001
3,	John Bessant, Joseph Tiddl	Innovation and entrepreneurship	John Wiley and Sons	2007
4,	Stepehen P.Robbins	Organization theory - structure, design and applications	Prentice-Hall International, Inc.	1987
5,	Đaković, V., Anđelić, G., Borocki, J.	Performance of extreme value theory in emerging markets: an empirical treatment	African Journal of Business Management	2010
6,	Maksimović, R., Lalić, B.	Flexibility and Complexity of Effective Enterprises	Strojniški vestnik - Journal of Mechanical Engineering	2008

Table 5.2 Course specification

Course:		Managerial decision-making
Course id:	IMDR66	
Number of ECTS:	14	
Teacher:		Mitrović M. Slavica
Course status:		Elective

Number of active teaching classes (weekly)

Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The following are the objectives of the course Managerial decision-making: 1) to master the basic knowledge in the area of ??managerial decision-making in the industrial system; 2) to introduce students to methods and techniques of making managerial business decisions; 3) to train students in applying these tools and techniques; and 4) to introduce students with the rules of decision-making, factors of influence, as well as properties of decision makers. The purpose of the course is to provide students of management with competencies that will enable them to apply the basic principles and approaches for making functional managerial decisions in industrial systems.

2. Educational outcomes (acquired knowledge):

Students attending the subject of Managerial decision-making and passing the exam are enabled for the following: 1) applying the principles and approaches towards making functional decisions; and 2) using decision-making software for making the business system more successful. The student of management acquires competence in applying the principles and use of software for decision making, as the basis for improving the quality of industrial systems.

3. Course content/structure:

Introduction to managerial decision making; The process of strategic decision-making (good and bad decisions, Types of decision); Factors and stages of decision-making (constraints, environments, methods of decision-making); The context and framework of strategic decision-making, methods of growth in new markets; Personal factors of decision-making (knowledge, skills, and personality traits); Managerial / entrepreneurial decision-making (management style / decision-making style, responsibility and authority); Implementation of business decisions (resources required for the implementation of decisions, their monitoring and evaluation); Models of strategic managerial decision-making (Functional decision making); Decision-making software in business systems: Doctus, Excel Solver; Methods and techniques of strategic decision-making: Structured conflict, Delphi technique, electronic brainstorming, nominal group technique.

4. Teaching methods:

Lectures are presented with the aid of computers, consultation, and seminar papers - presentations.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Slavica Mitrović	Menadžersko donošenje odluka - autorizovana predavanja	Fakultet tehničkih nauka	2012
2,	George Wright	Strategic Decision making	John Wiley&Sons	2001
3,	Bhushan, Navneet, Rai, Kanwal	Strategic Decision making	Springer	2004
4,	Bazerman, M.H.	Judgment in managerial decision making	John Wiley & Sons	2002
5,	Slavica Mitrovic et al.	EMPLOYEE TIME MANAGEMENT: A CASE STUDY FROM SERBIA	Metalurgia International	2013

Table 5.2 Course specification

Course:		Selected Chapters in Product Lifecycle Management
Course id:	IMDR67	
Number of ECTS:	14	
Teacher:		Anišić M. Zoran
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal of the course is mastering the skills required to effectively manage product lifecycle in the conditions of constantly changing functional requirements of the market, production system as well as environmental requirements during its service. Learning about the concept and factors of product lifecycle management (PLM) through building basic structures that ensure the effective creation, sharing and storing information about the product as well as its application in modern strategy of product management.

2. Educational outcomes (acquired knowledge):

Outcomes of the course are acquired knowledge about the structure of the product and family architecture of related products. The acquired engineering knowledge related to each of the phases of the lifecycle under the integrated software for monitoring and management.

3. Course content/structure:

The principles of integrated product development process. Product lifecycle, planning and management. Definition of the product. Specifications and market position of products. Structural Scheme for product and communication between the parts, components and product systems. Presentation and management of product family and product line. Functional requirements and decomposition through the application of Mass Customization and Open Innovation strategies. Product configurator.

4. Teaching methods:

Lectures. Consultations. Seminar paper.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	30.00
			Oral part of the exam	Yes	20.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Anišić, Z., Krsmanović, C.	Assembly Initiated Production as a Prerequisite for Mass Customization and Effective Manufacturing,	Strojniški vestnik - Journal of Mechanical Engineering.	2008
2,	Gecevska, V., Chiabert, P, Anisic, Z., Lombardi, F.	Product lifecycle management through innovative and competitive business environment	JIEM, 2010 –3(2): 323-336 –Online	2010
3,	Saaksvuori A., Immonen A.	Product Lifeycle Management	Springer-Verlag	2008
4,	Stark, J.	Product Lifecycle Management: 21st century Paradigm for Product Realisation.	Springer-Verlag	2004
5,	Grieves, M.	Product Lifecycle Management: Driving the Next Generation of Lean Thinking.	McGraw-Hill.	2005

Table 5.2 Course specification

Course:		Business Communication in Effective Systems
Course id:	IMDR68	
Number of ECTS:	14	
Teacher:		Lalić S. Danijela
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal is that students who study effective communication systems become familiar with main issues of business communication and thus, through the combination of theoretical background and current research work obtain basic knowledge for further research in the subject field and to link the current situation with the situation in real effective business systems.

2. Educational outcomes (acquired knowledge):

Students will master the issues of effective business communication, as well as relevant sources of the latest achievements in this area and thus, be prepared to approach research problems related to business communication in unstable conditions.

3. Course content/structure:

1. Introduction: Effective systems - effective (internal and external) communication 10%
2. Theoretical background 30%
3. Current situation in the field of research 10%
4. Current research, "open" questions and examples of good practice 30%
5. Presentation of independent research work 20%

4. Teaching methods:

Auditory and research work (with emphasis on research techniques on the Internet)

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Exercise attendance	Yes	5.00	Oral part of the exam	Yes	70.00
Lecture attendance	Yes	5.00			
Term paper	Yes	20.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	John V. Thill & Courtland L. Bovee	Excellence in Business Communication	Prentice Hall	2011
2,	Courtland L. Bovee & John V. Thill	Business Communication Today	Prentice Hall	2010
3,	Deborah Roebuck	Improving Business Communication Skills	Prentice Hall	2006
4,	Thomas Cheesebro, Linda O Connor & Francisco Rios	Communication Skills Preparing for Career Success	Prentice Hall	2007
5,	-	Journal of Business Communication	Pretraživo na Kobson servisu - poslednjih 10 godina	2011
6,	-	Business Communication Quarterly	Pretraživo na Kobson servisu - poslednjih 10 godina	2011
7,	-	Business Communications Review	Pretraživo na Kobson servisu- poslednjih 10 godina	2011
8,	-	Journal of Business Communication	Pretraživo na Kobson servisu - poslednjih 10 godina	-
9,	Kolarić, B.,Grubić-Nešić,L.,Radojčić,S.	The challenges of the customer services for modern market requests: a case study of Telecom Serbia	African Journal of Business Management	2010

Table 5.2 Course specification

Course:		Selected chapters from energy efficiency of compressed air systems
Course id:	IMDR86	
Number of ECTS:	14	
Teachers:		Šešlija D. Dragan, Dudić P. Slobodan
Course status:		Elective

Number of active teaching classes (weekly)

Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The educational goal is to deepen the knowledge of doctoral students in the field of energy efficiency of compressed air automated systems and, in that sense, to became familiar with advanced pneumatic control systems, which is used in modern compressed air systems.

2. Educational outcomes (acquired knowledge):

Learning outcomes are the knowledge and skills of students in independent and team scientific and research work in the field of energy efficiency of compressed air.

3. Course content/structure:

Pneumatic control systems with the end position control, pneumatic control systems with stops between the final position, modeling of components (compressed air cylinders, control valves, ...), simulation models of pneumatic components, the application and effectiveness of different control techniques (P, I , D, PI, PID) on energy efficiency, Fuzzy regulation and energy efficiency of compressed air systems, Sliding mode and the energy efficiency of compressed air systems, Servopneumatic control and energy efficiency of compressed air systems, application of PWM control to increase the energy efficiency of compressed air systems, application of PCM control to increase the energy efficiency of compressed air systems, application of PNM control to increase the energy efficiency of compressed air systems, Influence of compressed air quality on energy efficiency, Non-conventional pneumatic actuators influence on energy efficiency, Pneumatic systems with closed circuits, Energy efficiency of complex (with pneumatic and / or hydraulic components) robotic cells.

4. Teaching methods:

Lectures are conducted in a combined way. Leacturing of the theoretical part is followed by examples which serve to clarify the theoretical part of the curriculum. In addition to the lectures, consultations are held regularly. Through study research student, studying scientific journals and other literature and conducting experiments, independently deepens the subject. In addition to working with the student teacher is trained to write his own scientific work in the chosen field.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Dudić, S., Ignjatović, I., Šešlija, D., Blagojević, V., Stojiljković, M,	Leakage quantification of compressed air using ultrasound and infrared thermography	Measurement	2012
2,	Ignjatović, I., Šešlija, D., Tarjan, L., Dudić S,	Wireless sensor system for monitoring of compressed air filters	Journal of Scientific and Industrial Research	2012
3,	Blagojević V, Šešlija D, Stojiljković M	Cost effectiveness of restoring energy in execution part of pneumatic system	Journal of Scientific and Industrial Research	2011
4,	Čajetinac, S., Šešlija, D., Aleksandrov, S., Todorović, M.	PLC Controller used for PWM Control and for Identification of Frequency Characteristics of a Pneumatic Actuator	Elektronika Ir Elektrotechnika	2012
5,	Ignjatović, I., Komenda, T., Šešlija, D., Mališa, V.	Optimisation of compressed air and electricity consumption in a complex robotic cell	Robotics and Computer-integrated Manufacturing	2012

Table 5.2 Course specification

Course:		Financial engineering of public sector
Course id:	IMDR87	
Number of ECTS:	14	
Teachers:		Dobromirov P. Dušan, Radišić M. Mladen
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Enabling students to adopt the principles of functioning of public institutions and to identify the most important trends of the application of modern engineering tools in the public sector, with the introduction to the key factors that determine national fiscal structure. The most important educational goals are to understand the basic concepts of defining the mobilization and disbursement of public resources; the acquisition of knowledge in the area of ??the company's relations to the public sector and benefits from the public sector; acquiring knowledge about the application of modern engineering tools in the public sector.

2. Educational outcomes (acquired knowledge):

Students will gain knowledge about the role and importance of public sector functioning for enterprises and industrial systems and understand the methods of analysis and decision-making in the public sector, as well as public sector regulation forms. The knowledge gained helps students understand the basic concepts of public sector management and participate in defining the relationship of the companies to the public sector.

3. Course content/structure:

The role and importance of the public sector. Public sector organization models. Public sector management. Defining the concepts of mobilization and disbursement of public resources. Relationships between the various government levels. The application of modern engineering tools in the public sector.

4. Teaching methods:

Lectures. Consultations. Essay.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Lecture attendance	Yes	10.00	Oral part of the exam	Yes	50.00
Term paper	Yes	40.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Rosen, S.H., Gayer, T.	Public Finance	McGraw-Hill /Irwin, New York	2007
2,	Radišić, M., Nedeljković, A.	5C Model - Business Case Study Solving Methodology	The New Educational Review (ISSN: 1732-6729)	2012
3,	Hughes, O. E.	Public management and administration: An introduction	Palgrave, New York	2003

Table 5.2 Course specification

Course:		Planning and implementing cost structure of the investment cycle
Course id:	IMDR88	
Number of ECTS:	14	
Teachers:		Ivanišević V. Andrea, Marić B. Branislav
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Teaching enables students to master the complete process of planning and implementing cost structure of the investment cycle. The most important educational objectives relating to the preparation of students and adapting to new trends in the management of the investment cycle cost structure (and implementation plan) that include development of various projects of this type.

2. Educational outcomes (acquired knowledge):

Students will gain knowledge in the planning and implementation of cost structure of the investment cycle and learn about the latest trends.

3. Course content/structure:

Analysis of the necessary conditions for investment. The structure of the investment (a new building or machinery, technology, extension, expansion, investment and maintenance). Analysis of the market-defining programs. Return on investment budget and budget impact to the business (loans, profits, increased employment and the like.). Source documentation for the decision (investment initiatives, programs, business plan), sources of funding (own funds, loans, banks, funds). Planning investment flow. Documentation and approval (approval) for investment. Legislation and EU regulations. The realization of investment, contracting, performance, download. Activate investment.

4. Teaching methods:

Lectures. Consultation. Essay.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Branislav Marić, Andrea Ivanišević	Planiranje i realizacija troškovne strukture investicionog ciklusa (elektronska skripta)	Fakultet tehničkih nauka Novi sad	2012
2,	Branislav Marić	Upravljanje investicijama	Fakultet za preduzetni menadžment	2006
3,	Božidar Leković, Andrea Ivanišević, Branislav Marić, Jelena Demko-Rihter	ASSESSMENT OF THE MOST SIGNIFICANT IMPACTS OF ENVIRONMENT ON THE CHANGES IN COMPANY COST STRUCTURE	Ekonomska istraživanja-Economic Research ISSN 1331-677X	2012

Table 5.2 Course specification

Course:		Controlling and Internal Audit in Corporate Governance.
Course id:	IMDR89	
Number of ECTS:	14	
Teachers:		Perović I. Veselin, Nerandžić B. Branislav
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of the course is to teach students about future PhD engineers with modern corporate governance instruments and instruments of implementation of modern management model of corporate governance. The basic knowledge and understanding of internal auditing and controlling, mainly audits of corporations in order to achieve the strategic goals of the organization and the industrial system and reduce business risks to achieve it.

2. Educational outcomes (acquired knowledge):

Acquiring knowledge and skills necessary for the implementation of standards, procedures and internal control models of companies and other organizations. Knowledge of the practical application of tools and techniques of controlling business analysis reports in a company or other organization. Students will be able to: identify strategic components of control and internal audit, using models and tools for the analysis of controlling in businesses and organizations, draw conclusions, propose and compare different strategies that shape the reports and recommendations to the shareholders and management of the company, to participate in the application strategy to position the company with highly skilled engineers who are located at the position of analysts by enhancing performance measurement systems controlling companies and organizations involved in scientific research teams of enterprise management tools.

3. Course content/structure:

Strategic and operational Controlling Instruments. Controlling the preparation of business reports; Preparations for the analysis of business reports, check the operation of the information system of internal control, audit of financial statements; Wider rating company's solvency, financial and non-financial ratios, grade of integration of business processes; Problems surface for analysis, assessment of business model rizikas , methods of analysis, internal audit, and international standards, planning and stages of internal and operational auditing, internal audit role in the creation of the management of business risks.

4. Teaching methods:

Lectures using audiovisual equipment. Consultation. Lectures by experienced executives in part or whole enterprises function in the role of guest lecturers.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Nerandžić B.	Interna i operativna revizija	Stylos, Novi Sad	2007
2,	Perović V	Kontroling	Rodacomm, Novi Sad	2007
3,	Perović V., Nerandžić B., Bojanić R.	INFLUENCE OF CONTROLLING THE INVESTMENT PROJECTS IN ERP(M) WITHPRIMARY FOCUS ON THE CASH FLOW IN THE COMPANY	Metalurgia International	2012

Table 5.2 Course specification

Course:		Selected Chapters of Strategic Managment Accounting
Course id:	IMDR90	
Number of ECTS:	14	
Teachers:		Nerandžić B. Branislav, Perović I. Veselin
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The aim of this course implies completion and integration of component of strategic thinking necessary to PhD candidates - engineers who takes positions within the functions of finance, accounting, planning, control and financial reporting. Acquiring basic knowledge and understanding of accounting information as a management tool, in order to achieve the strategic goal of the industrial system and organization as well as reducing business risks in achieving strategic goal.

2. Educational outcomes (acquired knowledge):

Acquiring knowledge and skills needed to implement standards, procedures and models for the evaluation of company's solvency in order to integrate successfully all functions in the company. Knowledge needed for the practical application of tools and techniques for strategic management accounting in the company and other organizations. Students are qualified to identify strategically components of information base of accounting system, use models and tools in order to analyze competition in the company and other organizations, make conclusions, propose and compare different strategies, create reports and recommendations for capital owners and management, participate in the application of strategy as an engineer who holds PhD and takes positions of scientific worker or analyst through the improvement of system of performance measurement of the company and other organizations.

3. Course content/structure:

The structure of strategic management accounting. Making of financial and business reports; Analysis of business reports of competitors from the point of the achievement of the strategic objectives of the company or corporation. Monitoring the operation of information systems and system of internal controls of the organization; Audit of financial statements; Evaluation of the company's solvency; Assessment of the integration of business processes; Analysis of the financial result in terms of strategic corporate objectives; Analysis of financial situation; Analysis of the production factors; Analysis of the production operations; Analysis of the consolidated business reports of multinational corporations; Transfer pricing in multidivisional business systems.

4. Teaching methods:

Lecturers use audiovisual equipment. Case studies. Consultation. Occasionally lecturers are experienced executives from the companies in the role of guest lecturers

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Nerandžić B.,Perović V.	Upravljačko računovodstvo	FTN, Novi Sad	2009
2,	Perović V.,Nerandžić B	Finansijsko poslovanje	FTN, Novi Sad	2010
3,	Milićević V.	Strategijsko upravljačko računovodstvo	EF, Beograd	2003

Table 5.2 Course specification

Course:		Product Family Development and Product Configurators
Course id:	IMDR91	
Number of ECTS:	14	
Teacher:		Anišić M. Zoran
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Mastering the development of modular product family architecture suitable for creation of a large number of product variants, according to individual customer requirements. Also, product configurator design able to connect customer requirements with functional characteristics of the product.

2. Educational outcomes (acquired knowledge):

After completing and passing the exam, students are able to design the architecture of the product and the production program that is suitable for the configuration to suit individual customer requirements. Students are also able to design the structure of the product configurator according to the required depth and width of customization.

3. Course content/structure:

Basics of Mass Customization strategy. Product family structure and development of structural scheme for a production programme. Linking consumer demands, attributes and their values ??to functional characteristics of a product. Determination of width and depth of customization. Different types configurators. The structure and design of a product configurator. Various case studies of commercial configurators.

4. Teaching methods:

Lectures are auditory, followed by appropriate slides, while excercises are performing in small groups mostly auditory, but partly in a computer laboratory.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	30.00
			Oral part of the exam	Yes	20.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Hvam, L., Mortensen, N.H., Riis, J.	Product customization	Springer	2008
2,	Anišić, Z.	Razvoj i menadžment proizvoda u toku životnog ciklusa	FTN	2011
3,	Piller, F., Tseng, M.	Handbook of Research in Mass Customization and Personalization	World Scientific Publishing Company	2009
4,	Simpson, T., Siddique, Z., Jiao, R.	Product Platform and Product Family Design: Methods and Applications	Springer	2005

Table 5.2 Course specification

Course:		Advanced Forecasting Methods and Techniques
Course id:	IMDR92	
Number of ECTS:	14	
Teacher:		Anišić M. Zoran
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The goal of the course is mastering the advanced techniques of exploratory and normative techniques in the function of technological and business forecasting.

2. Educational outcomes (acquired knowledge):

After completing the course and passing the exam, the student is able to use the advanced techniques of business and technology forecasting, which can be quantitative or qualitative. Students will be able to fully implement forecasting in real production and service systems.

3. Course content/structure:

Parameter selection and data collection. Data processing. The choice of forecasting methods. Exploratory methods: time series analysis, writing, morphological analysis. Normative methods: PATTERN methods and significance tree. Principles of prediction. Interpretation of forecasting results.

4. Teaching methods:

The lectures are auditory providing necessary theoretical background, while the exercises include examples of work assignments which are related to practical problems in the design of future technological trends.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project task	Yes	50.00	Theoretical part of the exam	Yes	30.00
			Oral part of the exam	Yes	20.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Anišić, Z.	Tehnološko i poslovno predviđanje	FTN - skripta	2012
2,	Armstrong, J.	Principles of Forecasting: A Handbook for Reasearchers and Practitioners	Norwell, Kanada	2001
3,	Martino J. P.	Technological Forecasting for Decision Making	McGraw-Hill	1993

Table 5.2 Course specification

Course:		Virtual Enterprises and Collaborative Systems
Course id:	IMDR93	
Number of ECTS:	14	
Teachers:		Herakovič S. Niko, Lazarević M. Milovan, Šormaz N. Dušan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The course teaches the students the basic knowledge necessary to understand and analyze the latest approach in the development of virtual enterprises and collaborative systems and their organization and management. In addition, students will learn about the latest research in this area and will be trained for independent research in the subject area.

2. Educational outcomes (acquired knowledge):

After completing the course and passing the exam, students will be able to independently design and analyze virtual enterprise, as well as conducting research using valid scientific methods, with the aim of improving and optimizing existing virtual enterprises.

3. Course content/structure:

Introduction to the concept of virtual enterprise. Virtual environment. Agile virtual enterprise. The integration of virtual enterprises. Knowledge management in virtual enterprises. The concept of digital factory and digital production. Distributed production in virtual enterprises. The formation of virtual enterprise. Cooperative work. Distributed manufacturing companies and engineering teams. Production planning. Scheduling. Automation and Control. Simulations. Digital engineering. Collaborative design at an early stage. Work processes and communication systems. The human and social aspects of virtual enterprise.

4. Teaching methods:

Lectures: (Mentor with the student chooses one or more modules, depending on the scope of the module). Consultation. Lectures are conducted in combination. Leaving the theoretical part, followed by examples which serve to clarify the theoretical part of the curriculum. In addition to lectures, are held regularly consultation. Through study research student, studying scientific journals and other literature deepens own curriculum with lectures. In addition to working with the teachers, the students are trained to write your own scientific work.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Lazarević M., Ostojić G., Ćosić I., Stankovski S., Vukelić Đ., Zečević I.	Product lifecycle management (PLM) methodology for product tracking based on radio-frequency identification (RFID) technology	Scientific Research and Essays	2011
2,	L. M. Camarinha-Matos , H. Afsarmanesh, H.H. Erbe	Advances in Networked Enterprises: Virtual Organisations, Balanced Automation, and Systems Integration	Springer	2010
3,	Wang L., Nee Y.C.A.	Collaborative Design and Planning for Digital Manufacturing	Springer-Verlag London Ltd.	2009
4,	Koç M., Ni J., Lee J.	Introduction to e-Manufacturing, Information Technology Handbook	CRC Press	2005

Table 5.2 Course specification

Course:		Trends in the environmental management systems
Course id:	IMDR94	
Number of ECTS:	14	
Teachers:		Beker A. Ivan, Jocanović T. Mitar, Kamberović L. Bato, Milisavljević M. Stevan, Radlovački S. Vladan, Šević D. Dragoljub
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The course introduces students to research in the field of environmental management system in terms of relations with the logistics processes, processes related to the quality management system, hydraulic systems and processes related to customer relationship. Students will become familiar with the development of the area in the past two decades, and the latest research and forecasts about developments in the future. The knowledge acquired will enable students a thorough understanding of the field of the system of environmental management, which will form the basis for independent research.

2. Educational outcomes (acquired knowledge):

After completing courses and passing the exam, students will learn existing models of environmental management system from the standpoint of quality management systems, hydraulic systems, processes related to customer relationship, and logistics processes. Students will also learn the ability to create research related to the area and to critically analyze existing processes of environmental management system in the monitored enterprise.

3. Course content/structure:

Course covers the development of the concept of sustainable development, global environmental problems, causes and consequences of environmental degradation, advanced principles of strategy and policy for sustainable development, environmental risk management principles.

4. Teaching methods:

Classes include lectures on the analysis of the examples, and the choice of different strategies and assessing the strategies to protect the environment. Students create project and present it to other students making the group after which a debate is realised. The exam consists of two parts: theory and tasks.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Ken Whitelaw	ISO 14001 Environmental Systems Handbook	Elsiver Butterworth-Heinemann	2012
2,	Grupa autora	SISTEM MENADžMENTA KVALITETOM	Fakultet tehničkih nauka, IIS - Istraživački i tehnološki centar, Novi Sad	2012
3,	Međunarodni standard	SRPS ISO 14000	Institut za standardizaciju Srbije	2005

Table 5.2 Course specification

Course:		Trends in Customer Relationship Management
Course id:	IMDR95	
Number of ECTS:	14	
Teachers:		Beker A. Ivan, Jocanović T. Mitar, Kamberović L. Bato, Milisavljević M. Stevan, Radlovački S. Vladan, Šević D. Dragoljub
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The subject introduces students to research in this area is characterized by an intense and innovative development. Students will become familiar with the development of the area in the past two decades, and the latest research and forecasts about developments in the future. The knowledge acquired will enable students a thorough understanding of the field of customer relationship management, which will form the basis for independent research.

2. Educational outcomes (acquired knowledge):

After completing courses and passing the exam, students will master the existing models of the management relationships with customers that are represented in the world. Understanding the existing models will let you select the correct strategy in forming relationships with customers.

3. Course content/structure:

Organization and CRM strategies, CRM as an integral business strategy, organization-oriented relations, communication channels through higher; Customizing offers individual buyer; Politics relationships with customers, analytical CRM, data analysis and "datamining"; segmentation and selection, "Cross-sell "analysis, the effects of marketing activities, reporting of results, operational CRM

4. Teaching methods:

Lectures, research work, consultations. The rating is based on the success of the project and an oral exam.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Mitrović S., Milisavljević S., Ćosić I., Leković B., Grubić Nešić L., Ivanišević A.	Changes in leadership styles in a transitional economy: A Serbian case study	African Journal of Business Management	2011
2,	Grönroos Christian	Service Management and Marketing: Customer Management in Service Competition	Chichester: Wiley	2007
3,	Hughes. A	How to measure CRM success	Database marketing Institute Ltd	2009

Table 5.2 Course specification

Course:		Project portfolio management
Course id:	IMDR96	
Number of ECTS:	14	
Teacher:		Morača D. Slobodan
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

Modern business characterized by multidisciplinarity, the use of modern technology and the increased intensity of changes, therefore it is necessary to train the participants to manage change through projects and train them for coordinating and directing the project from a variety of functions that businesses and projects are implemented by a number of companies in accordance with the requirements set and defined goals. Sustainability in the global market depends on the success of the company to manage changes through a series of multidisciplinary projects implemented by the internal and external environment so that it can be said that the modern form of management - project management. Process quality, effectiveness and efficiency of operations caused by the successful implementation of several projects using the knowledge, tools and techniques to manage a portfolio of projects to internal and external level. The main educational objective of this course is to enable students to succe

2. Educational outcomes (acquired knowledge):

After completing and passing the course, students are able to: identify the needs of stakeholders (users, corporate, organizations, institutions ...) to launch its portfolio of projects, perform the necessary analysis to determine the feasibility and viability of the projects, accurately define and agree on project ideas, define , plan and integrate the activities of multiple projects, analyzing and allocating the necessary resources in accordance with project priorities, and analyze the costs of the project, in collaboration with other members of the team involved in the implementation and control of the project portfolio, to the final results of the activities of the project to provide use and design documentation and the experience archive.

3. Course content/structure:

The integration process; Joint action of companies; Changes in the company; Managing change and managing a portfolio of projects; Standards and methodologies of project management; Requirements analysis; Methods and techniques; Portfolio preparation and initiating projects; Planning and integration of activities and resources for multiple projects; Project costs; Quality Management; Managing, implementation and coordination; Design documentation, Establishing control system of project portfolio; Project resources and procurement; Closure projects.

4. Teaching methods:

Multimedia lectures and exercises with examples of projects portfolio and explanation of the specific methods and techniques. Lectures in part by experienced project managers in the role of guest lecturers. In the exercises to encourage small group work, students are trained to implement project portfolio management methodology on specific examples of the application of engineering methods and techniques. Exercises are performed by a computer in a laboratory. Part of the teaching will be done in one of the companies.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Presentation	Yes	10.00	Oral part of the exam	Yes	50.00
Project	Yes	40.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Grupa Autora	PMBOK Vodič – četvrto izdanje	FTN	2010
2,	Parviz F. Rad, Ginger Levin	Project Portfolio Management Tools and Techniques	IIL Publishing, New York	2006
3,	Grupa autora	The Standard for Portfolio Management - 2nd Edition	PMI	2008
4,	Moraca S., Hadzistevic M. Drstevnsek I. Radakovic N.	Application of Group Technology in Complex Cluster Type Organizational Systems	Journal of Mechanical Engineering	2010

Table 5.2 Course specification

Course:		Entrepreneurial Management
Course id:	IMDR97	
Number of ECTS:	14	
Teacher:		Mitrović M. Slavica
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses: None

1. Educational goal:

The following are the objectives of the course of Entrepreneurial Management: 1) to acquire the basic knowledge in the field of entrepreneurial management in the modern business environment; 2) to introduce students with the basic determinants and forms of entrepreneurial management; 3) to acquire the basic knowledge and key skills of managing successfully both small and medium sized enterprises, as well as large industrial systems, and 4) to introduce students to management styles in companies.

2. Educational outcomes (acquired knowledge):

Students attending the course and passing the exam are able to: 1) create the preconditions for successful entrepreneurial management in the actual economic environment of both small and large organizations; 2) apply the determinants of entrepreneurial management in organizations; and 3) to implement management styles according to the required level of managerial effectiveness. This course provides students with competencies for managing and improving the organization towards innovating and creating new products and services.

3. Course content/structure:

Introduction to entrepreneurial management; Forms of entrepreneurial management: operative and proactive; Determinants of entrepreneurial management: focus on change, a focus on business opportunities and focus on the organization; Personal factors of entrepreneurs-managers; Managerial/entrepreneurial style of management; Application of styles management; Models and software of managerial/entrepreneurial governance.

4. Teaching methods:

The instruction is provided through lectures, theoretical subject matter with the required number of case studies, as well as through practical exercises aided by computers, consultation, and seminar papers – presentations.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Lecture attendance	Yes	5.00	Oral part of the exam	Yes	50.00
Project	Yes	45.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Slavica Mitrović et.al	Manager's Assessment of Organizational Culture	E+M Ekonomie a Management	2013
2,	Slavica Mitrović	Preduzetnički menadžment - elektronska skripta	Fakultet tehničkih nauka	2013
3,	N. V. R. Naidu, Naidu	Management and Entrepreneurship	International Publishing House Pvt. Ld	2008

Table 5.2 Course specification

Course:		Modern concepts, methods and tools of human resource management
Course id:	IMDR98	
Number of ECTS:	14	
Teacher:		Katić R. Ivana
Course status:		Elective

Number of active teaching classes (weekly)

Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses

1. Educational goal:

Gaining knowledge of modern methods and techniques in the field of human resource management with an emphasis on the practical application of key concepts in business.

2. Educational outcomes (acquired knowledge):

Students will be able to: (1) adopt the current methods and tools in the field of human resource management (2) the application of new skills to master the concepts of human resources, (3) gain knowledge of the business processes of international human resource management (4) analyze and identify professional and organizational profile (5) manage interpersonal relationships using the tools of psychological concepts in a measurable and cost-effective manner (6) identify, analyze and improve business activities based on the knowledge and skills in the field of human resource management.

3. Course content/structure:

Development trends in the management of human resources in the future: challenges for Human Resources in the 21st century, changing nature of work, employment, human resources in the public and private sectors.
International human resource management definitions, settings, international organizational models, cultural differences.
Human resource management in practice: adoption of new skills and tools used in practice, the modern tools of the process of recruiting employees, employee competence (types, frame, for reasons of competence, competency model development), interpersonal relations and intelligence employees (setting, the nature of relationships, trust and organizational relationships)
Psychological contracts: definitions, significance, nature of contract, maintaining a positive agreement.
Organizational Portfolio: diagnosis reference frame of the organization, staff, alignment of organizational and individual needs, organizational development and transformation
Professional identity: the professional game, professional style, talent acquisition, the balance between life and work.

4. Teaching methods:

Teaching includes lectures on the subject, with examples and exercises designed through team discussions, workshops, internet research, case studies.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Exercise attendance	Yes	5.00	Theoretical part of the exam	Yes	50.00
Homework	Yes	20.00			
Lecture attendance	Yes	5.00			
Term paper	Yes	20.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Ronald R. Sims	Human Resource Management:Contemporary Issues, Challenges and Opportunities	Information Age Publishing, United States of America	2007
2,	Price,A.	Human resource management	Cengage Learning,EMEA, UK	2011
3,	Losey,M.Meisinger,S.Ulrich,D	The future of human resource management	John Wiley&Sons, USA	2005
4,	Armstrong,M	Armstrongs handbook of hrm practice	Kogan Page	2012

Table 5.2 Course specification

Course:		Data ACQUISITION, ANALYSIS AND INTERPRETATION 2
Course id:	IMDR99	
Number of ECTS:	14	
Teacher:		Pečujlija D. Mladen
Course status:		Elective

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
5	0	0	4	0

Precondition courses

1. Educational goal:

The subject aims to enable students to understand many basic concepts, processes, and issues that arise when performing empirical studies in most disciplines of management, and thus create a conceptual basis for later studies in facilities that include this type of knowledge.

2. Educational outcomes (acquired knowledge):

Students are trained in-house research design, data collection, data processing, multivariate methods (exploratory factor analysis, EFA, confirmatory factor analysis CFA, structural modeling, SEM, cluster analysis, canonical correlation analysis, the discriminatory analysis, a method of neural networks, data interpretation and preparation report on the research conducted and the use of the software package SPSS, AMOS, LISREL.

3. Course content/structure:

The presentation material is a continuation of the course, its advanced section where students are trained to perform the collection, processing and analysis of data using multivariate procedures that are consistent with the trends of the world's leading journals in the field (in depth). These are the procedures and konfarmitivne exploratory factor analysis, cluster analysis and Structural modeling method. The focus is primarily on logic and above all practice mentioned at the end of the course describes the structure of a standard written report on the investigation. During the course, for illustration shows a large number of (mostly simplified) examples of research in many areas of management.

4. Teaching methods:

Lectures, computer exercises, consultations

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Computer exercise attendance	Yes	5.00	Oral part of the exam	Yes	30.00
Project	Yes	30.00	Practical part of the exam - tasks	Yes	20.00
Project task	Yes	15.00			

Literature

Ord.	Author	Title	Publisher	Year
1,	Maruyama, G.M.	Basics of Structural Equation Modeling	Sage, Thousand Oaks, CA	1998
2,	Nunnally, J.M	Psychometric theory		1998
3,	Cohen, J., Cohen, P., West, S.G. and Aiken, L.S	Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences	Erlbaum, Mahwah, NJ	2003
4,	Mladen Pečujlija, Ilija Ćosić,	A professor's moral thinking at the abstract level vs the professor's moral thinking in real life situation (consistency problem).	Springer	2011

Table 5.2 Course specification

Course:		Selected Chapters in Design for Excellence			
Course id:	IMDRPI				
Number of ECTS:	14				
Teachers:		Anišić M. Zoran, Ćosić P. Ilija			
Course status:		Elective			
Number of active teaching classes (weekly)					
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:	
5	0	0	4	0	
Precondition courses		None			

1. Educational goal:

Acquiring specific knowledge in the field of comparative (simultaneous) engineering.

2. Educational outcomes (acquired knowledge):

Ability to deal with scientific and research work in the field.

3. Course content/structure:

Basic concept and history of DFX, Predecessors of the design for excellence, Abilities for assembly and manufacture, Basic idea and necessity of applying DFX, Diverse DFX approaches, Basic principles for DFX, Organization and Management of DFX approach, Procedure for product development, Comparative or simultaneous engineering (SE), Teamwork and cooperation, Evaluation of proposed solutions for improvement, Dimensions of DFX, Design for Assembly (DFA), Design for Manufacture (DFM), Design for Quality (DFQ), Design for Cost Optimization (DFC), Design for reliability, Design for service and maintenance, Design for safety, Design for environment protection, Design for simple usage, Design for fast market introduction, Computer-aided DFX and the integration with CAD, IIS-DFX developed tools in CAD, Tendencies for future development of the DFX approach.

4. Teaching methods:

Lectures. (Mentor and students select one or more modules depending on the size of the module content.) Consultations. Lectures are organized in combined form. The presentation of the theoretical part is followed by the corresponding examples. In addition to lectures there are regular consultations. In study and research, students investigate through scientific journals and other literature independently to upgrade the lectures. In working with the teacher, student is becoming capable for individual writing of a scientific paper.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Project	Yes	50.00	Theoretical part of the exam	Yes	30.00
			Oral part of the exam	Yes	20.00

Literature

Ord.	Author	Title	Publisher	Year
1,	Zelenović, D. i ostali	Integralni razvoj proizvoda - osnove	FTN - Novi Sad	1998
2,	Huang, G.	Design for "X" - Concurrent Engineering Imperatives	Chapman & Hall	2000
3,	Bralla, J.G.	Design for eXcellence	McGraw-Hill	1996
4,	Andreasen, M., Kahler, S., Lund, T.	Design for Assembly	JFS Public, UK	1999

Table 5.2 Course specification

Course:		Doctoral Dissertation (Theoretical Bases)
Course id:	SID01	
Number of ECTS:	30	
Teachers:		
Course status:	Mandatory	

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
0	0	0	20	0

Precondition courses: None

1. Educational goal:

The application of fundamental, theoretical and methodological, scientific and professional, and professional and applicative knowledge, methods and contemporary knowledge from the magazines from the SCI list in order to solve concrete problems within the courses at Doctoral studies.

2. Educational outcomes (acquired knowledge):

Enabling students to individually connect the contents from the courses at Doctoral studies, apply previously acquired as well as new knowledge for observing the structure of the set problems and its systematic analysis in order to elaborate conclusions on possible directions in its solving. Through individual usage of literature, students broaden their knowledge and utilizing new methods individually and creatively, they use new knowledge in solving the set problems.

3. Course content/structure:

It is formulated individually in accordance with further research. Students read scientific literature, and perform analyses in order to find solutions for a concrete task which is defined by setting the task on the side of the supervisor and other lecturers at Doctoral studies. Theoretical bases present a classification examinations. Students are prepared to take the classification examination.

4. Teaching methods:

Student`s co-supervisor sets the seminar paper task and delivers it to the student. The student has the obligation to elaborate the paper within the set theme defined by the paper task, utilizing the literature proposed by the co-supervisor. During the paper elaboration, the co-supervisor can provide additional instructions to the student direct them to certain literature and additionally direct them towards the elaboration of a quality paper. During the study research work, the student has tutorials with the co-supervisor and course lecturers, and if needed, with other lecturers dealing with the problems in the field of the set paper task. Within the set theme, the student can also perform certain measuring, research, calculations, surveys and other researches, statistic data processing, if it is necessary for the task. After the defence of the paper, the candidate has to pass the oral examination in the field of the passed examinations, in front of a committee. If the examination is

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature				
Ord.	Author	Title	Publisher	Year
1,	grupa autora	časopisi sa liste Kobsona		sve
2,	grupa autora	časopisi i doktorske disertacije iz date problematike		sve

Table 5.2 Course specification

Course:		Doctoral Dissertation – Study and Research
Course id:	SID02	
Number of ECTS:	30	
Teachers:		
Course status:	Mandatory	

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
0	0	0	30	0

Precondition courses: None

1. Educational goal:

The application of fundamental, theoretical and methodological, scientific and professional, and professional and applicative knowledge and methods in solving concrete problems within the selected field. In this segment of Doctoral dissertation, students investigate the problem, its structure and complexity and on the basis of the performed analyses draw conclusions on possible manner in its solving. Researching the literature, students are introduced to methods attended for creative solving of new tasks and the engineering practice in their solving. The objective of students` activity within this segment of research is to acquire necessary experience through solving complex problems and tasks and recognizing the possibility for applying previously acquired knowledge in practice.

2. Educational outcomes (acquired knowledge):

Enabling students to individually apply previously acquired knowledge from diverse areas already studied in order to observe the structure of the set problem and its systematic analysis for drawing conclusions on possible directions in its solving. Through individual usage of literature, students broaden their knowledge from the selected field and they investigate diverse methods and papers related to the similar fields. Thus, students develop the competence to perform analyses and identify problems within the set theme. Practical application of the acquired knowledge from diverse areas develops in students the ability to overview the place and the role of engineers in the selected field, the demand for cooperation with other professions and the team work.

3. Course content/structure:

It is formulated individually in accordance with the elaboration of the concrete Doctoral dissertation, its complexity and structure. Students read scientific literature, Doctoral dissertations by other students dealing with similar theme; they perform analyses in order to find solutions for a concrete task defined by the task of the Doctoral dissertation.

4. Teaching methods:

The supervisor of the Doctoral dissertation sets the dissertation task and delivers it to the student. The student has the obligation to elaborate the dissertation within the set theme defined by the Doctoral dissertation task, utilizing the literature proposed by the supervisor. During the elaboration of the Doctoral dissertation, the supervisor can provide additional instructions to the student direct them to certain literature and additionally direct them towards the elaboration of a quality Doctoral dissertation. During the study research work, the student has tutorials with the supervisor, and if needed, with other lecturers dealing with the problems in the field of the set dissertation task. Within the set theme, the student can also perform certain measuring, research, calculations, surveys and other researches, statistic data processing, if it is predicted by the task of the Doctoral dissertation.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	grupa autora	časopisi sa liste Kobson		sve
2,	grupa autora	časopisi i doktorske disertacije iz date problematike		sve

Table 5.2 Course specification

Course:		Doctoral Dissertation – Study and Research
Course id:	SID03	
Number of ECTS:	10	
Teachers:		
Course status:	Mandatory	

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
0	0	0	10	0

Precondition courses: None

1. Educational goal:

The continuation of study and research from previous semester. The application of fundamental, theoretical and methodological, scientific and professional, and professional and applicative knowledge and methods in solving concrete problems within the selected field. In this segment of Doctoral dissertation, students investigate the problem, its structure and complexity and on the basis of the performed analyses draw conclusions on possible manner in its solving. Researching the literature, students are introduced to methods attended for creative solving of new tasks and the engineering practice in their solving. The objective of students` activity within this segment of research is to acquire necessary experience through solving complex problems and tasks and recognizing the possibility for applying previously acquired knowledge in practice.

2. Educational outcomes (acquired knowledge):

Enabling students to individually apply previously acquired knowledge from diverse areas already studied in order to observe the structure of the set problem and its systematic analysis for drawing conclusions on possible directions in its solving. Through individual usage of literature, students broaden their knowledge from the selected field and they investigate diverse methods and papers related to the similar fields. Thus, students develop the competence to perform analyses and identify problems within the set theme. Practical application of the acquired knowledge from diverse areas develops in students the ability to overview the place and the role of engineers in the selected field, the demand for cooperation with other professions and the team work.

3. Course content/structure:

It is formulated individually in accordance with the elaboration of the concrete Doctoral dissertation, its complexity and structure. Students read scientific literature, Doctoral dissertations by other students dealing with similar theme; they perform analyses in order to find solutions for a concrete task defined by the task of the Doctoral dissertation.

4. Teaching methods:

The supervisor of the Doctoral dissertation sets the dissertation task and delivers it to the student. The student has the obligation to elaborate the dissertation within the set theme defined by the Doctoral dissertation task, utilizing the literature proposed by the supervisor. During the elaboration of the Doctoral dissertation, the supervisor can provide additional instructions to the student direct them to certain literature and additionally direct them towards the elaboration of a quality Doctoral dissertation. During the study research work, the student has tutorials with the supervisor, and if needed, with other lecturers dealing with the problems in the field of the set dissertation task. Within the set theme, the student can also perform certain measuring, research, calculations, surveys and other researches, statistic data processing, if it is predicted by the task of the Doctoral dissertation.

Knowledge evaluation (maximum 100 points)

Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Term paper	Yes	50.00	Oral part of the exam	Yes	50.00

Literature

Ord.	Author	Title	Publisher	Year
1,	grupa autora	časopisi sa liste Kobsona		sve
2,	grupa autora	časopisi i doktorske disertacije iz date problematike		sve

Table 5.2 Course specification

Course:		Doctoral Thesis - Realization and Defence of Thesis
Course id:	DZR03	
Number of ECTS:	20	
Teachers:		
Course status:	Mandatory	

Number of active teaching classes (weekly)				
Lectures:	Practical classes:	Other teaching types:	Study research work:	Other classes:
0	0	0	0	20

Precondition courses: None

1. Educational goal:

Acquiring knowledge about structure and form of writing the dissertation report after analysis, and other activities carried out within the assigned theme of Doctoral dissertation. By writing the Doctoral dissertation, students gain experience in writing papers within which it is necessary to describe the problem, implement methods and procedures and obtained results, as well as to give new scientific contribution to the science development and to the application of the scientific research in practice. In addition, the objective of writing and defense of the Doctoral dissertation is to develop student skills for independent paper preparation in a suitable form for the purpose of public presentation, as well as to respond to comments and questions related to the given topic.

2. Educational outcomes (acquired knowledge):

Training students for a systematic approach in solving the given problems, carrying out analyses, applying knowledge and accepting knowledge from other areas in order to find creative solutions for a given problem. Through independent studying and solving tasks in a given topic, they acquire the knowledge about the complexity of the problems in the field of their profession. Through elaboration of Doctoral dissertation, students gain certain experiences that can be applied in practice when solving problems in the field of their profession. The student acquires necessary experience on how to present the results of independent or team work in practice by preparing the results for public defense, by public defense, and by answering questions and complaints of the Commission.

3. Course content/structure:

It is individually formed in accordance with the needs and the field covered by a given Doctoral dissertation. In agreement with a mentor, a student makes the Doctoral dissertation in a written form in accordance with the rules provided by the Faculty of Technical Sciences. The student prepares and defends the written Doctoral dissertation in public, in agreement with the mentor and in accordance with the prescribed rules and procedures.

4. Teaching methods:

During the elaboration of the Doctoral dissertation, the student consults with his/her mentor, and if necessary with other teachers dealing within a sphere of the Doctoral dissertation. The student writes the Dcotoral dissertation, and submits the bound copies to the Commission upon the approval of the Commission for assessment and defense. The Defense of the Doctoral dissertation is performed in public, and after the presentation, the student is obliged to orally answer the questions and comments.

Knowledge evaluation (maximum 100 points)					
Pre-examination obligations	Mandatory	Points	Final exam	Mandatory	Points
Writing the PhD thesis	Yes	50.00	PhD thesis defence	Yes	50.00

Standard 06. Programme Quality, Contemporaneity and International Compliance

The study programme is consistent with the modern scientific trends and state of the profession in the world, and is comparable with similar programs at foreign universities.
The study programme as conceptualized is thorough and comprehensive, providing students with the latest scientific and professional knowledge in this area and following new achievements in science.
The study programme of Industrial Engineering / Egineering Management is comparable and complies with several doctorial academic studies both throughout the world and in Europe.
1. Northwestern University, Evanston, IL, USA
http://www.iems.northwestern.edu/images/pdf/MajorMinor.pdf
2. Helsinki University of Technology, Helsinki, Finland
http://www.tuta.hut.fi/studies/postgraduates/pstgrGUIDE_dr1995.pdf
3. Koç University, Istanbul, Turkey
http://www.iems.northwestern.edu/images/PDF/CoreTopics.pdf
4. Groupe des Ecoles des Mines, Paris, Sain-Etienne & Nantes, France
http://www.gemtech.fr/66919641/1/fiche___pagelibre/#4

http://kontakt.tu-hamburg.de/en/gen/fsp.html
https://engineering.purdue.edu/IE/Academics/PhD
Both formally and structurally, the study programme is consistent with the approved subject-specific standards of accreditation and complies with European standards in terms of enrollment, length of study, conditions of transition to the next year, graduation and type of study.

Standard 07. Student Enrollment

Faculty of Technical Sciences announces an admission of candidates to the program of doctoral studies Industrial Engineering / Engineering Management in accordance with social needs, their free resources and approved number of students in the accreditation process. The number of students who will be enrolled and funding of their studies (budget or self-financing) is defined each year by a Scientific Council of faculty.

For admission can apply Candidates who have completed the corresponding Masters or graduate studies and whose previous studies evaluated a total of at least 300 ECTS, which is defined in the Regulations on student enrollment in courses.

Candidates who, in the opinion of the Commission, have completed an appropriate program of study are eligible to enroll in doctoral studies. Commission for the registration shall decide whether candidates are eligible to take entrance examination enrollment. If Quality Commission decision on taking the classification exam, the candidates take this the exam which addresses to testing knowledge of the program of study. If the number of candidates is less than the number of accredited students and the Commission finds that the candidates have completed an appropriate program of study, the Commission may decide that it is not necessary to organize the classification exam.
The final ranking of candidates for admission is based on success in previous studies, the duration of the study and achieved success on the entrance exam, as defined in Regulation on student enrollment in courses.
The Commission, in accordance with Regulation on student enrollment in degree programs, has the right to approve the registration of candidates who have not completed the corresponding Masters or postgraduate studies which are worth a minimum of 300 ECTS, and only in the event that vacancies remain after the registration of all candidates who meet the requirements Competition set (corresponding previous studies, passed the classification exam). Candidates with the professional judgment of the Commission, have not completed the relevant study program of undergraduate studies may be granted admission if they pass the entrance exam.
Members of the Council of doctoral studies are at the same time members of the Commission for the registration of doctoral studies in accordance with Regulation on student enrollment in degree programs.

Standard 08. Student Evaluation and Progress

The final grade for each of the courses of this programme is formed by continuous monitoring of the students` work and the results during the school year and the final exam.
The student masters the programme by taking exams, earning thereby a certain number of ECTS credits, in accordance with the study programme. Each individual course in this programme carries a certain number of ECTS credits earned by the student when successfully passing the exam.
The number of ECTS credits is identified based on the student`s workload in mastering a particular course and by applying a uniform methodology of the Faculty of Technical Sciences for all study programs. The success of students in mastering a particular course is continuously monitored during the class and is expressed in credits. The maximum number of credits a student can earn on the course is 100.
The student earns credits on the course by working during the classes, and by satisfying pre-exam requirements. The minimum and maximum number of credits a student can achieve by satisfying pre-exam requirements is 30 and 70, respectively.
Each course from the curriculum has a clear and published way of earning credits. The way of earning credits during the classes includes a number of credits that students earn from each individual type of activity in the classroom or by satisfying pre-exam requirements.
The total success of the student on the course is expressed by grades from 5 (failed) to 10 (excellent). The student`s grade is based on the total number of credits earned by satisfying the pre-exam requirements and on the quality of acquired knowledge and skills.
Studying the degree program is realized as follows:
At the enrollment, the Head of the study programme (study group) appoints each student a mentor from among professors who will lead them until a mentor is choosen.
At the end of the semester, the mentor submits a report to the Head of the study (group) on student`s work during the research and his results.
Requirements for enrolling the second year of the study (third semester) is defined by Regulation.
The right to take the qualifying exam for the preparation and defense of the doctoral dissertation (research work on theoretical bases of doctoral dissertation) is gained by the student who has certified the second year of study and passed all the exams foreseen by the study programme.
Students who do not meet the requirement for taking the exam from the theoretical basis of the doctoral dissertation have the possibility to continue their studies at specialist academic studies with the exams being recognized.
Research work on the theoretical bases of the doctoral dissertation is a qualifying exam for the doctoral dissertation. Theoretical bases are taken as an exam (written and / or oral) by fields (questions) from at least three courses of the study program. The list of fields (questions) from which the qualifying exam is taken is submitted to the candidate by the Head of the doctoral study program within 14 days upon the request. The qualifying exam is taken before a committee of at least three members, appointed by the Manager of doctoral studies upon the proposal submitted by the Committee for quality of the study programme. Upon the request of students, the theoretical bases of doctoral dissertation can be taken not earlier than 30 days and not later than 12 months from taking the final exam.
Examinations for doctoral studies can be taken up to three times.
The final part of doctoral studies is the preparation and defense of the doctoral thesis.
Exams for doctoral studies can be taken up to three times.
The final part of doctoral studies is the creation and defense of a thesis.

Standard 09. Teaching Staff

For realization of the study program of Industrial Engineering / Engineering Management the teaching staff is provided with the necessary professional and academic qualifications, as evidenced by the list of papers and data on participation in national and international research projects. The competence of lecturers is based on scientific papers published in international journals, with at least one paper being published or accepted for publication in a SCI list journal, scientific papers published in national journals, papers published in proceedings of international conferences, monographs, patents, textbooks, new products or significantly improved existing products.

The mentor has at least five scientific papers published or accepted for publication in scientific journals in a given area. The mentor can not lead simultaneously more than five doctoral students. The selection of a mentor is defined so that mentor must have at least 5 papers published in SCI journals.

The number of professors meet the needs of the curriculum and depend on the number of courses and number of classes on these courses. The total number of lecturers is sufficient to cover the total number of classes at the study programme(lectures, consultations, practical work, ...). The minimum number of full-time lecturers participating in a given academic program is at least five.
Scientific and professional qualifications of the teaching staff match the educational and scientific field and the level of their responsibilities. Each lecturer has at least 10 references from specific scientific or technical fields of teaching in the program of study.
All data on lecturers and associates (CV, elections for the position, references) are available to the public.

Science, arts and professional qualifications

Name and last name:	Buchmeister S. Borut
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	1996	Faculty of Mechanical Engineering, University of Maribor - Maribor	Production Systems, Organization and Management
Magister thesis	1990	Faculty of Mechanical Engineering, University of Maribor - Maribor	Production Systems, Organization and Management
Bachelor's thesis	1986	Faculty of Mechanical Engineering, University of Maribor - Maribor	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	M316	Production Systems	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
2.	IM1104	Strategic Management	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1106	Business Process Simulation	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
4.	IM1118	Business Productivity Tools	(I20) Engineering Management, Undergraduate Academic Studies
5.	HDOK4S	Selected chapters from automation of work processes	(I12) Industrial Engineering, Specialised Academic Studies
6.	I071B	Strateško upravljanje projektima(uneti naziv na engleskom)	(Z20) Environmental Engineering, Master Academic Studies
7.	IM2101	Intelligent Enterprising and Effective Management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
8.	IM2103	New technologies in engineering and management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
9.	HDOK-4	Selected Chapters in Production Process Automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
10.	HDOKL4	Selected chapters from automation of work processes	(H00) Mechatronics, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	PANDŽA, Krsto, POLAJNAR, Andrej, BUCHMEISTER, Borut, THORPE, Richard. Evolutionary perspectives on the capability accumulation process. Int. j. oper. prod. manage., 2003, vol. 23, no. 8, str. 822-849. [COBISS.SI-ID 8111638], [JCR, WoS do 6. 12. 2011: št. citatov (TC): 9, čistih citatov (CI): 9, normirano št. čistih citatov (NC): 35, Scopus do 17. 6. 2012: št. citatov (TC): 11, čistih citatov (CI): 11, normirano št. čistih citatov (NC): 43]
2.	BUCHMEISTER, Borut, KREMLJAK, Zvonko, PANDŽA, Krsto, POLAJNAR, Andrej. Simulation study on the performance analysis of various sequencing rules. Int. j. simul. model., June/September 2004, vol. 3, no. 2/3, str. 80-89. [COBISS.SI-ID 9075990]
3.	PANDŽA, Krsto, POLAJNAR, Andrej, BUCHMEISTER, Borut. Strategic management of advanced manufacturing technology. Int. j. adv. manuf. technol., 2005, vol. 25, 3/4, str. 402-408. http://dx.doi.org/10.1007/s00170-003-1804-x. [COBISS.SI-ID 9383190], [JCR, WoS do 6. 5. 2011: št. citatov (TC): 6, čistih citatov (CI): 5, normirano št. čistih citatov (NC): 9, Scopus do 10. 9. 2012: št. citatov (TC): 14, čistih citatov (CI): 13, normirano št. čistih citatov (NC): 23]
4.	KREMLJAK, Zvonko, POLAJNAR, Andrej, BUCHMEISTER, Borut. Hevristični model razvoja proizvodnih zmogljivosti = A heuristic model for the development of production capabilities. Stroj. vestn., 2005, letn. 51, št. 11, str. 674-691. [COBISS.SI-ID 8659739], [JCR, WoS do 6. 11. 2012: št. citatov (TC): 6, čistih citatov (CI): 5, normirano št. čistih citatov (NC): 8, Scopus do 18. 6. 2012: št. citatov (TC): 7, čistih citatov (CI): 6, normirano št. čistih citatov (NC): 9]
5.	TASIČ, Tadej, BUCHMEISTER, Borut, AČKO, Bojan. Razvoj naprednih metod za vodenje proizvodnih postopkov = The development of advanced methods for scheduling production processes. Stroj. vestn., 2007, letn. 53, št. 12, str. 844-857. [COBISS.SI-ID 12075030], [JCR, WoS do 6. 12. 2011: št. citatov (TC): 9, čistih citatov (CI): 8, normirano št. čistih citatov (NC): 11, Scopus do 1. 8. 2012: št. citatov (TC): 9, čistih citatov (CI): 8, normirano št. čistih citatov (NC): 11]

Representative refferences (minimum 5, not more than 10)

6.	KREMLJAK, Zvonko, BUCHMEISTER, Borut. Uncertainty and development of capabilities, (DAAAM Publishing series, Management Science). Vienna: DAAAM International Publishing, 2006. X, 143 str., graf. prikazi. ISBN 3-901509-55-0. [COBISS.SI-ID 57398785]
7.	POLAJNAR, Andrej, BUCHMEISTER, Borut, LEBER, Marjan. Proizvodni menedžment. Ponatis. V Mariboru: Fakulteta za strojništvo, 2005. VI, 415 str., 28 str. pril., ilustr., preglednice. ISBN 86-435-0379-7. [COBISS.SI-ID 54649089]
8.	BUCHMEISTER, Borut, PANDŽA, Krsto, PALČIČ, Iztok. Idejna študija o ustanavljanju regionalnega logističnega centra za vzdrževanje in popravila vojaških in namenskih vozil. Maribor: Fakulteta za strojništvo, 2002. 28, 6 f. pril., ilustr. [COBISS.SI-ID 7612438]l
9.	PALČIČ, Iztok, BALAŽIC, Matej, MILFELNER, Matjaž, BUCHMEISTER, Borut. Potential of laser engineered net shaping (LENS) technology. Mater. manuf. process., 2009, vol. 24, no. 7/8, str. 750-753, doi: 10.1080/10426910902809776. [COBISS.SI-ID 13243670], [JCR, WoS do 6. 11. 2012: št. citatov (TC): 6, čistih citatov (CI): 5, normirano št. čistih citatov (NC): 5, Scopus do 8. 8. 2012: št. citatov (TC): 7, čistih citatov (CI): 6, normirano št. čistih citatov (NC): 6]
10.	PALČIČ, Iztok, BUCHMEISTER, Borut, POLAJNAR, Andrej. Analysis of innovation concepts in Slovenian manufacturing companies. Stroj. vestn., 2010, vol. 56, no. 12, str. 803-810. http://www.sv-jme.eu/scripts/download.phpfile=/data/upload/2010/12/03_2010_083_Palcic_3k.pdf. [COBISS.SI-ID 14634774], [JCR, WoS do 6. 11. 2012: št. citatov (TC): 7, čistih citatov (CI): 7, normirano št. čistih citatov (NC): 8, Scopus do 17. 10. 2012: št. citatov (TC): 8, čistih citatov (CI): 8, normirano št. čistih citatov (NC): 9]

Summary data for teacher's scientific or art and professional activity:

Quotation total :	43			
Total of SCI(SSCI) list papers :	15			
Current projects :	Domestic :	1	International :	1

Science, arts and professional qualifications

Name and last name:	Adžić Z. Nevenka
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.09.1978
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2002	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	1990	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	1986	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1976	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E121	Mathematical Analysis 2	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
2.	E221A	Mathematical Analysis 2	(E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
3.	GG10	Mathematical Methods 3	(G00) Civil Engineering, Undergraduate Academic Studies
4.	M106	Mathematics 2	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
5.	S017	Mathematics 2	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
6.	S0213	Mathematical Statistics	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
7.	Z104	Mathematics 1	(Z01) Safety at Work, Undergraduate Academic Studies (ZC0) Clean Energy Technologies, Undergraduate Academic Studies (ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
8.	BMI91	Mathematics 1	(BM0) Biomedical Engineering, Undergraduate Academic Studies
9.	BMI92	Mathematics 2	(BM0) Biomedical Engineering, Undergraduate Academic Studies
10.	E101A	Discrete Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
11.	IM1012	Probability and Statistics	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
12.	IM1523	Discrete Mathematics	(M30) Energy and Process Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
13.	P216	Numerical Analysis	(P00) Production Engineering, Undergraduate Academic Studies
14.	0M517	Numerical Analysis	(OM1) Mathematics in Engineering, Master Academic Studies
15.	0ML517	Numerical Analysis	(OM1) Mathematics in Engineering, Master Academic Studies
16.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
17.	D0M24	Numerical Solutions of Differential Equations	(OM1) Mathematics in Engineering, Doctoral Academic Studies
18.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
19.	AID06	Graph theory	(F20) Engineering Animation, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	N. Adzic, On the spectral solution for boundary value problem, ZAMM 70,(1990) 6, T647-T649.
2.	V. Vrcelj, N. Adzic, Z. Uzelac: A numerical asymptotic solution for singular perturbation problems, International journal of computer mathematics, Vol.39, (1991) 229-238.
3.	N. Adzic: Modified hermite polynomials in the spectral approximation for boundary layer problems, Bulletin of the Australian mathematical society, Vol.45, (1992) 267-276.<\eng>
4.	N. Adzic: Spectral approximation for single turing point problem, ZAMM72(1992)6, T621-T624.
5.	N. Adzic: Nonclassical orthogonal polynomials and singularly perturbed problems, ZAMM73(1993) 7/8, T868-T871.
6.	N. Adzic: Spectral approximation and asymptotic behaviour of boundary layer problems, ZAMM74(1994)6, T-553-T555.
7.	N. Adzic, Z. Uzelac: A combination of spline and spectral approximation for a class of singularly perturbed problems, ZAMM78 (1998), S853-S854
8.	Z. Uzelac, N. Adzic: The Approximate Solution for Problems with Nonlocal Boundary Conditions, ZAMM79 (1999), S881-S882
9.	N. Adzic, Z. Uzelac: On spectral approximation for some two-dimensional singularly perturbed problems, ZAMM79 (1999), S851-S852
10.	N. Adzic: On the spectral approximation for singularly perturbed problems,ZAMM 71(1991)6,T773-T776.

Summary data for teacher's scientific or art and professional activity:				
Quotation total :	5			
Total of SCI(SSCI) list papers :	10			
Current projects :	Domestic :	2	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Anišić M. Zoran
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2002	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Magister thesis	1997	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	1993	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II1012	Assembly Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
2.	IM1011	Applied Operational Research	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
3.	IM1013	Product Development	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1112	Technological and Business Forcasting	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1212	Decision Theory	(I20) Engineering Management, Undergraduate Academic Studies
6.	IMDS67	Selected Chapters in Product Lifecycle Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
7.	IMDSPI	Selected Chapters in Design for Excellence	(I12) Industrial Engineering, Specialised Academic Studies
8.	PLM02	Product Development and Management in PLM	(I10) Industrial Engineering, Master Academic Studies (I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
9.	IM2207	Technology management	(I20) Engineering Management, Master Academic Studies
10.	IM2213	Product and Service Management	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
11.	IM2216	Technology transfer and intellectual property management	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies (I20) Engineering Management, Master Academic Studies
12.	PLM02	Applied Product Development	(I20) Engineering Management, Specialised Professional Studies
13.	IMDR67	Selected Chapters in Product Lifecycle Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
14.	IMDR91	Product Family Development and Product Configurators	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
15.	IMDR92	Advanced Forecasting Methods and Techniques	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
16.	IMDRPI	Selected Chapters in Design for Excellence	(F00) Graphic Engineering and Design, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Ćosić, I., Anišić, Z., Lazarević, M.: Tehnološki sistemi u montaži, FTN, Novi Sad, str.290, UDK 621.717-52(075.8), ISBN 978-86-7892-448-4, 2012

Representative refferences (minimum 5, not more than 10)

2.	Ćosić, I., Anišić, Z.: Tehnologije montaže - priručnik za vežbe, FTN Novi Sad, str.255, UDK 658.515(075.8)(076) ISBN 978-86-7892-390-6, 2012.
3.	Ćosić, I., Anišić, Z.: MONTAŽNE TEHNOLOGIJE – POSTUPCI I SISTEMI ZA SPAJANJE, Novi Sad, Fakultet tehničkih nauka, 2006. 130str., UDK: 621.88(075.8), ISBN 86-85211-73-5.
4.	Anišić, Z.: RAZVOJ POSTUPKA ZA DINAMIČKO MODELIRANJE I TEHNOEKONOMSKU OPTIMIZACIJU MONTAŽNIH SISTEMA, Fakultet tehničkih nauka, Novi Sad, 1997,
5.	Anišić, Z.: SOME RESULTS OF THE IMPLEMENTATION OF THE MC CONCEPT IN SMALL COMPANIES, 2nd International Conference on Mass Customization in Central Europe, Rzeszow, Poland: Univesrity for Technology and Informatics, 2006, str. 5-25, ISBN 83-87658-96-0.
6.	Suzić N., Anišić Z., Ćosić I.: Reconfiguring Production and Organizational Structures for Mass Customization in Furniture Industry; Chapter 20 of Innovative Production Systems Key to Future Inteligent Manufacturing; Scientific Monography, Maribor, University of Maribor, Faculty of Mechanical Engineeing, Maribor; Faculty of Mechanical Engineering, Skopje, 2010, str. 257-275, ISBN 978-961-248-250-3
7.	Anišić, Z., Krsmanović, C.: ASSEMBLY INITIATED PRODUCTION AS A PREREQUISITE FOR MASS CUSTOMIZATION AND EFFECTIVE MANUFACTURING, Strojniški vestnik - Journal of Mechanical Engineering 54(2008)9, 607-618, UDC 658.5.
8.	Firstner (Fürstner) I., Anišić Z., Takač M.: Product Configurator Self-Adapting to Different Levels od Customer Knowledge, Acta Polytechnica Hungarica – Journal of Applied Sciences, 2012, Vol. 9, No 4, pp. 129-150, ISSN 1785-8860
9.	Suzić N., Stevanov B., Ćosić I., Anišić Z., Sremčev N.: Customizing Products trough Application of Group Technology: A Case Study of Furniture Manufacturing, Strojniski vestnik = Journal of Mechanical Engineering, 2012, ISSN 0039-2480
10.	Gečevska V., Lombardi F., Čuš F., Anišić Z., Angelidis D., Veza I., Vasilevska S., Ćosić P.: PLM – Product Lifeycle Management Strategy for Innovative and Competitive Business Environment, Maribor, University of Maribor, Faculty of Mechanical Engineering, Faculty of Mechanical Engineering Skopje, 2010, str. 193-208, ISBN 978-961-248-250-3

Summary data for teacher's scientific or art and professional activity:

Quotation total :	43			
Total of SCI(SSCI) list papers :	3			
Current projects :	Domestic :	0	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Atanacković M. Teodor
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	18.03.1975
Scientific or art field:	Deformable Body Mechanics

Academic carieer	Year	Institution	Field
Academic title election:	1988	Faculty of Technical Sciences - Novi Sad	Deformable Body Mechanics
PhD thesis	1974	Faculty of Technical Sciences - Novi Sad	Deformable Body Mechanics
Magister thesis	1973	Faculty of Technical Sciences - Novi Sad	Deformable Body Mechanics
Bachelor's thesis	1969	Faculty of Technical Sciences - Novi Sad	Thermal Energetics and Thermotechnics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	A237	Material Resistance	(A00) Architecture, Undergraduate Academic Studies
2.	H202	Strength of materials	(H00) Mechatronics, Undergraduate Academic Studies
3.	A002S	Scientific Research Method	(A00) Architecture, Specialised Academic Studies (E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
4.	DAU003	Selected Chapters in Mechanics	(E20) Computing and Control Engineering, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
5.	DZ001	Scientific Research Method	(A00) Architecture, Doctoral Academic Studies (AS0) Scenic Design, Doctoral Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
6.	SID04	Current State in the Field	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies
7.	SID04	Present State in the Field	(A00) Architecture, Doctoral Academic Studies (AS0) Scenic Design, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	T. M. Atanackovic, Stability Theory of Elastic Rods. World Scientific, 1997.
2.	T. M. Atanackovic, A. Guran, Theory of Elasticity for Scientists and Engineers. Birkhauser, 2000..
3.	B. D Vujanovic, T. M. Atanackovic, An Introduction to Modern Variational Techniques in Mechanics and Engineering. Birkhauser, Boston 2004..
4.	T.M. Atanackovic, Stability of a Compressible Elastic Rod with Imperfections. Acta Mechanica. 76, 203?222 (1989)..
5.	T.M. Atanackovic and M. Achenbach, Moment-curvature relations for a pseudoplastic beam. Continuum Mech. Thermodyn. 1, 73-80 (1989)...
6.	T.M. Atanackovic and I. Müller, A New form of ther Coherency Energy in Pseudoelasticity. Meccanica, 30, 467-474 (1995).
7.	T. M. Atanackovic, Optimal shape of column with own weight: bi and single modal optimization. Meccanica 41, 173-196 (2006).
8.	T. M. Atanackovic, S. Pilipovic, D. Zorica, Diffusion wave equation with two fractional derivatives of different order. J. Phys. A: Math. Theor. 40, 5319-5333 (2007).
9.	T. M. Atanackovic, Optimal shape of an elastic rod in flexural – torsional buckling. Z. Angew. Math. Mech.(ZAMM) 87, No. 6, 399 – 405 (2007).
10.	T. M. Atanackovic and B. N. Novakovic, Optimal Shape of an elastic column on elastic foundation. European J. Mechanics, A/Solids, 25, 154-165 (2006).

Summary data for teacher's scientific or art and professional activity:

Quotation total :	220			
Total of SCI(SSCI) list papers :	120			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Avdalović A. Veselin
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2012		Production Systems, Organization and Management
PhD thesis	2000	Faculty of Economics - Subotica	Economic Science
Magister thesis	1997	Faculty of Economics - Subotica	Economic Science
Bachelor's thesis	1992	Faculty of Economics - Subotica	Economics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	URZP47	Fire Risk Management in Industry	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
2.	URZP60	Risk Analysis Methods	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
3.	IM1024	Risk Management and insurance	(I20) Engineering Management, Undergraduate Academic Studies
4.	S0I321	Insurance for traffic and transport	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
5.	URZP80	Basic principals of insurance	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
6.	OIR001	Basic insurance	(I20) Engineering Management, Specialised Professional Studies
7.	OIR002	Insurance risks	(I20) Engineering Management, Specialised Professional Studies
8.	IM2719	Loss Assessment	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
9.	IM2720	Reinsurance	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
10.	IMDS75	Selected Topics in Risk Management and Insurance Management	(I22) Engineering Management, Specialised Academic Studies
11.	IMDR75	Selected Topics in Risk Management and Insurance Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Menadžment rizikom u osiguranju, Beograd, Želind, 2000. ISBN 86-7307-104-6
2. Osiguranje i upravljanje rizikom, Subotica, Birografika, 2003. UDK: COBISS.SR-ID 185914119
3. Menadžment - marketing osiguranja, Subotica, Merkur, 2004. UDK: COBISS.SR-ID 196573959
4. Osiguranje i upravljanje rizikom, Novi Sad, DDOR, 2005. UDK: COBISS.SR-ID 120990476
5. Osiguranje i teorija rizika, Beogradska bankarska akademija i CAM Novi Sad, 2006. ISBN 86-7852-007-8
6. Osiguranje, Beograd, Beogradska bankarska akademija, 2007. ISBN 978-86-7852-013-6
7. Principi osiguranja, Novi Sad, Fakultet tehničkih nauka, 2007. ISBN 978-86-7892-058-5
8. Ispitivanje instrumentalnih komponenti u menadžmentu društva za osiguranje i reosiguranje, Univerzitet u Novom Sadu, Ekonomski fakultet Subotica, 1997.
9. Menadžment kontroling društva za osiguranje, Univerzitet u Novom Sadu, Ekonomski fakultet, Subotica, 2000.
10. Veselin Avdalović: Kreativne tehnike u definisanju i rešavanju strategijskih problema organizacije, Strategijski menadžment, 1997, No. 2, str. 64- 69, ISSN 0354-8414.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0

Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	1	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Beker A. Ivan
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.12.1987
Scientific or art field:	Quality, Effectiveness and Logistics

Academic carieer	Year	Institution	Field
Academic title election:	2012		Quality, Effectiveness and Logistics
PhD thesis	2001	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	1996	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1986	Faculty of Technical Sciences - Novi Sad	Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	URZP49	Logistics in the Conditions of Catastrophic Events	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
2.	II1016	Reliability of technical systems and Maintenance	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	II1040	Organization and mamanagement of maintenance	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	II1043	Maintenance techniques and technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
5.	IM1030	Integral Systems Support - Logistic	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
6.	IM1036	Reliability Theory	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1049	Supply chain Management	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1614	Organization and Management of Logistic	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1615	Maintenance of Technical Equipment	(I20) Engineering Management, Undergraduate Academic Studies
10.	IM1618	Design and Analysis of Maintenance Procedure	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
11.	IM1620	Reverse and Green Logistic	(I20) Engineering Management, Undergraduate Academic Studies
12.	IM1622	Information Security Management System	(I20) Engineering Management, Undergraduate Academic Studies
13.	IM1623	Occupational Health and Safety Management System	(I20) Engineering Management, Undergraduate Academic Studies
14.	I501	Risk Management	(I10) Industrial Engineering, Master Academic Studies
15.	I841	Spare parts management	(I10) Industrial Engineering, Master Academic Studies
16.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
17.	IMDS95	Trends in Customer Relationship Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
18.	PLM10	Product Servicing and Maintenance	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
19.	LIM16	Production Logistics	(LIM) Logistic Engineering and Management, Master Academic Studies
20.	LIM18	Life Cycle Costs and Supply	(LIM) Logistic Engineering and Management, Master Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
21.	LIM30	Inventory Planning and Management	(LIM) Logistic Engineering and Management, Master Academic Studies
22.	I843	Maintenance effectiveness	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
23.	IIDS12	Quality and organizational performance	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
24.	IIDS30	Trends in the environmental management systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
25.	IIDS7	Selected topics in quality engineering and logistics	(I12) Industrial Engineering, Specialised Academic Studies
26.	IM2607	Risk management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
27.	IM2615	Lean Logistics	(I20) Engineering Management, Master Academic Studies
28.	IM2617	Information Systems to Support Quality, Logistics and Maintenance	(I20) Engineering Management, Master Academic Studies
29.	IM2618	Transportation management	(I20) Engineering Management, Master Academic Studies
30.	IM2619	Stock planning and management	(I20) Engineering Management, Master Academic Studies
31.	IM2620	Lean Maintenance	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
32.	IM2622	Design and Implementation of Health and Safety System	(I20) Engineering Management, Master Academic Studies
33.	IMDS74	Selected Topics in Quality Management and Logistics	(I22) Engineering Management, Specialised Academic Studies
34.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
35.	IMDR94	Trends in the environmental management systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
36.	IMDR95	Trends in Customer Relationship Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
37.	IMDR74	Selected Topics in Quality Management and Logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
38.	IMDR79	Selected topics in quality engineering and logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
39.	IMDR83	Quality abd organisational performance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
40.	ZRD232	Logistics in the Security Services and Health at Work	(Z01) Safety at Work, Doctoral Academic Studies
41.	ZRD29A	Selected Topics in Systems Reliability	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Brkljač N., Šević D., Beker I., Kesić I., Milisavljević S.: Procedure for treatment of hazardous waste by MID-MIX procedure in Serbia, International Journal of the Physical Sciences, 2012, Vol. 7, No 18, pp. 2639-2646, ISSN 1992-1950
2. Radlovački V., Pečujlija M., Kamberović B., Jovanović R., Delić M., Beker I.: SATISFACTION OF HIGH SCHOOL STUDENTS WITH THE APPLICABILITY OF THEIR KNOWLEDGE, TTEM. Tehnics tehnologies education management, 2012, Vol. 7, No 2, pp. 777-785, ISSN 1840-1503
3. Jocanović M., Šević D., Karanović V., Beker I., Dudić S.: Increased Efficiency of Hydraulic Systems Through Reliability Theory and Monitoring of System Operating Parameters, Strojniški vestnik - Journal of Mechanical Engineering, 2012, Vol. 58, No 4, pp. 281-288, ISSN 0039-2480
4. Radlovački V., Beker I., Majstorović V., Pečujlija M., Stanivuković D., Kamberović B.: Quality Managers' Estimates of Quality Management Principles Application in Certified Organisations in Transitional Conditions - Is Serbia Close to TQM, Strojniški vestnik - Journal of Mechanical Engineering, 2011, Vol. 57, No 11, pp. 851-861, ISSN 0039-2480
5. Pouzdanost tehničkih sistema, autori prof. dr Gradimir Ivanović, prof. dr Dragutin Stanivuković, prof. dr Ivan Beker; Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, 2010, ISBN 978-86-7892-247-3
6. Beker I.: ZAPTIVANjE I ZAPTIVNI MATERIJALI, FTN -Institut za industrijske sisteme i IIS - Istraživački i tehnološki centar, Novi Sad, 2001
7. D. Stanivuković, B. Sabo, T. Furman. I. Beker, V. Bajić, J. Dakić: Tehnologije reparature i regeneracije delova, Časopis Traktori i pogonske mašine, Novi Sad, oktobar 1998
8. D. Šević, I. Beker, S. Milisavljević: UPOREDNA ANALIZA ZAHTEVA STANDARDA ISO 14001:2004 I STANDARDA ISO 14001:1996., International Journal Total Quality Management & Excellence, Vol.34, No 3 – 4, 2006.

Representative refferences (minimum 5, not more than 10)

9.	I. Beker, N. Radaković: ISKUSTVA NA IMPLEMENTACIJI ISO 27001 STANDARDA, International Journal Total Quality Management & Excellence, Vol.34, No 3 – 4, 2006.
10.	D. Stanivuković, S. Kecojević, I. Beker: Projektovanje održavanja na modularnom principu, 1 str., Tribologija u industriji, godina XV, broj 2 - juni 1993., Kragujevac, 1993.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	4			
Current projects :	Domestic :	0	International :	4

Science, arts and professional qualifications

Name and last name:	Borocki V. Jelena
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.11.2007
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2009	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	1997	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	1993	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E2I41	Information System Engineering	(E20) Computing and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies
2.	EOS33	Entrepreneurial management	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
3.	II1041	Innovation and Entrepreneurship	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	IM1005	Entrepreneurship	(I20) Engineering Management, Undergraduate Academic Studies (Z01) Safety at Work, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
5.	IM1021	Developmental Processes in Company	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1031	Enterprise's organization	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
7.	IM1045	Innovation in Enterprises	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1206	Innovation and Change Management	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1214	Management of Research and Development	(I20) Engineering Management, Undergraduate Academic Studies
10.	IM1216	Entrepreneurship in high technology	(I20) Engineering Management, Undergraduate Academic Studies
11.	IM1217	Entrepreneurship and New Business Venturing	(I20) Engineering Management, Undergraduate Academic Studies
12.	IM1218	Models of open innovations and corporate entrepreneurship	(I20) Engineering Management, Undergraduate Academic Studies
13.	IM1220	Entrepreneurial strategies	(I20) Engineering Management, Undergraduate Academic Studies
14.	IM1222	Managing intellectual capital of enterprise	(I20) Engineering Management, Undergraduate Academic Studies
15.	EE546	Entrepreneurship in Electrical Engineering	(E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
16.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
17.	IMDS61	Innovative business operations of enterprise	(I22) Engineering Management, Specialised Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
18.	IMDS65	Entrepreneurship and Organizational Development	(I22) Engineering Management, Specialised Academic Studies
19.	MBA412	Strategy of Technological Innovations	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
20.	MBA414	Integrated Business Processes	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
21.	MBA515	decision macing and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
22.	IIDS19	Organizational structures	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
23.	IM2217	Technology based Entrepreneurship	(I20) Engineering Management, Master Academic Studies
24.	IM2219	Strategic Entrepreneurship	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
25.	IM2220	Instruments of entrepreneurship and regional development	(I20) Engineering Management, Master Academic Studies
26.	IM2221	Innovation measurement	(I20) Engineering Management, Master Academic Studies
27.	IMDS70	Advanced topics on Innovation and Entrepreneurship	(I22) Engineering Management, Specialised Academic Studies
28.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
29.	IMDR12	Organizational structures	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
30.	IMDR61	Enterprise Innovative Business	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
31.	IMDR65	Entrepreneurship and Organizational Development	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
32.	IMDR70	Advanced topics on Innovation and Entrepreneurship	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Bojović, V., Borocki, J., Mirosavljev, M., Radovanović J., Rašković, V., Šenk, V., VODIČ ZA INOVATIVNE PREDUZETNIKE
2. Borocki, J., Cosic, I., Lalic, B., Maksimovic, R., Analysis of company development factors in manufacturing and service company: a strategic approach, Strojniski vestnik - Journal of Mechanical Engineering, 0039-2480, pp.55-68
3. Katic (Drezgic) I., Borocki J., Zekic S., Penezic N.: Entrepreneurship significance in restructuring process, TTEM. Tehnics tehnologies education management, 2011, Vol. 6, No 4, pp. 902-907, ISSN 1840-1503
4. Raskovic, V., Senk, V., Borocki, J., Cosic, I.: PROMOTING ENTREPRENEURIAL THINKING IN WOULD-BE AND EXISTING HIGH-TECH COMPANIES IN SERBIA, Promoting Entrepreneurship by Universities, Hämeenlinna, Finland: FINPIN, HAMK University of Applied Sciences and Häme Convention Bureau, april, 2008, pp. 83- 90, ISBN 978-951-827-096-9.
5. Djakovic, V., Andjelic, G., Borocki, J., Performance of extreme value theory in emerging markets: an empirical treatment, African Journal of Business and Management, ISSN: 1993-8233
6. Vidicki P., Borocki J., Senk V., Raskovic V.: Innovation activities in enterprise: different models of measurement, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Faculty of Technical Science, September 14-16, 2011, pp. 473-478, ISBN 978-86-7892-341-8, UDK: 658.5
7. Borocki J., Senk V.: ANALYSIS OF INNOVATION FACTORS OF MICRO AND SMALL COMPANIES: A STRATEGIC APPROACH, 3. International Conference for Entrepreneurship, Innovation and Regional Development ICEIRD, Novi Sad: Proceedings of the 3rd nternational Conference on Entrepreneurs, Innovation and Regional Development - ICEIRD 2010, Novi Sad, Faculty of Technical Sciences, Department of Industrial Engineering and Management, 27-29 Maj, 2010, pp. 61-68, ISBN 978-86-7892-250-3
8. Borocki, J., Maksimovic, R.: STRATEGIC PLANNING IN A FUNCTION OF ORGANIZATIONAL INNOVATIVENESS, International Conference on INDUSTRIAL SYSTEMS IS'08, Novi Sad: University of Novi Sad, Faculty of Technical Sciences, 02-03. October, 2008, pp. 415- 420, UDK: 658.5(082), ISBN 978-86-7892-135-3.

Representative refferences (minimum 5, not more than 10)

9.	Borocki J., Raskovic V., Senk V.: EDUCATING WOULD-BE AND EXISTING HIGH- TECH ENTREPRENEURS IN THE MARKET AND BUSINESS AREA , 1. International Conference for Entrepreneurship, Innovation and Regional Development ICEIRD, Skoplje: Business Start-up Centre, University "Ss. Ciril and Methodius" - Skopje, 9-11 Maj, 2008, pp. 72-77, ISBN 978-9989-2636-4-4, UDK: 001.896(062),005(062),005.591(062),334.722(062)
10.	Borocki J.: Doktorska disertacija Naziv: RAZVOJ MODELA STRATEGIJSKOG PLANIRANJA U FUNKCIJI INOVATIVNOSTI PREDUZEĆA, Novi Sad, 2009

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	3			
Current projects :	Domestic :	2	International :	1

Science, arts and professional qualifications

Name and last name:	Borovac A. Branislav
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.1975
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	1998	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	1986	Faculty of Technical Sciences - Novi Sad	Robotics and Flexible Automation
Magister thesis	1982	Faculty of Technical Sciences - Novi Sad	Robotics and Flexible Automation
Bachelor's thesis	1975	Faculty of Technical Sciences - Novi Sad	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EM436	Mechatronics	(M30) Energy and Process Engineering, Undergraduate Academic Studies
2.	H102	Fundamentals in Product Development	(H00) Mechatronics, Undergraduate Academic Studies
3.	H1404	Mechatronics	(H00) Mechatronics, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
4.	H308	Industrial Robotics	(H00) Mechatronics, Undergraduate Academic Studies
5.	I600	Industrial Robotics	(F10) Engineering Animation, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
6.	BM116A	Basics of medical robotics	(BM0) Biomedical Engineering, Undergraduate Academic Studies
7.	EM436A	Mechatronics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
8.	II1035	Industrial robotics	(I10) Industrial Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
9.	H1503	Non Industrial Robotics and Automation in Buildings	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
10.	HDOK1S	Selected topics in industrial robotics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
11.	HDOK2S	Selected topics in non-industrial robotics	(I12) Industrial Engineering, Specialised Academic Studies
12.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
13.	NIT05	Advanced Technology for Material Handling	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
14.	AD0007	Interactive systems in architecture	(AD0) Digital Techniques, Design and Production in Architecture and Urban Planning, Master Academic Studies
15.	H828	Advanced robotics	(H00) Mechatronics, Master Academic Studies
16.	H829	Advanced robotics	(I10) Industrial Engineering, Master Academic Studies (M40) Technical Mechanics and Technical Design, Master Academic Studies
17.	IIDS6	Selected chapters in automation	(I12) Industrial Engineering, Specialised Academic Studies
18.	GD018	Automation and Robotics in Construction	(G00) Civil Engineering, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
19.	HDOK-1	Selected Chapters in Industrial Robotics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
20.	HDOK-2	Selected Chapters in Non-Industrial Robotics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
21.	HDOKL1	Selected topics in non-industrial robotics	(H00) Mechatronics, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies
22.	HDOKL2	Selected topics in non-industrial robotics	(H00) Mechatronics, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies
23.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	IMDR80	Selected chapters in automation	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	M. Vukobratović, V. Potkonjak, K. Babković, B. Borovac, Simulation model of general human and humanoid motion, Multibody System Dynamics, Volume 17, Number 1, (February, 2007), pp. 71-96 (ISSN 1384-5640 (Print) 1573-272X (Online))
2.	Vukobratović M., Borovac B., Potkonjak V., Towards a Unified Understanding of Basic Notions and Terms in Humanoid Robotics, Robotica (2007) Vol. 25, pp. 87-101
3.	Vukobratović M., Borovac B., Potkonjak V., ZMP: A Review of Some Basic Misunder-standings, Int. Jour. of Humanoid Robotics, Vol. 3, No. 2 (2006), pp. 153-176
4.	V. Potkonjak, M. Vukobratović, K. Babković, B. Borovac, General Model of Dynamics of Human and Humanoid Motion: Feasibility, Potentials and Verification, Int. Jour. of Humanoid Robotics, Vol. 3, No. 2 (2006), pp. 21-48
5.	Vukobratović M., Borovac B., Babković K., "Contribution to the Study of Anthropomorphism of Humanoid Robots", Int. Jour. of Humanoid Robotics, Vol. 2, No. 3 (2005), pp. 361-387
6.	Vukobratović M., Borovac B., Note on the Article "Zero-Moment Point- Thirty Five Years of its Life", Int. Jour. of Humanoid Robotics, Vol. 2, No.2, June 2005, pp. 225-227
7.	Vukobratović M., Borovac B., "Zero-Moment Point- Thirty Five Years of its Life", Int. Jour. of Humanoid Robotics, Vol. 1, No.1, March 2004, pp. 157-173
8.	M. Vukobratović, D. Andrić, B. Borovac, “How to Achieve Various Gait Patterns from Single Nominal “, International Journal of Advanced Robotic Systems, Vol. 1., No. 2, Page 99-108, 2004
9.	L. Juhas, A. Vujanić, N. Adamović, L. Nagy, B. Borovac "A Platform for Micro-Positioning Based on Piezo-Legs", The Journal of Mechatronics, Vol. 11, (2001), pp.869-897
10.	M. Vukobratović, D. Andrić, B. Borovac, “Humanoid Robot Motion in Unstructured Environment - Generation of Various Gait Patterns from a Single Nominal “, Cutting Edge Robotics, Edited by V. Kordic, A. Lazanica, M. Merdan, Published by pIV pro literatur Ver-lag Robert Mayer-Scholz, © 2005 Advanced Robotic Systems International, Page 577-598, 2005

Summary data for teacher's scientific or art and professional activity:

Quotation total :	1998			
Total of SCI(SSCI) list papers :	35			
Current projects :	Domestic :	2	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Bošković M. Dragan
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Information-Communication Systems

Academic carieer	Year	Institution	Field
Academic title election:	2009		Information-Communication Systems
PhD thesis	1991	University of Bath - Bristol	Electrical and Computer Engineering
Magister thesis	1988	School of Electrical Engineering - Beograd	Electrical and Computer Engineering
Bachelor's thesis	1983	School of Electrical Engineering - Beograd	Electrical and Computer Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EM404A	Computer Electronics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
2.	IM1512	Object-oriented Infromation Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
3.	IM1515	Mobile information technologies	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1520	Service-Oriented Architectures	(I20) Engineering Management, Undergraduate Academic Studies
5.	IIDS8	Selected chapters from Information, management and communication systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies
6.	IM2507	Automation of production systems management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
7.	IM2517	e Government systems	(I20) Engineering Management, Master Academic Studies
8.	IMDS73	Selected chapters from Information management	(I22) Engineering Management, Specialised Academic Studies
9.	IMDR73	Selected chapters from Information management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
10.	IMDR81	Selected chapters from Information, management and communication systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Pennock, S.R. Boskovic, D.M. Rozzi, T., Analysis of coupled inset dielectric guides under LSE and LSM polarization', IEEE Transactions on Microwave Theory and Techniques, May 1992 Volume: 40, Issue: 5 On page(s): 916-924 Digital Object Identifier: 10.1109/22.137398
2. Bourse, D.; El-Khazen, K.; Lee, A.; Grandblaise, D.; Boscovic, D. "Business perspectives of end-to-end reconfigurability", IEEE Wireless Communications, [see also IEEE Personal Communications] Volume 13, Issue 3, June 2006 Page(s):46 – 57.
3. Demestichas, P.; Stavroulaki, V.; Boscovic, D.; Lee, A.; Strassner, J. 'm@ANGEL: autonomic management platform for seamless cognitive connectivity to the mobile internet', IEEE Communications Magazine, Volume 44, Issue 6, June 2006 Page(s):118 – 127.
4. D. Boscovic, M. Needham, F. Vakil and J. Yang, Low Carbon Economy considerations in designing and operating Content Delivery Networks for VoD ,Journal of Green Engineering, ISSN 1904-4720, River Publishers 2010
5. Dragan Boskovic, Vakil Faramak, Milenko Tosic, Stanisa Dautovic, Greening of video streaming to mobile devices by pervasive wireless CDN – Journal of Green Engineering, ISSN 1904-4720, River Publishers 2011
6. Faure, C.; Tin Lin Lee; Boscovic, D., 'UMTS border planning issues', IEEE VTS 53rd Vehicular Technology Conference, 2001. VTC 2001 Spring. Volume 4, 6-9 May 2001 Page(s):2761 - 2765 vol.4 Digital Object Identifier 10.1109/VETECS.2001.944103.
7. Dragan Bošković, Faramak Vakil, Content Delivery Networks for Video on Demand and IPTV Telekomunikacije, Vol 4 December 2009
8. Bourse, D.; El-Khazen, K.; Lee, A.; Boscovic, D.; Business Models of End-to-End Reconfigurable Systems Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd Volume 1, 2006 Page(s):57 - 61 Digital Object Identifier 10.1109/VETECS.2006.1682775

Representative refferences (minimum 5, not more than 10)

9.	Dragan Boskovic, Vakil Faramak, Milenko Tosic, Stanisa Dautovic Pervasive wireless CDN for greening video streaming to mobile devices ,– MiPRO conference, Opatija 2011
10.	Ning Xu, Jin Yang, Mike Needham, Dragan Boscovic, Faramak Vakil - Toward the Green Video CDN IEEE/ACM Int'l Conference on Green Computing Hangshou, Zhejiang Province, China, December 18-December 2010

Summary data for teacher's scientific or art and professional activity:

Quotation total :	30			
Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	0	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Budinski-Petković M. Ljuba
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.1989
Scientific or art field:	Physics

Academic carieer	Year	Institution	Field
Academic title election:	2009		Physics
PhD thesis	1998	Faculty of Sciences - Novi Sad	Physics
Magister thesis	1996	Faculty of Physics - Beograd	Physics
Bachelor's thesis	1988	Faculty of Sciences - Novi Sad	Physics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E215	Physics	(E20) Computing and Control Engineering, Undergraduate Academic Studies
2.	H101	Physics	(F10) Engineering Animation, Undergraduate Academic Studies (GI0) Geodesy and Geomatics, Undergraduate Academic Studies (H00) Mechatronics, Undergraduate Academic Studies
3.	IAFI01	Colors and Light	(F10) Engineering Animation, Undergraduate Academic Studies
4.	BMI93	Physics	(BM0) Biomedical Engineering, Undergraduate Academic Studies
5.	DZ01FS	Selected Chapters in Physics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
6.	DZ01F	Selected Chapters in Physics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Budinski-Petković Lj., Lončarević I., Petkovic M., Jaksic Z., Vrhovac S.: Percolation in random sequential adsorption of extended objects on a triangular lattice, Physical Review E, 2012, Vol. 85, No 061117, pp. 1-8
2. Šćepanović J., Lončarević I., Budinski-Petković Lj., Jakšić Z., Vrhovac S.: Relaxation properties in a diffusive model of k-mers with constrained movements on a triangular lattice, Physical Review E, 2011, Vol. 84, No 031109, pp. 1-13
3. Budinski-Petković Lj., Lončarević I., Jakšić Z., Vrhovac S., Švrakić N.: Simulation study of anisotropic random sequential adsorption of extended objects on a triangular lattice, Physical Review E, 2011, Vol. 84, No 5, pp. 5160-1

Representative refferences (minimum 5, not more than 10)

4.	Lončarević I., Budinski-Petković Lj., Vrhovac S., Belić A.: Generalized random sequential adsorption of polydisperse mixtures on a one-dimensional lattice, Journal of Statistical Mechanics: Theory and Experiment, 2010, ISSN 1742-5468
5.	Lončarević I., Budinski-Petković Lj., Vrhovac Lj., Belić A.: Adsorption, desorption, and diffusion of k-mers on a one-dimensional lattice, Physical Review E, 2009, Vol. 80, No 2
6.	Budinski-Petković Lj., Vrhovac S., Lončarević I.: Random sequential adsorption of polydisperse mixtures on discrete substrates, Physical Review E, 2008, Vol. 78, No 061603, pp. 1-7
7.	Lončarević I., Budinski-Petković Lj., Vrhovac S.: Simulation study of random sequential adsorption of mixtures on a triangular lattice , The European Physical Journal E, 2007, Vol. 24, pp. 19-26, ISSN 1292-8941
8.	Lončarević I., Budinski-Petković Lj., Vrhovac S.: Reversible random sequential adsorption of mixtures on a triangular lattice, Physical Review E, 2007, Vol. 76, No 031104, pp. 1-9
9.	Arsenović D., Vrhovac S., Jakšić Z., Budinski-Petković Lj., Belić A.: Simulation study of granular compaction dynamics under vertical tapping, Physical Review E, 2006, Vol. 74
10.	Lj. Budinski-Petković and S. B. Vrhovac: Memory effects in vibrated granular systems: Response properties in the generalized random sequential adsorption model, The European Physical Journal E, 2005, Vol. 16, pp. 89-96, ISSN 1292-8941

Summary data for teacher's scientific or art and professional activity:

Quotation total :	75			
Total of SCI(SSCI) list papers :	30			
Current projects :	Domestic :	1	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Bunčić M. Sonja
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2002	Faculty of Law - Novi Sad	Legal Science
Magister thesis	1999	Faculty of Law - Novi Sad	Legal Science
Bachelor's thesis	1984	Faculty of Law - Novi Sad	Legal Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	GI021	Structure Value Assessment	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
2.	GI405	Law and Legislation in Geodetic Profession	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
3.	IM1009	Business Law	(I20) Engineering Management, Undergraduate Academic Studies
4.	MBA307	European and international business and trade law	(IB0) Engineering Management - MBA, Specialised Professional Studies
5.	MBA521	The European Union-development process	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
6.	MBA523	European law/International law	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
7.	IM2121	Corporate governance	(I20) Engineering Management, Master Academic Studies
8.	IMDS82	Industrial eco-marketing management	(I22) Engineering Management, Specialised Academic Studies
9.	SDGI3D	Selected topics in real estate law	(GI0) Geodesy and Geomatics, Specialised Academic Studies
10.	IMDR82	Industrial eco-marketing management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Pravna priroda akcije
2.	Berzansko pravo
3.	Pravni položaj banke
4.	Buncic S., Filipovic M.:The future of international financial bussiness: Global regulatory framework, African Journal of Business Managment Vol 5 (9) , 4 May 2011, str 3749-3756
5.	Bunčić S:. G-20 od Pitsburga do Toronta put ka novoj finansijskoj regulativi, časopis Srpska politička misao 3/2010, str 271-288
6.	Bunčić S.:Dvadeset godina procesa privatizacije u zemljma na prostoru bivše Jugosslavije modeli i rezultati., Srpska politička misao , 2/2012 str. 201-222
7.	Bunčić S.:Lisabonski ugovor i EMU, Pravni život,Beograd,14/2008,s.127-137
8.	Bunčić S.: Zaštita manjiskih akcionara-da li novi Zakon o privrednim društvima donosi napredak? ,Pravni život 11/2011, str,137-153
9.	Bunčić S.:Pravni pristup određenju opcijskog posla , Pravni život 14/2009, s. 315-327
10.	Bunčić : Određenje pojma manjinski akcionari i njihova klasifikacija, Pravo i privreda,4-6/2011, str151-162

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	1			
Current projects :	Domestic :	1	International :	1

Science, arts and professional qualifications

Name and last name:	Crnojević S. Vladimir
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	10.11.1995
Scientific or art field:	Telecommunications and Signal Processing

Academic carieer	Year	Institution	Field
Academic title election:	2010		Telecommunications and Signal Processing
PhD thesis	2004	Faculty of Technical Sciences - Novi Sad	Telecommunications and Signal Processing
Magister thesis	1999	Faculty of Technical Sciences - Novi Sad	Telecommunications and Signal Processing
Bachelor's thesis	1995	Faculty of Technical Sciences - Novi Sad	Telecommunications and Signal Processing

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EK412	Shape Recognition	(BM0) Biomedical Engineering, Undergraduate Academic Studies
2.	EK421	Digital Image Processing	(F10) Engineering Animation, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
3.	URZP32	Systems for Detection, Alarm and Warning	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
4.	BM129A	Digital Image Processing	(BM0) Biomedical Engineering, Undergraduate Academic Studies
5.	E137	Basics of Telecommunications	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
6.	EK463	Pattern Recognition	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
7.	DE311S	Selected topics in Pattern Recognition	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
8.	DE412S	Digital image processing algorithms	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
9.	DE511S	Wireless sensor networks	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
10.	EK520	Medical Image Processing	(E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
11.	EK522	Computer Vision (Digital Image Processing 2)	(F20) Engineering Animation, Master Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
12.	H1420	Fundamentals in Mechanical Vision	(H00) Mechatronics, Master Academic Studies
13.	IMDS54	Computer Vision in Industrial Engineering and Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
14.	ZP508	Design and Maintenance of the Fire Detection Systems	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
15.	DE311	Selected Chapters in Pattern Recognition	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
16.	DE412	Digital Image Processing Algorithms	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
17.	DE511	Wireless Sensor Networks	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
18.	IMDR54	Computer Vision in Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

Representative refferences (minimum 5, not more than 10)

1.	Dejan Vukobratovic, Cedomir Stefanovic, Vladimir Crnojevic, Francesco Chiti, Romano Fantacci: "Rateless Packet Approach for Data Gathering in Wireless Sensor Networks", IEEE Journal on Selected Areas in Communications, Vol. 28, No. 7, pp. 1169-1179, September 2010.
2.	Petrovic, N.I.; Crnojevic, V.: Universal Impulse Noise Filter Based on Genetic Programming, IEEE Transactions on Image Processing, 2008, Vol. 17, No. 7, str. 1109- 1120, ISSN 1057-7149
3.	D. Culibrk, M. Mirkovic, V.Zlokolica, M. Pokric, V. crnojevic, D. Kukolj, "Salient Motion Features for Video Quality Assessment", IEEE Trans. on Image Processing, Volume: 20 Issue:4, pp(s): 948 - 958, ISSN: 1057-7149
4.	Cedomir Stefanovic, Dejan Vukobratovic, Francesco Chiti, Lorenzo Niccolai, Vladimir Crnojevic, Romano Fantacci: "Urban Infrastructure-to-Vehicle Traffic Data Dissemination Using UEP Rateless Codes", IEEE Journal on Selected Areas in Communications, Vol. 29, No. 1, pp. 94-102, January 2011.
5.	Vladimir Crnojević, Nemanja Petrović, „Impulse Noise Filtering Using Robust Pixel-Wise S-estimate of Variance“, EURASIP Journal on Advances in Signal Processing, vol. 2010, Article ID 830702, 10 pages, 2010,
6.	V. Crnojević, V. Šenk, Ž. Trpovski, “Advanced Impulse Detection Based on Pixel-Wise MAD”, IEEE Signal Processing Letters, vol.11, No. 7, 2004, str. 589-593. Crnojević, V. Šenk, Ž. Trpovski, “Advanced Impulse Detection Based on Pixel-Wise MAD”, IEEE Signal Processing Letters, vol.11, No. 7, 2004, str. 589-593.
7.	B. Antić, V. Crnojević, „Joint Domain-Range Modeling of Dynamic Scenes with Adaptive Kernel Bandwidth“, pp.777-788, LNCS 4678, Springer-Verlag, Berlin Heidelberg 2007.
8.	N. Petrović, V. Crnojević, „Evolutionary Tree-Structured Filter for Impulse Noise Removal“, pp.103-113, LNCS 4179, Springer-Verlag, Berlin Heidelberg 2006.
9.	N. Petrović, V. Crnojević, „Impulse Noise Detection Based on Robust Statistics and Genetic Programming“, pp.643-649, LNCS 3708, Springer-Verlag, Berlin Heidelberg 2005.
10.	V. Crnojević,„Impulse Noise Filter With Adaptive Mad-Based Threshold“, International Conference on Image Processing, Genoa, Italy, 11-14. September, 2005.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	135			
Total of SCI(SSCI) list papers :	10			
Current projects :	Domestic :	3	International :	10

Science, arts and professional qualifications

Name and last name:	Čuš -. Franci
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Proizvodni sistemi, organizacija i menadžment (menađment inovacija i

Academic carieer	Year	Institution	Field
Academic title election:	2009		Proizvodni sistemi, organizacija i menadžment (menađment inovacija i promena)
PhD thesis	1988	Faculty of Mechanical Engineering - Maribor	Processes for Material Removal Processing
Magister thesis	1985	Faculty of Mechanical Engineering - Maribor	Processes for Material Removal Processing
Bachelor's thesis	1978	Faculty of Mechanical Engineering - Maribor	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	Z421	Operacioni menadžment(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
2.	II1053	Production Systems	(F00) Graphic Engineering and Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
3.	IM1114	Energy Flows in the Enterprise	(I20) Engineering Management, Undergraduate Academic Studies
4.	ZR401A	Science on Work	(Z01) Safety at Work, Undergraduate Academic Studies
5.	HDOK4S	Selected chapters from automation of work processes	(I12) Industrial Engineering, Specialised Academic Studies
6.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
7.	ZR502	Occupational Risk Assessment	(Z01) Safety at Work, Master Academic Studies
8.	IM2102	Manufacturing strategy (KAIZEN, LEAN, KANBAN, EFPS)	(I10) Industrial Engineering, Master Academic Studies (M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
9.	IM2124	Production and Service Systems	(H00) Mechatronics, Master Academic Studies (M50) Energy Management, Master Academic Studies
10.	IM2207	Technology management	(I20) Engineering Management, Master Academic Studies
11.	IM2215	Value engineering	(I20) Engineering Management, Master Academic Studies
12.	HDOK-4	Selected Chapters in Production Process Automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
13.	HDOKL4	Selected chapters from automation of work processes	(H00) Mechatronics, Doctoral Academic Studies
14.	IMDR57	Strategic Planning and Designing Procedures and Systems at the End of Product Lifecycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
15.	ZRD27A	Operations management in the security and occupational safety	(Z01) Safety at Work, Doctoral Academic Studies
16.	ZRD28A	Selected topics in the science of occupational safety	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. ČUŠ, Franc, BALIČ, Jože. Optimization of cutting process by GA approach. Robot. comput.-integr. manuf.. [Print ed.], 2003, vol. 19, iss. 1/2, str. 113-121.
2. ČUŠ, Franc, MURŠEC, Bogomir. Databases for technological information systems. J. mater. process. technol.. [Print ed.], Dec. 2004, vol. 157/158, str. 75-81.
3. ČUŠ, Franc, ŽUPERL, Uroš, MILFELNER, Matjaž. Dynamic neural network approach for tool cutting force modelling of end milling operations. Int. j. gen. syst., October 2006, vol. 35, no 5, str. 603-618. [COBISS.SI-ID 10604310]
4. ČUŠ, Franc, MILFELNER, Matjaž, BALIČ, Jože. An intelligent system for monitoring and optimization of ball-end milling process. J. mater. process. technol.. [Print ed.], June 2006, vol. 175, iss. 1/3, str. 90-97.
5. ČUŠ, Franc, ŽUPERL, Uroš, KIKER, Edvard, MILFELNER, Matjaž. Adaptive controller design for feedrate maximization of machining process. J. Achiev. Mater. Manuf. Eng., Jul.-Aug. 2006, vol. 17, iss. 1/2, str. 237-240.

Representative refferences (minimum 5, not more than 10)

6.	ČUŠ, Franc, ŽUPERL, Uroš. Approach to optimization of cutting conditions by using artificial neural networks. J. mater. process. technol.. [Print ed.], 2006, vol. 173, iss. 3, str. 281-290.
7.	ČUŠ, Franc, BALIČ, Jože, ŽUPERL, Uroš. Hybrid ANFIS-ants system based optimisation of turning parameters. J. Achiev. Mater. Manuf. Eng., Sep. 2009, vol. 36, iss. 1, str. 79-86.
8.	ŠOSTAR, Adolf, ČUŠ, Franc. Vpliv toplotne obdelave na obdelovalnost materialov pri vrtanju. Stroj. vestn., 1983, let. 29, št. 10-12, str. 215-218. [COBISS.SI-ID 3324444]
9.	ŠOSTAR, Adolf, ČUŠ, Franc. Načrtovanje preizkusov in izračun eksponentov za optimiranje odrezovanja. Stroj. vestn., 1984, let. 30, št. 9-10, str. 197-203. [COBISS.SI-ID 3324700]
10.	ČUŠ, Franc. Odvisnosti in zakonitosti postopka čelnega frezanja. Stroj. vestn., 1986, 32, št. 4/6, str. 60-63. [COBISS.SI-ID 94468]

Summary data for teacher's scientific or art and professional activity:

Quotation total :	21			
Total of SCI(SSCI) list papers :	28			
Current projects :	Domestic :	0	International :	1

Science, arts and professional qualifications

Name and last name:	Ćosić P. Ilija
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	22.12.1972
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	1993	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	1983	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Magister thesis	1979	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	1972	Faculty of Mechanical Engineering - Novi Sad	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	M316	Production Systems	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
2.	II1017	Production System Design	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	II1053	Production Systems	(F00) Graphic Engineering and Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
4.	IM1027	Production systems	(I20) Engineering Management, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
5.	IM1039	Fundamentals of Operations management	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies (ZC0) Clean Energy Technologies, Undergraduate Academic Studies (ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
6.	IM1116	Work Study and Ergonomics	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
7.	ZR401A	Science on Work	(Z01) Safety at Work, Undergraduate Academic Studies
8.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
9.	IMDSPI	Selected Chapters in Design for Excellence	(I12) Industrial Engineering, Specialised Academic Studies
10.	IS001	Effective management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	ZR502	Occupational Risk Assessment	(Z01) Safety at Work, Master Academic Studies
12.	IIDS5	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies
13.	IIDS9	Effective Production and Service Systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
14.	IM2101	Intelligent Enterprising and Effective Management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
15.	IM2102	Manufacturing strategy (KAIZEN, LEAN, KANBAN, EFPS)	(I10) Industrial Engineering, Master Academic Studies (M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
16.	IM2119	Layout and location of the enterprise	(I20) Engineering Management, Master Academic Studies
17.	IM2124	Production and Service Systems	(H00) Mechatronics, Master Academic Studies (M50) Energy Management, Master Academic Studies
18.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
19.	IMDR31	Effective Production and Service Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
20.	IMDR56	Traceability of Product Lifecycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
21.	IMDR57	Strategic Planning and Designing Procedures and Systems at the End of Product Lifecycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
22.	IMDRPI	Selected Chapters in Design for Excellence	(F00) Graphic Engineering and Design, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
23.	IMDR5	Selected chapters in enterprise's design, organization and control	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	IMDR85	Effective technological and production structures	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	ZRD27A	Operations management in the security and occupational safety	(Z01) Safety at Work, Doctoral Academic Studies
26.	ZRD28A	Selected topics in the science of occupational safety	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Ćosić I.: Development of Knowledge-Based Syslem for the Configuralion of Assembly Systems, Knowledge-Based Selection arid Arragement of Parts Bins at Assembly Workplaces (TEBES) - Europian Communities Brusseles, 1991
2. Simeunović N., Ćosić I., Radaković N., Lalić B.: The General Work Procedure Model for the Service Product, Beč, DAAAM International Scientific Book, 2009, str. 281-288, ISBN 987-3-901509-71-1, UDK: ISSN 1726-9687
3. Pečujlija M., Ćosić I., Ivanišević V.: A professor's moral thinking at the abstract level vs the professor's moral thinking in real life situation (consistency problem), Science and Engineering Ethics, 2011, Vol. 17, No 2, pp. 299-320, ISSN 1353-3452
4. Zelenović D., Ćosić I., Šormaz D., Šišarica Z.: An approach to the design of more effective production systems , International Journal of Production Research, 1987, Vol. 25, No 1, pp. 3-15, ISSN 0020-7543
5. Kirin S., Sedmak A., Grubić-Nešić L., Ćosić I.: Project risk management in complex petrochemical system, Hemijska industrija, 2012, pp. 52-52, ISSN 0354-7531, UDK: doi:10.2298/HEMIND110709052K
6. Lazarević M., Ostojić G., Ćosić I., Stankovski S., Vukelić Đ., Zečević I.: Product lifecycle management (PLM) methodology for product tracking based on radio-frequency identification (RFID) technology, Scientific Research and Essays, 2011, Vol. 6, No 22, pp. 4776-4787, ISSN 1992-2248
7. Tešić Z., Lalić D., Ćosić I., Mitrović V.: Integration of information for manufacturing shop control, Strojniski vestnik = Journal of Mechanical Engineering, 2010, Vol. 56, No 3, pp. 217-223, ISSN 0039-2480
8. Stankovski S., Lazarević M., Ostojić G., Ćosić I., Purić R.: RFID Technology in Product/Part Tracking During the Whole Life Cycle , Assembly Automation, 2009, Vol. 29, No 4, pp. 364-370, ISSN 0144-5154
9. Ostojić G., Lazarević M., Stankovski S., Ćosić I.: RFID Technology Application in Disassembly Systems , Strojniski vestnik = Journal of Mechanical Engineering, 2008, Vol. 54, No 11, pp. 759-767, ISSN 0039-2480, UDK: 658.5
10. Sremčev N., Ćosić I., Suzić N., Stevanov B.: APPLICATION OF PLM SYSTEMS IN GROUP TECHNOLOGY APPROACH, 23. DAAAM International Symposium, Zadar: DAAAM International, Vienna, Austria, EU, 2012, 24-27 Oktobar, 2012, pp. 981-984, ISBN 978-3-901509-91-9, UDK: ISSN 2304-1382

Summary data for teacher's scientific or art and professional activity:

Quotation total :	96			
Total of SCI(SSCI) list papers :	15			
Current projects :	Domestic :	2	International :	2

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Ćosić I. Đorđe
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.01.2007
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2010	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2007	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	2001	Faculty of Technical Sciences - Novi Sad	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	URZP33	Role and Importance of Prevention in Risk Reduction	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
2.	URZP36	Risks in Manipulating Hazardous Substances	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
3.	URZP41	Disasters and Vulnerability	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
4.	URZP46	Cycle Elements of Catastrophic Events	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
5.	URZP56	Fundamentals of Risk and Fire Protection Management	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
6.	IM1024	Risk Management and insurance	(I20) Engineering Management, Undergraduate Academic Studies
7.	S0I321	Insurance for traffic and transport	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
8.	URZP80	Basic principals of insurance	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
9.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
10.	OIR001	Basic insurance	(I20) Engineering Management, Specialised Professional Studies
11.	OIR002	Insurance risks	(I20) Engineering Management, Specialised Professional Studies
12.	Z511	Institucionalni okviri upravljanja akcidentnim rizicima(uneti naziv na engleskom)	(Z20) Environmental Engineering, Master Academic Studies
13.	ZP501	Integrated Natural Disaster Risk Management	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
14.	IM2707	Methods for the analysis of insurance risk	(I20) Engineering Management, Master Academic Studies
15.	IM2714	Disaster risk management cycle	(I20) Engineering Management, Master Academic Studies
16.	IM2717	Management of strategic and operational risks of insurance companies	(OM1) Mathematics in Engineering, Master Academic Studies
17.	IM2719	Loss Assessment	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
18.	IMDS75	Selected Topics in Risk Management and Insurance Management	(I22) Engineering Management, Specialised Academic Studies
19.	MPK009	Enviromental hazards	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies
20.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
21.	IMDR75	Selected Topics in Risk Management and Insurance Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
22.	ZRD233	Selected topics in the field of insurance from the standpoint of safety and health at work	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Pečujlija M., Ćosić Đ.: An Orthodox Christian Reflection: Genetic Enhancement Must not be the Creation Primacy Problem between Man and God, The American Journal of Bioethics, 2010, Vol. 10, No 4, pp. 78-80, ISSN 1526-5161
2.	Pečujlija M., Ćosić Đ., Bojanić R., Radišić S., Ivanović G., Delić Z.: Employees' Attitudes Towards Company Privatization as Possible Predictors of a High Performance Working System, African Journal of Business Management, 2011, Vol. 5, No 3, pp. 1663-1672, ISSN 1993-8233
3.	Ćosić Đ., Popov S., Sakulski D., Pavlović A.: Geo-Information Technology for Disaster Risk Assessment, Acta Geotechnica Slovenica, 2011, Vol. 8, No 2011/1, pp. 64-74, ISSN 1854-0171
4.	Pečujlija M., Azemović N., Azemović R., Ćosić Đ.: Leadership and productivity in transition: employees view in Serbia, Journal for East European Management Studies, 2011, Vol. 16, No 3, pp. 251-263, ISSN 0949-6181
5.	Njegomir V., Ćosić Đ.: Ekonomske implikacije klimatskih promena na sektor osiguranja i reosiguranja, Teme, 2012, Vol. 36, No 2, pp. 679-701, ISSN 0353-7919
6.	Sakulski D., Ćosić Đ., Popov S.: Implementation of Innovative Technologies for Disaster Risk Reduction, 1. International Conference Natural Hazards, Novi Sad: University of Novi Sad, Faculty of Science, 5 Maj, 2012, pp. 15-16, ISBN 978-86-7031-276-0
7.	Sakulski D., Ćosić Đ., Popov S., Pavlović A., Laban M.: Disaster risk management and fire safety, 1. International conference Protection, Ecology, Security, Bar: Fakultet za pomorstvo Kotor, 24-26 Maj, 2012, pp. 75-81
8.	Simić J., Popov S., Ćosić Đ., Sakulski D., Novaković T., Popović Lj., Pavlović A., Luhović A.: The aspect of bringing data in spatial relationship during the process of teaching at the subject “Disaster risk management” , UDK: 37.01:004 (082)
9.	Pavlović A., Ćosić Đ., Popov S., Kolaković S.: Indikatori praćenja hazardnih pojava poplave i suše u cilju poboljšanja planiranja melioracija, Tematski zbornik radova "Melioracije 07 - stanje i perspektive-", 2012, No 12, pp. 136-146, ISSN 978-86-7520-107-6, UDK: 626.8(082)
10.	Popović Lj., Popov S., Ćosić Đ., Sakulski D.: Impact of Visualization on Data Availability, UDK: CIP je dostupan u Univerzitetskoj biblioteci Rijeke pod brojem 121219001

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	2	International :	1

Science, arts and professional qualifications

Name and last name:	Ćulibrk R. Dubravko
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.02.2001
Scientific or art field:	Information-Communication Systems

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
PhD thesis	2006	Faculty of Technical Sciences - Novi Sad	Computer Engineering
Magister thesis	2003	Faculty of Technical Sciences - Novi Sad	Computer Engineering
Bachelor's thesis	2000	Faculty of Technical Sciences - Novi Sad	Computer Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	GI100	Computer Practicum	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
2.	IGB340	Fundamentals of Engineering Animation	(F10) Engineering Animation, Undergraduate Academic Studies
3.	II1002	Computer Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	II1024	Algorithms and Data Structures	(I10) Industrial Engineering, Undergraduate Academic Studies
5.	IM1010	Fundamentals of Information Technologies	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1038	Introduction to Business Intelligence Systems	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1517	Computer application development	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1522	Algorithms and Data Structures	(I20) Engineering Management, Undergraduate Academic Studies
9.	F402	Electronic Publishing	(F00) Graphic Engineering and Design, Master Academic Studies
10.	IMDS34	Raster and Image Processing Technologies in Engineering and Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
11.	IMDS54	Computer Vision in Industrial Engineering and Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
12.	IMDS55	Data Mining	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
13.	MBA411	Business intelligence concepts	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
14.	MM004	Theory and Practice of Media Communication	(I20) Engineering Management, Specialised Professional Studies
15.	MUO004	Information Systems in Education	(I20) Engineering Management, Specialised Professional Studies
16.	I835	Data mining methods	(I10) Industrial Engineering, Master Academic Studies
17.	I913	Expert systems and tools for knowledge management	(I10) Industrial Engineering, Master Academic Studies
18.	IIDS8	Selected chapters from Information, management and communication systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies
19.	IM2519	Advanced Information Technology	(I20) Engineering Management, Master Academic Studies
20.	IMDS73	Selected chapters from Information management	(I22) Engineering Management, Specialised Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
21.	IMDR34	Raster and Image Processing Technologies in Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
22.	IMDR54	Computer Vision in Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
23.	IMDR55	Data Research	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	IMDR73	Selected chapters from Information management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	IMDR81	Selected chapters from Information, management and communication systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	D. Culibrk, O. Marques, D. Socek, H. Kalva and B. Furht, "Neural Network Approach to Background Modeling for Video Object Segmentation", IEEE Trans. on Neural Networks, September 2007.
2.	D. Socek, D. Culibrk, O.F. Marques, H. Kalva and B. Furht, “A Hybrid Color-Based Foreground Object Detection Method for Automated Marine Surveillance”, in Proc. Advanced Concepts for Intelligent Vision Systems (ACIVS 2005), Antwerp, Belgium, September 20-23, 2005
3.	Ćulibrk, D., Daniel Socek and Michal Sramka: Cryptanalysis of a Symmetric Probabilistic Encryption Scheme Based on Chaotic Attractors of Neural Networks, Tatra Mountains Mathematical Publications, 2007, Vol. 37, str. 75- 91
4.	"New approaches to encryption and steganography for digital videos", Daniel Socek, Hari Kalva, Spyros S. Magliveras, Oge Marques, Dubravko Culibrk and Borko Furht, Multimedia systems, vol. 13, No 3, pp.
5.	Daniel Socek, Spyros Magliveras, Dubravko Ćulibrk, Oge Marques, Hari Kalva, and Borko Furht: Digital Video Encryption Algorithms Based on Correlation-Preserving Permutations, EURASIP Journal on Information Security, 2007, ISSN 1687-4161. 5.
6.	Dubravko Ćulibrk, Borislav Antić, Vladimir Crnojević: Real-time Stable Texture Regions Extraction for Motion-based Object Segmentation, 20th British Machine Vision Conference, BMVC 2009, London, UK: British Machine Vision Association, 7.-10. September, 2009
7.	D. Culibrk, M. Mirkovic, V.Zlokolica, M. Pokric, V. crnojevic, D. Kukolj, “Salient Motion Features for Video Quality Assessment”, IEEE Trans. on Image Processing, Volume: 20 Issue:4, pp(s): 948 – 958, ISSN: 1057-7149, 2011.
8.	J. Radonić, D. Ćulibrk, M. Vojinović-Miloradov, B. Kukić, M. Turk-Sekulić, Prediction Of Gas-Particle Partitioning Of Pahs Based On M5' Model Trees, Thermal Science, No. 1, vol. 15, pp.105-114 , 2011.
9.	Mladen Pečujlija, Dubravko Ćulibrk, Why We Believe The Computer When It Lies, Computers in Human Behavior, Volume 28, Issue 1, January 2012, Pages 143–152.
10.	D. Ćulibrk, M. Mancas, V. Crnojevic, 2012, "Dynamic Texture Recognition Based on Compression Artifacts", in Towards Advanced Data Analysis by Combining Soft Computing and Statistics in Fuzziness and Soft Computing Volume 285, 2013, pp 253-266.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	11			
Current projects :	Domestic :	2	International :	4

Science, arts and professional qualifications

Name and last name:	Dobromirov P. Dušan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2006
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2010		Production Systems, Organization and Management
PhD thesis	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Magister thesis	2006	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	2001	Faculty of Technical Sciences - Novi Sad	Management and Business

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IM1406	Investments Risk Management	(I20) Engineering Management, Undergraduate Academic Studies
2.	IM1413	Corporate restructuring	(I20) Engineering Management, Undergraduate Academic Studies
3.	M3499	Energy markets	(M30) Energy and Process Engineering, Undergraduate Academic Studies
4.	I904/S	The Theory and Practice of Corporate Finance	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
5.	IM005	International financial transactions	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
6.	IM006	Money and banking practical aspects	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
7.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
8.	IMDS47	Behavioral Corporate Finance	(I22) Engineering Management, Specialised Academic Studies
9.	IMDS87	Financial engineering of public sector	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
10.	SZP003	Selected Chapters in Applied Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	IM2407	International business and finance	(I20) Engineering Management, Master Academic Studies
12.	IM2420	Algoritmic trading	(I20) Engineering Management, Master Academic Studies
13.	IM2423	Energy markets	(M50) Energy Management, Master Academic Studies
14.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
15.	IMDR47	Behavioral Corporate Finance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
16.	IMDR87	Financial engineering of public sector	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Dušan Dobromirov "Strategija uvođenja i razvoja tržišta valutnih finansijskih derivata"
2.	Sando S., Radišić M., Dobromirov D.: Emerging markets - Galapagos for behavioral financial research (in print), Actual Problems of Economics, 2012, ISSN 1993-6788
3.	Marić B., Dobromirov D., Radišić M.: Researching the dependence between the dynamic indicators of investment profitability, African Journal of Business Management, 2011, Vol. 5, No 13, pp. 5076-5082, ISSN 1993-8233
4.	Bojović Ž., Šećerov E., Dobromirov D., Šenk V.: Maximizing the Profit of Telecom Telcos by a Novel Traffic Scheduling Policy , Electronics and electrical engineering, 2011, Vol. 7, No 113, pp. 67-73, ISSN 1392-1215
5.	Radišić M., Marić B., Dobromirov D.: SMEs and entrepreneurs investments' profitability effects within the transition period in the Republic of Serbia, African Journal of Business Management, 2011, Vol. 5, No 7, pp. 2654-2659, ISSN 1993-8233
6.	Dobromirov D., Radišić M., Kupusinac A.: Emerging markets arbitrages' perception: Risk versus growth potential, African Journal of Business Management, 2011, Vol. 5, No 3, pp. 713-721, ISSN 1993-8233
7.	Bojović ž., Šenk V., Dobromirov D., Bojović P.: Intervendor working of VOIP networks , Journal of the Institute of Telecommunications Professionals, 2011, Vol. 5, No 3, pp. 26-32, ISSN 1755-9278
8.	Borocki J., Dobromirov D., Radišić M., Milinković M.: Key success factors of companies' innovation activities, 2. Preduzetnička konferencija "Zapošljavanje kroz prizmu preduzetništva", Podgorica: Ekonomski fakultet, Univerzitet Crne Gore, 18 Maj, 2012
9.	Bašić B., Marić B., Dobromirov D., Radišić M.: MASS CUSTOMIZATION APPROACH IN PUBLIC SECTOR - AN EXAMPLE FROM TAX ADMINISTRATION, 5. International Conference on Mass Customization and Personalization in Central Europe MCP-CE, Novi Sad: Faculty of Technical Sciences, 19-21 Septembar, 2012, pp. 13-21, ISBN 978-86-7892-432-3
10.	Ferenčak M., Stanišić I., Radišić M., Dobromirov D.: Level of frictional unemployment in the Republic of Serbia, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Departman za industrijsko inženjerstvo i menadžment, Novi Sad, 14-16 Septembar, 2011, pp. 537-541, ISBN 978-86-7892-341-8

Summary data for teacher's scientific or art and professional activity:

Quotation total :	1			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Doroslovački D. Rade
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.1978
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2000	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	1989	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	1984	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1976	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E213	Discrete Mathematics and Linear Algebra	(E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
2.	E101	Discrete Mathematics	(ES0) Power Software Engineering, Undergraduate Academic Studies
3.	E101A	Discrete Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
4.	IM1523	Discrete Mathematics	(M30) Energy and Process Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
5.	IM1706	Actuerial Mathematics	(I20) Engineering Management, Undergraduate Academic Studies
6.	SE0009	Discrete Mathematics	(SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
7.	0M503	Combinatorics and Graph Theory	(OM1) Mathematics in Engineering, Master Academic Studies
8.	0M509	Applied Abstract Algebra	(OM1) Mathematics in Engineering, Master Academic Studies
9.	0M511	Geometry	(OM1) Mathematics in Engineering, Master Academic Studies
10.	0ML503	Combinatorics and Graph Theory	(OM1) Mathematics in Engineering, Master Academic Studies
11.	0ML509	Applaid Abstract Algebra	(OM1) Mathematics in Engineering, Master Academic Studies
12.	0ML511	Geometry	(OM1) Mathematics in Engineering, Master Academic Studies
13.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
14.	OM519	Actuerial Mathematics	(OM1) Mathematics in Engineering, Master Academic Studies
15.	OML519	Actuerial Mathematics	(OM1) Mathematics in Engineering, Master Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
16.	D0M08	Applied Abstract Algebra	(OM1) Mathematics in Engineering, Doctoral Academic Studies
17.	D0M17	Combinatorics	(OM1) Mathematics in Engineering, Doctoral Academic Studies
18.	D0M20	Graph Theory	(OM1) Mathematics in Engineering, Doctoral Academic Studies
19.	D0M34	Actuarial Mathematics	(OM1) Mathematics in Engineering, Doctoral Academic Studies
20.	DOM31	Combinatorial Matrix Theory	(OM1) Mathematics in Engineering, Doctoral Academic Studies
21.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	R. Doroslovački, R. Tošić and I. Stojmenović: Generating and counting triangular system, BIT: 27(1987) 18-24, Kobenhavn, R 54
2.	R. Doroslovački , R . Tošić i J. Gutman: Topological properties of benzenoid systems, XXXVIII, the boundary code, Match in mathematical chemistry (19) (219-228) Max- Plank-Institut fur Stranhlenchemije, Mulheim (1986)
3.	Rade Doroslovački: Binary Sequences without 01...10, Matematički vesnik, Mathematical Society of Serbia, 46 (1994), 93-98.
4.	Rade Doroslovački: On binary n-words with forbidden 4-subwords, (1997/01) Novi Sad Juornal of Mathematics.
5.	R. Doroslovački, J. Pantović, G.Vojvodić: Note on Itersection of Maximal Clones, (1998/02) Novi Sad, Journal of Mathematics.
6.	R. Doroslovački, J. Pantović, G. Vojvodić: Classification of Maps by their Membership in Maximal Clones that contain Minimum and Complement, Matematički vesnik,, Mathematical Society of Serbia, 51, (1999), 21-28
7.	Rade Doroslovački, Jovanka Pantović and Gradimir Vojvodić: One Interval in the Lattice of Partial Hyperclones, Czechoslovaka Mathematical Journal, 55 (130),2005, 719-724, (R52)
8.	O. Bodroža-Pantić, R. Doroslovački, K. Doroslovački, AN ELEMENTARY PROOF OF A THEOREM CONCERNING THE DIVISION OF A REGION INTO TWO," in Rocky Mountain Journal of Mathematics, Vol. 37, No.5, 2007, R 52
9.	O. Bodroža-Pantić, R. Doroslovački, The Gutman formulas for algebraic structure count, Journal of Mathematical Chemistrz Vol.35,No.2, Februar 2004, R 51.
10.	Ratko Tošić, Gradimir Vojvodić, Dragan Mašulović, Rade Doroslovački, Jovanka Rosić: Two examples of relative completeness, Multiple Valued Logic, An International Journal (Journal of Multiple-Valued Logic and Soft Computing), (1996), Vol. 2, pp. 67-78.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	60			
Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Dudić P. Slobodan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	21.08.1995
Scientific or art field:	Mechatronics, Robotics and Automation and Intelligent Systems

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Intelligent Systems
PhD thesis	2012	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Intelligent Systems
Magister thesis	1999	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	1995	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H102	Fundamentals in Product Development	(H00) Mechatronics, Undergraduate Academic Studies
2.	H1401	Material Handling Technologies	(H00) Mechatronics, Undergraduate Academic Studies
3.	H1403	Automation of work processes	(H00) Mechatronics, Undergraduate Academic Studies
4.	H1504	Computer Integration of Production Systems	(H00) Mechatronics, Undergraduate Academic Studies
5.	H310	Components of technological systems	(H00) Mechatronics, Undergraduate Academic Studies
6.	II1011	Automation of work processes 1	(I10) Industrial Engineering, Undergraduate Academic Studies
7.	II1013	Material Handling Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
8.	II1023	Packaging technology	(I10) Industrial Engineering, Undergraduate Academic Studies
9.	II1038	Automation of work processes 2	(I10) Industrial Engineering, Undergraduate Academic Studies
10.	II1042	Automation of Continual Processes	(I10) Industrial Engineering, Undergraduate Academic Studies
11.	IM1114	Energy Flows in the Enterprise	(I20) Engineering Management, Undergraduate Academic Studies
12.	H505	Implementation of automated systems	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
13.	HDOK4S	Selected chapters from automation of work processes	(I12) Industrial Engineering, Specialised Academic Studies
14.	I829	Automation of packaging processes	(I10) Industrial Engineering, Master Academic Studies
15.	I830	Energy efficiency of compressed air systems	(I10) Industrial Engineering, Master Academic Studies
16.	PLM02	Product Development and Management in PLM	(I10) Industrial Engineering, Master Academic Studies (I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
17.	PLM04	Sustainable Production and LCA	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
18.	LIM34	Material Handling	(LIM) Logistic Engineering and Management, Master Academic Studies
19.	NIT02	Factory Automation	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
20.	NIT05	Advanced Technology for Material Handling	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
21.	BMIM4C	Fluid filtration and separation	(BM0) Biomedical Engineering, Master Academic Studies
22.	I911	Sustainable production	(I10) Industrial Engineering, Master Academic Studies
23.	IIDS27	Selected chapters of the energy efficiency of automated systems	(I12) Industrial Engineering, Specialised Academic Studies
24.	IIDS6	Selected chapters in automation	(I12) Industrial Engineering, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
25.	IM2103	New technologies in engineering and management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
26.	IMDR86	Selected chapters from energy efficiency of compressed air systems	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
27.	IMDR80	Selected chapters in automation	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Šešlija D., Ignjatović I., Dudić S.: Increasing the Energy Efficiency in Compressed Air Systems, Rijeka, InTech, 2012, str. 151-174, ISBN 978-953-51-0800-9
2.	Dudić S., Ignjatović I., Šešlija D., Blagojević V., Miodrag S.: Leakage quantification of compressed air using ultrasound and infrared thermography, MEASUREMENT, 2012, Vol. 45, No 7, pp. 1689-1694, ISSN 0263-2241
3.	Ignjatović I., Šešlija D., Tarjan L., Dudić S.: Wireless sensor system for monitoring of compressed air filters, Journal of Scientific and Industrial Research (JSIR), 2012, Vol. 71, No 5, pp. 334-340, ISSN 0022-4456
4.	Jocanović M., Šević D., Karanović V., Beker I., Dudić S.: Increased Efficiency of Hydraulic Systems Through Reliability Theory and Monitoring of System Operating Parameters, Strojniški vestnik - Journal of Mechanical Engineering, 2012, Vol. 58, No 4, pp. 281-288, ISSN 0039-2480
5.	Dudić S., Ignjatović I., Šešlija D., Blagojević V., Stojiljković M.: Leakage quantification of compressed air on pipes using thermovision, Thermal Science, 2012, Vol. 16, No 2, pp. 621-631, ISSN 0354-9836
6.	Šešlija D., Ignjatović I., Dudić S., Lagod B.: Potential energy savings in compressed air systems in Serbia, African Journal of Business Management, 2011, Vol. 5, No 14, pp. 5637-5645, ISSN 1993-8233
7.	Blagojević V., Šešlija D., Stojiljković M., Dudić S.: Efficient control of servo pneumatic actuator system utilizing by-pass valve and digital sliding mode, Sadhana - Academy Proceedings in Engineering Science, 2012, ISSN 0256-2499
8.	Šešlija D., Ignjatović I., Dudić S.: Compressed air system structure and energy efficiency, 15. Symposium on Thermal Science and Engineering of Serbia, Soko Banja: University of Nis, Faculty of Mechanical Engineering and Society of Thermal Engineers of Serbia, 18-21 Oktobar, 2011, pp. 649-658, ISBN 978-86-6055-018-9
9.	Šešlija D., Dudić S., Ignjatović I.: Cost effectiveness t of pressure regulation on return stroke of pneumatic actuators, 11. International Scientific Conference "Flexible Technologies" - MMA, Novi Sad: Fakultet tehničkih nauka, 20-21 Septembar, 2012
10.	Dudić S., Ignjatović I., Šešlija D.: Usage of non-destructive methods in compressed air system, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Faculty of Technical Sciences, 14-16 Septembar, 2011, pp. 101-104, ISBN 978-86-7892-341-8

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	0	International :	0

Science, arts and professional qualifications

Name and last name:	Duđak D. Ljubica
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.09.1991
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2010	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2006	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1991	Faculty of Technical Sciences - Novi Sad	Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II934	Psichology of Work	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
2.	ISIT05	Introduction to organization and management	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
3.	II1022	Human resources in the process of work	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	IM1031	Enterprise's organization	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
5.	IM1050	Human Resources in the Knowledge Economy	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1912	Human Resource Planning	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1917	Employee Development and Training	(I20) Engineering Management, Undergraduate Academic Studies
8.	S01361	Business decision making	(S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
9.	HR005	PR Plan Development and Application	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
10.	HR016	Strategije i tehnike odnosa sa javnošću	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	HR017	Corporate Communication Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
12.	I076/S	Leadership and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
13.	I205/S	Razvoj ljudskih resursa	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
14.	I935/S	Motivating Employees	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
15.	IMDS52	Strategic Development of Human Resources	(I22) Engineering Management, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
16.	MBA513	leadership development and teamworking	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
17.	MBA515	decision macing and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
18.	MBA524	interculture business communications	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
19.	SZP003	Selected Chapters in Applied Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
20.	IM2121	Corporate governance	(I20) Engineering Management, Master Academic Studies
21.	IM2915	The performance of employees	(I20) Engineering Management, Master Academic Studies
22.	IM2919	Corporate social responsibility	(I20) Engineering Management, Master Academic Studies
23.	IMDS77	Selected Chapters from Human Resource Management	(I22) Engineering Management, Specialised Academic Studies
24.	IMDR52	Strategic Development of Human Resources	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	IMDR77	Selected Chapters from Human Resource Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	ZRD234	The strategy of human resource development from the standpoint of safety and health at work	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Lj. Duđak: Strategijski plan razvoja kadrova u preduzeću, Strategijski menadžment, 1996, Vol. 1, No. 4, str. 16- 23, ISSN 0354-8414
2. Lj. Duđak: OBUKA I RAZVOJ ZAPOSLENIH – FUNKCIJA MENADŽMENTA LJUDSKIH RESURSA, 12. Međunarodna naučno-stručna konferencija, Novi Sad: FTN - Institut za industrijsko inženjerstvo i menadžment, 22./23. novembar, 2002, str. 326- 331, UDK: 658.5
3. Lj. Duđak: SELEKCIJA KAO INSTRUMENT MENADŽMENTA LJUDSKIH RESURSA, 13. Međunarodna naučno - stručna konferencija "Industrijski sistemi – IS "05", Herceg Novi, Novi Sad: FTN - Odsek za industrijsko inženjerstvo i menadžment, 07./09. septembar, 2005, str. 725- 732, UDK: 658.5(082), ISBN 86-7780-008-5
4. Duđak Lj.: DEVELOPMENT AND TRAINING OF EMPLOYEES – THE ROAD TOWARDS AN INTELLIGENT BUSINESS, XIV Međunarodna konferencija INDUSTRIJSKI SISTEMI - IS 08 , UDK: 685.5(082)
5. Duđak Lj., Grubić-Nešić L., Andevski M.: Characteristics of Organizational Culture Necessary for Development and Training of Employees, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Fakultet tehničkih nauka, 14-16 Septembar, 2011, pp. 552-556, ISBN 978-86-7892-341-8
6. Duđak Lj., Savić-Šikoparija T., Hristić D.: The Importance of Internal and External Communication for the Acceptance and Implementation of Company's Corporate Responsibility, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Fakultet tehničkih nauka, 14-16 Septembar, 2011, pp. 563-568, ISBN 978-86-7892-341-8
7. Andevski M., Duđak Lj., Katić (Drezgić) I.: Director Role in Creating Culture Learning Organization at School , 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Fakultet tehničkih nauka, 14-16 Septembar, 2011, pp. 456-460, ISBN 978-86-7892-341-8
8. Grubić-Nešić L., Čabrilo S., Duđak Lj.: Istraživanje stavova prema promenama", 9. Međunarodna naučno-stručna konferencija „Na putu ka dobu znanja", Fakultet za menadžment, 2011, UDK: 316.4
9. Hristić D., Grubić-Nešić L., Duđak Lj.: The Differences in Approaching Management by Mangers of Different Gender -an Example from Serbia, African Journal of Business Management, 2011, Vol. 5, No 26, ISSN 1993-8233
10. Grubić-Nešić L., Duđak Lj.: Ljudski resursi i razvoj industrijskog inženjerstva, Beograd, Ekonomski institut, 2011, str. 153-166, ISBN 978-86-7329-086-7

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	1			
Current projects :	Domestic :	1	International :	0

Science, arts and professional qualifications

Name and last name:	Filipović V. Jovan
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Organizational Sciences - Beograd
	01.10.2000
Scientific or art field:	Quality, Effectiveness and Logistics

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Organizational Sciences - Beograd	Quality, Effectiveness and Logistics
Education Specialist Thesis	2011	University of Ljubljana - Ljubljana	Engineering Management
PhD thesis	1994		Mechanical Engineering
Magister thesis	1990	Faculty of Technical Sciences - Novi Sad	Machine Tools, Flexible Technological Systems and Automatization Processes Design
Bachelor's thesis	1986	Faculty of Mechanical Engineering - Beograd	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IMDR74	Selected Topics in Quality Management and Logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
2.	IMDR79	Selected topics in quality engineering and logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Filipović, J., “Management of the Serbian Diaspora Virtual University as a Complex Organization”-Lambert Academic Publishing, Germany, 2012
2.	Filipović, J., Devjak, S. and Putnik, G. „Knowledge Based Economy: The Role of Expert Diaspora”, Panoeconomicus, Vol. 59, pp. 369-386, 2012, SSCI, DOI:10.2298/PAN1203369F, (IF2011=0.396)
3.	Milunovic, S. and Filipovic, J. “Methodology for Quality Management of Projects in Manufacturing Industries”, Total Quality Management and Business Excellence, DOI: 10.1080/14783363.2012.728851, (IF 2011=0,589)
4.	G. Pejovic, J. Filipovic and Lj. Tasic, “How to Remove Barriers to Medicines Trade in Emerging Economies: the Role of Medicines Regulatory Authority in Serbia”, Accreditation and Quality Assurance 2011 16 (4-5):253-261, doi: 10.1007/s00769-010-0749-7, IF2011= 1,036
5.	Popović, F. J., Filipović, V. J. and Božanić, V. N., „Paradigm Shift Needed – Municipal Solid Waste Management in Belgrade, Serbia“ Chemical Industry, SCIe, doi:10.2298/HEMIND120620087P, IF 2012=0,205
6.	J. Filipovic, R. Viskanta and F.P.Incropera, 1994., "An Analysis of Subcooled Turbulent Film Boiling on a Moving Isothermal Surface", Int.J. Heat Mass Transfer, Vol. 37, No. 18., pp. 2661-2673., vrhunski medjunarodni
7.	J. Filipovic, R.Viskanta and F.P.Incropera, 1993, "Similarity Solution for Laminar Film Boiling Over a Moving Isothermal Surface", Int.J. Heat Mass Transfer, Vol.36, No.12, pp. 2957-2963
8.	J. Filipovic, R.Viskanta and F.P.Incropera, 1994, "Cooling of a Moving Steel Strip by an Array of Round Jets", Steel Research, Vol.65, pp. 541-547
9.	J. Filipovic, R.Viskanta, F.P.Incropera and T.A.Veslocki, 1991, "Thermal Behavior of a Moving Steel Strip by an Array of Planar Water Jets", Steel Research, Vol.63, pp. 438-446
10.	J. Filipovic, R.Viskanta and F.P.Incropera, 1991, "A Parametric Study of the Accelerated Cooling of Steel Strip", Steel Research, Vol.63, pp.496-499

Summary data for teacher's scientific or art and professional activity:

Quotation total :	63			
Total of SCI(SSCI) list papers :	9			
Current projects :	Domestic :	3	International :	3

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Folić J. Radomir
Academic title:	Emeritus Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.03.1980
Scientific or art field:	Constructions in Civil Engineering

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Constructions in Civil Engineering
PhD thesis	1983	Faculty of Civil Engineering - Beograd	Theory of Construction
Magister thesis	1974	Faculty of Civil Engineering - Zagreb	Theory of Construction
Bachelor's thesis	1963	Faculty of Civil Engineering - Beograd	Constructions in Civil Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	A002S	Scientific Research Method	(A00) Architecture, Specialised Academic Studies (E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
2.	GG505	Concrete Bridges	(G00) Civil Engineering, Master Academic Studies
3.	GS015	Scientific Research Method	(G10) Energy Efficiency in Buildings, Specialised Academic Studies
4.	A120S	Proces, principi i tehnike naučnog istraživanja-odabrana poglavlja	(A00) Architecture, Specialised Academic Studies
5.	GG531	Odabrana poglavlja zidanih konstrukcija	(G00) Civil Engineering, Master Academic Studies
6.	DGI002	Selected Chapters in Engineering Geodesy	(GI0) Geodesy and Geomatics, Doctoral Academic Studies
7.	DZ001	Scientific Research Method	(A00) Architecture, Doctoral Academic Studies (AS0) Scenic Design, Doctoral Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
8.	A120	Proces, principi i tehnike naučnog istraživanja - odabrana poglavlja(uneti naziv na engleskom)	(A00) Architecture, Doctoral Academic Studies
9.	GD027	Process, principles and techniques of scientific research - selected chapters	(G00) Civil Engineering, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

Representative refferences (minimum 5, not more than 10)

1.	Folić, R. (1983): Spojevi i veze montažnih betonskih zgrada. U knjizi Montažni građevinski objekti, (Ed. B. Žeželj, A.Flašar) Ekonomika, Beograd, str. 117-167. (9 autorskih tabaka)
2.	Folić, R. (1983): Statika konstrukcija - Zbirka rešenih zadataka. FTN IIG, Novi Sad, str. 1-486. II izdanje (1987). III izdanje Građevinska knjiga, Beograd (1991).
3.	Folić, R., Tatomirović, M. (1999): Spregnute betonske konstrukcije-I deo. Građevinski kalendar, 1999. str. 289-386; II deo, Građevinski kalendar, 2001, str. 217-290
4.	Folić, R. (1991): Classification of damage and its causes as applied to precast concrete buildings. Material and Structures. RILEM - Journal, Chapman & Hall, Vol. 24, pp. 276-285.
5.	Folić, R., Ivanov, D. (1991): In situ behaviour of concrete structures deterioration of concrete, influence of earthquake and a fire in Diagnosis of Concrete Structures - State of the Art Report, Ed. by T. Javor, Expertcentrum, Bratislava, pp. 135-146.
6.	Folić, R. (1985): Analiza aktivne širine ploče i graničnih stanja kod elemenata od armiranog i prethodno napregnutog betona. FTN IIG Posebno izdanje 7, Novi Sad, str. 1-193.
7.	Folić, R., Radonjanin, V. (1998): Experimental research on polymer modified concrete, Materials Journal, ACI, VOL. 95 No. 4, July/August 1998, pp.463-470.
8.	Folić, R. (1991): A classification of damage to concrete buildings in earthquakes, illustrated by examples. Material and Structures, RILEM - Journal, Chapman & Hall, Vol. 24, pp. 286-292.
9.	Javor, T., Naus, D.J., Folić, R., Zakić, B.: (1992): Diagnosis of Concrete Structures. RILEM - Journal Materials and Structures, Chapman & Hall, Vol. 25, pp. 437-440.
10.	Folić, R., Radonjanin, V. (1998): Experimental research on polymer modified concrete, Materials Journal, ACI, VOL. 95 No. 4, July/August 1998, pp.463-470.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	11			
Total of SCI(SSCI) list papers :	8			
Current projects :	Domestic :	2	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Gilezan K. Silvia
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.04.1984
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2005	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	1993	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	1988	Faculty of Mathematics - Beograd	Mathematical Sciences
Bachelor's thesis	1981	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	GH404	Mathematical Statistics	(G00) Civil Engineering, Master Academic Studies (G00) Civil Engineering, Undergraduate Academic Studies
2.	GI303B	Probability and Mathematical Statistics	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
3.	IAM003	Formal Mathematical Models	(F10) Engineering Animation, Undergraduate Academic Studies
4.	S011	Mathematics 1	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
5.	Z203	Statistical Methods	(Z01) Safety at Work, Undergraduate Academic Studies (ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
6.	IM1012	Probability and Statistics	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
7.	0M506	Semantics of Programming Languages	(OM1) Mathematics in Engineering, Master Academic Studies
8.	0M507	Logic in Computer Science	(OM1) Mathematics in Engineering, Master Academic Studies
9.	0M513	Introduction to Functional Programming Languages	(OM1) Mathematics in Engineering, Master Academic Studies
10.	0ML506	Semantics of programming languages	(OM1) Mathematics in Engineering, Master Academic Studies
11.	0ML507	Logic in computer science	(OM1) Mathematics in Engineering, Master Academic Studies
12.	0ML513	Introduction to Functional Programming Languages	(OM1) Mathematics in Engineering, Master Academic Studies
13.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
14.	GH404	Mathematical Statistics	(G00) Civil Engineering, Master Academic Studies (G00) Civil Engineering, Undergraduate Academic Studies
15.	SD0M06	Logic in Computer Science	(GI0) Geodesy and Geomatics, Specialised Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
16.	MPK001	Statistical and Numerical Methods	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies
17.	D0M05	Semantics of Programming Languages	(OM1) Mathematics in Engineering, Doctoral Academic Studies
18.	D0M06	Logic in Computer Science	(OM1) Mathematics in Engineering, Doctoral Academic Studies
19.	D0M11	Models of Computation	(OM1) Mathematics in Engineering, Doctoral Academic Studies
20.	D0M12	Introduction to Functional Programming Languages	(OM1) Mathematics in Engineering, Doctoral Academic Studies
21.	D0M13	Theory of Mobile Processes	(OM1) Mathematics in Engineering, Doctoral Academic Studies
22.	D0M14	Process Algebra	(OM1) Mathematics in Engineering, Doctoral Academic Studies
23.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
24.	AID05	Theory of Mobile Processes	(F20) Engineering Animation, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	"Inhabitation in lambda calculus with intersection and union types", Journal of Logic and Computation 6 (1993) 671-685, Oxford University Press
2.	"Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: extending the Coppo-Dezani heritage, (sa D.Dougherty, P.Lescanne) Theoretical Computer Science 2007
3.	"Separating Points by Parallel Hyperplanes " (sa J. Pantovic, J. Zunic), IEEE Transactions of Neural Networks 18(5) (2007) 1356-1363
4.	"Lambda terms for natural deduction, sequent calculus and cut elimination" (sa H.P.Barendregt), Journal of Functional Programming, 10 (2000) 121-134.
5.	"Confluence of untyped lambda calculus via simple types" (with V.Kuncak), ICTCS"01, Lecture Notes in Computer Science 2201, 38-49.
6.	"Full intersection types and topologies in lambda calculus", Journal of Computer and System Sciences, 62 (2001) 1-14.
7.	"Behavioural inverse limit lambda models" (sa M. Dezani-Ciancaglini, S. Likavec), Theoretical Computer Science Vol 316/1-3 (2004) 49-74.
8.	"Strong normalization of the classical sequent calculus" (sa D. Dougherty, P. Lescanne, S.Likavec), Lecture Notes in Computer Science 3835 (2005) 169-183.
9.	"Security types for dynamic web data" (sa M.Dezani-Ciancaglini, J. Pantovic), Trustworthy Global Computing, TGC"06, Lecture Notes in Computer Science 4661 (2007) 263-280.
10.	Zbirka rešenih zadataka iz statistike (sa Z.Lužanin, Z.Ovcin, Lj.Nedović, T.Grbić, B.Mihailović) 2005

Summary data for teacher's scientific or art and professional activity:

Quotation total :	325

Total of SCI(SSCI) list papers :	17			
Current projects :	Domestic :	2	International :	4

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Gradojević J. Nikola
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Industrial Engineering and Engineering Management

Academic carieer	Year	Institution	Field
Academic title election:	2007		Industrial Engineering and Engineering Management
PhD thesis	2003	Sauder School of Bussiness,University of British Columbia - Vancouver	Economics
Magister thesis	1998	Central European University, Budapest - Budimpešta	Economics
Bachelor's thesis	1996	Faculty of Technical Sciences - Novi Sad	Electrical and Computer Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EP005	E-bankarstvo	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
2.	IMDS35	Selected Chapters in Investment Management	(I22) Engineering Management, Specialised Academic Studies
3.	IMDS48	Advanced Risk Management	(I22) Engineering Management, Specialised Academic Studies
4.	MBA606	Internet marketing	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
5.	IM2407	International business and finance	(I20) Engineering Management, Master Academic Studies
6.	IM2413	Enterprise portfolio management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
7.	IMDR35	Selected Chapters in Investment Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
8.	IMDR48	Advanced Risk Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Nikola Gradojevic, Ramazan Gençay. "Fuzzy logic, trading uncertainty and technical trading", Journal of Banking & Finance, Volume 37, Issue 2, February 2013, Pages 578–586.
2.	Lento, C., Gradojevic, N. (2013). “The Effectiveness of Option Pricing Models During Financial Crises.” In: Wehn, C.S., Hoppe, C., Gregoriou, G.N. (Eds)., Rethinking Valuation and Pricing Models: Lessons Learned from the Crisis and Future Challenges. Academic Press, Elsevier Inc., pp. 1–11.
3.	Dragan Kukolj, Nikola Gradojevic and Camillo Lento. “Improving Non-parametric Option Pricing during the Financial Crisis”, Computational Intelligence for Financial Engineering & Economics (CIFEr), 2012 IEEE Conference, pp. 93-99.
4.	Nikola Gradojevic, “Frequency Domain Analysis of Foreign Exchange Order Flows”, Economics Letters 115, 73-76 (2012).
5.	Nikola Gradojevic and Ramo Gençay, “Financial Applications of Non-extensive Entropy”, IEEE Signal Processing Magazine 28 (5), 116-141 (2011).
6.	Nikola Gradojevic, Dragan Kukolj, "Parametric option pricing: A divide-and-conquer approach", Physica D: Nonlinear Phenomena, Volume 240, Issue 19, 15 September 2011, pp. 1528–1535.
7.	Ramo Gençay and Nikola Gradojevic, “Errors-in-Variables Estimation with No Instruments”, Journal of Statistical Computation and Simulation 81 (11), 1545-1564 (2011).
8.	Nikola Gradojevic, Ramo Gençay, “Crash of "87 - Was it Expected? Aggregate Market Fears and Long Range Dependence”, Journal of Empirical Finance 17 (2), 270-282 (2010).
9.	Ramo Gençay, Nikola Gradojevic, Faruk Selcuk and Brandon Whitcher, “Asymmetry of Information Flow between Volatilities Across Time Scales”, Quantitative Finance 10 (8), 895-915 (2010).
10.	Nikola Gradojevic, V. Djakovic and G. Andjelic, "Random Walk Theory and Exchange Rate Dynamics in Transition Economies" , Panoeconomicus 57 (3), 303-320 (2010).

Summary data for teacher's scientific or art and professional activity:

Quotation total :	42
Total of SCI(SSCI) list papers :	14

Current projects :	Domestic :	0	International :	3

Science, arts and professional qualifications

Name and last name:	Grbić P. Tatjana
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.12.1995
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2009	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	2008	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	1999	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1993	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E135	Probability, Statistics and Stochastic Processes	(MR0) Measurement and Control Engineering, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
2.	E212	Mathematical Analysis 1	(E20) Computing and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
3.	GI303B	Probability and Mathematical Statistics	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
4.	Z104	Mathematics 1	(Z01) Safety at Work, Undergraduate Academic Studies (ZC0) Clean Energy Technologies, Undergraduate Academic Studies (ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
5.	Z203	Statistical Methods	(Z01) Safety at Work, Undergraduate Academic Studies (ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
6.	BMI91	Mathematics 1	(BM0) Biomedical Engineering, Undergraduate Academic Studies
7.	BMI92	Mathematics 2	(BM0) Biomedical Engineering, Undergraduate Academic Studies
8.	IA001	Algebra	(F10) Engineering Animation, Undergraduate Academic Studies
9.	IA002	Mathematical Analysis	(F10) Engineering Animation, Undergraduate Academic Studies
10.	P216	Numerical Analysis	(P00) Production Engineering, Undergraduate Academic Studies
11.	S01361	Business decision making	(S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
12.	0M505	Stochastic Processes	(OM1) Mathematics in Engineering, Master Academic Studies
13.	0ML505	Stochastic Processes	(OM1) Mathematics in Engineering, Master Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
14.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
15.	ZR503	Statistical Advanced Models	(Z01) Safety at Work, Master Academic Studies
16.	MPK001	Statistical and Numerical Methods	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies
17.	SDOM30	Probability, Statistics and Theory of Engineering Experiment	(Z00) Environmental Engineering, Specialised Academic Studies
18.	D0M01	Functional Analysis 1	(OM1) Mathematics in Engineering, Doctoral Academic Studies
19.	D0M07	Mathematical Foundations of Fuzzy Systems	(OM1) Mathematics in Engineering, Doctoral Academic Studies
20.	D0M19	Functional Analysis 2	(OM1) Mathematics in Engineering, Doctoral Academic Studies
21.	D0M21	Fuzzy Systems and Their Applications	(OM1) Mathematics in Engineering, Doctoral Academic Studies
22.	D0M50	Fuzzy Measures and Integrals	(OM1) Mathematics in Engineering, Doctoral Academic Studies
23.	D0M51	Large Deviations Principles	(OM1) Mathematics in Engineering, Doctoral Academic Studies
24.	D0M52	Random Sets	(OM1) Mathematics in Engineering, Doctoral Academic Studies
25.	D0M53	Statistical Processing of Fuzzy Data	(OM1) Mathematics in Engineering, Doctoral Academic Studies
26.	DOM30	Probability, Statistics and Theory of Engineering Experiment	(M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
27.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Ralević, N.M., Nedović, Lj., Grbić, T., :"The pseudo-linear superposition principle for nonlinear partial differential equations and representation of their solution by the pseudo-integral", Fuzzy sets and systems, 2005, No.155, 89-101

Representative refferences (minimum 5, not more than 10)

2.	Nedović, Lj., Ralević, N. M., Grbić, T.,: " Large deviation principle with generated pseudo measures", Fuzzy sets and systems, 2005, No. 105, 65-76
3.	Štajner-Papuga, I., Grbić, T., Dankova, M., "Pseud-Riemann-Stieltjes integral ", Information Sciences 179, 2009, 2923-2933
4.	M. Štrboja, T. Grbić, I. Štajner-Papuga, G. Grujić, S. Medić, Jensen and Chebyshev inequalities for pseudo-integrals of set-valued functions, FSS, doi:10.101016/j.fss.2012.07.011
5.	Grbić, T., Pap, E., : "Generalization Of Portamnteau theorem with respect to the pseudo-weak convergence of random closed sets", Theory of Probability and its Applications, 2009, 97-115
6.	T. Grbić, I. Štajner-Papuga, M. Štrboja, an approach to pseudo-integration of set-valued functions, Information Sciences 181 (2011), 2278-2292
7.	T. Grbić, S. Medić, I. Štajner-Papuga, T. Došenović, Inequalities of Jensen and Chebyshev type for interval-valued measures based on pseudo-integrals. In: Intelligent Systems: Models and Applications, E. Pap, Ed., Springer-Verlag, pp 23-41, DOI:10.1007/978-3-642-33959-2_2
8.	Štajner-Papuga, I., Grbić, T., Dankova, M., "Riemann-Stieltjes type integral based on generated pseudo-operations", NS J. Mathe., Vol. 36, No. 2, 111-124
9.	Nedović, Lj., Grbić, T., "The pseudo-probability", Journal of Electrical Engineering, 2002, Vol. 53, No. 12/s, 27-30
10.	Mihailović, B., Nedović, T., Grbić, T., "The induced Sugeno integral-based operator w.r.t. bi-fuzzy measures", Journal of Electrical engineering, Vol. 54, No. 12/s, 76-79

Summary data for teacher's scientific or art and professional activity:

Quotation total :	17			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	2	International :	0

Science, arts and professional qualifications

Name and last name:	Grubić-Nešić S. Leposava
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2007
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2003	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2002	Faculty of Entrepreneurial Management - Novi Sad	Engineering Management
Bachelor's thesis	1981	Faculty of Philosophy - Beograd	Psychological Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II934	Psichology of Work	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
2.	IM1025	Human resources management	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1906	Work motivation	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
4.	IM1916	Industrial psychology	(I20) Engineering Management, Undergraduate Academic Studies
5.	S0I322	Human Resources Management	(S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
6.	I076/S	Leadership and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
7.	I935/S	Motivating Employees	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
8.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
9.	IMDS51	Organizatonal behaviour	(I22) Engineering Management, Specialised Academic Studies
10.	MBA308	Business communication	(IB0) Engineering Management - MBA, Specialised Professional Studies
11.	MBA309	Human Resource Management in Knowledge Economy	(IB0) Engineering Management - MBA, Specialised Professional Studies
12.	MBA513	leadership development and teamworking	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
13.	MBA515	decision macing and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
14.	MBA522	Lobbying, presentation and negotiation skills	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
15.	MBA524	interculture business communications	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
16.	RPR013	Management of Human Resources	(RPR) Regional Development Planning and Management, Master Academic Studies
17.	IM2907	Leadership	(I20) Engineering Management, Master Academic Studies
18.	IM2913	Teamwork	(I20) Engineering Management, Master Academic Studies
19.	IMDS77	Selected Chapters from Human Resource Management	(I22) Engineering Management, Specialised Academic Studies
20.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
21.	IMDR51	Organisational Behavior	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
22.	IMDR77	Selected Chapters from Human Resource Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Razvoj ljudskih resursa, AB Print,Novi Sad, 2005.
2. Znati biti lider, AB print, Novi Sad, 2008.
3. Cabrilo, S.; Grubic-Nesic, L.(2012). „The role of creativity, innovation and invention in knowledge management“, in Buckley, S. and Jakovljevic, M (eds.) Knowledge Management Innovations for Interdisciplinary Education: Organisational Applications, Hershey, USA: IGI Global
4. Mitrovic,S.,Milisavljevic,S.,Cosic,I.,Lekovic,B.,Grubic-Nesic,L.,Ivanisevic,A., Changes in leadership styles in a transitional economy: A Serbian case study, African Journal of Business Management, Vol. 5(9), pp. 3563-3569, 2011. ISSN 1993-8233
5. Ratkovic-Njegovan, B.,Vukadinovic, M.,Grubic-Nesic, L.,Characteristics and Types of Authority: the Attitudes of Young People. A Case Study, Sociologija, 2011, Vol. 43(6), pp.657-673.
6. Kirin, S., Grubic-Nesic, L., Cosic, I. (2010). Increasing a Large Petrochemical Company Efficiency by Improvement of Decision Making Process, Hemijska Industrija, ISSN 0367-598X, doi: 102298/hemind 100710048k, vol.64 broj 5, str.465-472
7. Kolaric, B., Grubic-Nesic, L., Radojcic, S., (2011). The challenges of the customer services for modern market requests: a case study of Telecom Serbia, African journal of business management, ISSN 1993-8233, vol 5(1), pp. 156-167
8. Kirin S., Sedmak A., Grubic-Nesic L., Cosic I., (2012). Project risk management in complex petrochemical system, Hemijska industrija, 2012, pp. 52-52, ISSN 0354-7531, UDK: doi:10.2298/HEMIND110709052K
9. Grubic-Nesic, L., Vranjes, S., Ratkovic-Njegovan, B., Mitrovic S.: Atitudes of the employees about the organizational restructuring: a sample of organizations in Serbia, Metalurgia international, 2012, Vol. 17, No 12, ISSN 1582-2214
10. Konja, V., Grubic-Nesic, L., Mitrovic, S., (2012). Leader-member exchange: a short case study from a Serbian company, Metalurgia international, 2012, Vol. 17, No. 11, pp. 146-153, ISSN 1582-2214

Summary data for teacher's scientific or art and professional activity:

Quotation total :	6			
Total of SCI(SSCI) list papers :	8			
Current projects :	Domestic :	2	International :	2

Science, arts and professional qualifications

Name and last name:	Gvozdenac D. Dušan
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.06.1973
Scientific or art field:	Thermal Energetics and Thermotechnics

Academic carieer	Year	Institution	Field
Academic title election:	1993	Faculty of Technical Sciences - Novi Sad	Thermal Energetics and Thermotechnics
PhD thesis	1981	Faculty of Mechanical Engineering - Beograd	Thermal Energetics and Thermotechnics
Magister thesis	1978	Faculty of Technical Sciences - Novi Sad	Thermal Energetics and Thermotechnics
Bachelor's thesis	1973	Faculty of Technical Sciences - Novi Sad	Thermal Energetics and Thermotechnics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EOS38	Energetski menadžment	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
2.	M119	Energy Transformations	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
3.	M222A	Energy System Engineering	(M30) Energy and Process Engineering, Undergraduate Academic Studies
4.	M3311	Renewable Energy Sources	(M30) Energy and Process Engineering, Undergraduate Academic Studies (ZC0) Clean Energy Technologies, Undergraduate Academic Studies
5.	M3501	Refrigeration Devices	(M30) Energy and Process Engineering, Undergraduate Academic Studies
6.	Z206	Alternative Power Engineering	(Z20) Environmental Engineering, Undergraduate Academic Studies
7.	Z206A	Alternative Energy Sources	(Z01) Safety at Work, Undergraduate Academic Studies
8.	Z206	Alternativna energetika(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
9.	E2313	Fundamentals of Process and Energy Engineering	(E20) Computing and Control Engineering, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
10.	II1044	Energy flows and energy efficiency	(I10) Industrial Engineering, Undergraduate Academic Studies
11.	M211	Measurement and Regulation	(M30) Energy and Process Engineering, Undergraduate Academic Studies (ZC0) Clean Energy Technologies, Undergraduate Academic Studies
12.	M3031	Engineering Calculations of Energy Technologies Apparatus and Equipment	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
13.	M3494	Energy efficiency	(M30) Energy and Process Engineering, Undergraduate Academic Studies (ZC0) Clean Energy Technologies, Undergraduate Academic Studies
14.	I939	Merenje, nadzor i upravljanje	(M50) Energy Management, Master Academic Studies
15.	IMDS78	Odabrana poglavlja iz energetskog menadžmenta(uneti naziv na engleskom)	(I22) Engineering Management, Specialised Academic Studies
16.	M3503	Dinamika i modeliranje termoenergetskih postrojenja(uneti naziv na engleskom)	(M30) Energy and Process Engineering, Master Academic Studies
17.	M3M07	Energy storage	(ZC0) Clean Energy Technologies, Master Academic Studies
18.	M5022	Renewable energy sources	(M50) Energy Management, Master Academic Studies
19.	SZSP24	Savremeni principi energetskog menadžmenta	(Z00) Environmental Engineering, Specialised Academic Studies
20.	DM216	Energy Systems	(M00) Mechanical Engineering, Doctoral Academic Studies
21.	DM217	Energy Management in Idustry	(M00) Mechanical Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
22.	DM218	Contemporary Energy Technologies	(M00) Mechanical Engineering, Doctoral Academic Studies
23.	DM219	Energy Politics	(M00) Mechanical Engineering, Doctoral Academic Studies
24.	DM302	Engineering Experimental Methods	(H00) Mechatronics, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies
25.	DM309	Energy Management Methods	(M00) Mechanical Engineering, Doctoral Academic Studies
26.	DM332	Energy Management in Buildings	(M00) Mechanical Engineering, Doctoral Academic Studies
27.	DM333	Renewable Energy Resoruces	(M00) Mechanical Engineering, Doctoral Academic Studies
28.	ZSP24	Modern Principles of Energy Management	(Z00) Environmental Engineering, Doctoral Academic Studies
29.	IMDR78	Odabrana poglavlja iz energetskog menadžmenta(uneti naziv na engleskom)	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Energy Efficiency in Food Processing Industry – East European Experience, edited by D. Gvozdenac, UNDP/UNIDO Project DP/RER/83/003, Novi Sad, pp. 123, 1991.
2.	Conterporary problems in Power Engineering (monograph), Novi Sad/Thesaloniki, Gvozdenac D, Xypteras J, Dimić M. 1996.
3.	Measurement and regulation (Selected chapters for operators of large power plants), Institute of energy and process engineering, Novi Sad, Gvozdenac, D, Pešenjanski, I,1980. (in Serbian).
4.	Measurement and Regulation in Thermal Engineering, Faculty of Technical Sciences, Gvozdenac, D, Novi Sad, 2000. (in Serbian).
5.	Bilansiranje energetskih tokova, Pokrajinski centar za energetku efikasnost, Gvozdenac, D., Marić, M., Petrović, J., Novi Sad, 2006.
6.	Gvozdenac D, Menke C, Vallikul P, Petrovic J, Gvozdenac B: Assessment of potential for natural gas-based cogeneration in Thailand, Energy, Volume 34, Issue 4, 2009, pp 465-475
7.	A Mathematical Model for Heat Transfer in Combustion Chambers of Steam Generators, Gulič, M, Gvozdenac, D, Transactions of the ASME Journal of Engineering for Power, Vol. 103, 1981, pp. 545 – 551.
8.	Somcharoenwattana W, Menke C, Kamolpus D, Gvozdenac D: Study of Operational Parameters Improvement of Natural-Gas Cogeneration Plant in Public Buildings in Thailand, Energy and Buildings, Vol. 43, Issue 4, April, 2011. p. 925-934
9.	Two-pass counter cross-flow heat exchangers with both fluids unmixed throughout, Gvozdenac, D, Waerme - und Stoffuebertragung, Vol. 20, 1986, pp. 151 – 161.
10.	Analytical Solution of the Transient Response of Gas-to-Gas Cross-flow Heat Exchanger With Both Fluids Unmixed, Gvozdenac, D.D, ASME Journal of Heat Transfer, Vol. 108, 1986, pp. 722-727.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	71			
Total of SCI(SSCI) list papers :	26			
Current projects :	Domestic :	2	International :	1

Science, arts and professional qualifications

Name and last name:	Herakovič S. Niko
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	University of Ljubljana - Ljubljana
	01.01.2007
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	2012		Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	1995	University of Ljubljana - Ljubljana	Mechanical Engineering
Magister thesis	1991	University of Ljubljana - Ljubljana	Mechanical Engineering
Bachelor's thesis	1988	University of Ljubljana - Ljubljana	Mechanization and Constructional Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EOS19	Dismantling and recycling technologies	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
2.	H105	Fundamentals in Computer science	(H00) Mechatronics, Undergraduate Academic Studies
3.	H1410	Programming and application of programmable logic controllers	(H00) Mechatronics, Undergraduate Academic Studies
4.	BMI106	Rehabilitation devices and systems	(BM0) Biomedical Engineering, Undergraduate Academic Studies
5.	IM1116	Work Study and Ergonomics	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
6.	IMDS56	Product traceability during the lifetime	(I12) Industrial Engineering, Specialised Academic Studies
7.	IMDS57	Strategic Planning and Designing Procedures and Systems at the End of Product Lifecycle	(I12) Industrial Engineering, Specialised Academic Studies
8.	IMDS93	Virtual Enterprises and Collaborative Systems	(I22) Engineering Management, Specialised Academic Studies
9.	H799	Fieldbuses and protocols	(H00) Mechatronics, Master Academic Studies
10.	H828	Advanced robotics	(H00) Mechatronics, Master Academic Studies
11.	I907	Automated Assembly Systems for High Accuracy	(H00) Mechatronics, Master Academic Studies (PM0) Production Engineering, Master Academic Studies
12.	IIDS6	Selected chapters in automation	(I12) Industrial Engineering, Specialised Academic Studies
13.	IM2102	Manufacturing strategy (KAIZEN, LEAN, KANBAN, EFPS)	(I10) Industrial Engineering, Master Academic Studies (M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
14.	IM2124	Production and Service Systems	(H00) Mechatronics, Master Academic Studies (M50) Energy Management, Master Academic Studies
15.	IMDR56	Traceability of Product Lifecycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
16.	IMDR93	Virtual Enterprises and Collaborative Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	DEBEVEC, Mihael, HERAKOVIČ Niko. Management Of Resources In Small And Medium-Sized Production Enterprises. Iranian Journal of Science and Technology. 51/79. (Article will be published in october 2010 – Enclosure 6 – Certificate of the paper received for publication)
2.	MERWE, Jacob D. van der, MINARIK, Martin, BEROVIČ, Marin, HERAKOVIČ, Niko. Heat transfer in citric acid production with axial and radial flow impellers. Acta chim. slov.. [Tiskana izd.], 2010, vol. 57, no. 1, str. 150-156. http://acta.chemsoc.si/57/57-1-150.pdf. [COBISS.SI-ID 33809925]
3.	HERAKOVIČ, Niko, ŠIMIC, Marko, TRDIČ, Francelj, SKVARČ, Jure. A machine-vision system for automated quality control of welded rings. Mach. vis. appl., 2010, 15 str., doi: 10.1007/s00138-010-0293-9. ISSN 0932-8092. [COBISS.SI-ID 11512091], [JCR], 126/245
4.	HERAKOVIČ, Niko. Flow-force analysis in a hydraulic sliding-spool valve. Strojarstvo, 2007, letn. 49, št. 3, str. 117-126. [COBISS.SI-ID 10449691]

Representative refferences (minimum 5, not more than 10)

5.	HERAKOVIČ, Niko. Računalniški in strojni vid v robotizirani montaži = Computer and machine vision in robot-based assembly. Stroj. vestn., 2007, letn. 53, št. 12, str. 858-873. ISSN 0039-2480. [COBISS.SI-ID 10378267], [JCR, WoS], 100/107
6.	HERAKOVIČ, Niko, NOE, Dragica. Analiza delovanja pnevmatičnega ventila s predkrmilnim piezoventilom = Analysis of the operation of pilot-stage piezo-actuator valves. Stroj. vestn., 2006, letn. 52, št. 12, str. 835-851. [COBISS.SI-ID 9821723]
7.	HERAKOVIČ, Niko, DUHOVNIK, Jože, NOE, Dragica. Sila trenja v pnevmatičnem valju = Friction force in the pneumatic cylinder. Stroj. vestn., okt.-dec. 1992, let. 38, št. 10/12, str. 279-288, ilustr. [COBISS.SI-ID 62843136]
8.	POPOVIČ, Milan, KLUN, Boris, HERAKOVIČ, Niko, NOE, Dragica. Fractures of the skull base in the fossa media - a biomechanical experimental study. Period. biol., 1994, vol. 96, no.1, str. 41-44. [COBISS.SI-ID 2621979]
9.	HERAKOVIČ, Niko, HLADNIK, Marko. Apparatus for retaining a package of laminations of an electromagnetic core in a device for the production thereof : WO mednarodni PCT/SI2009/000060. Ženeva: WIPO, 2009. 34 f., ilustr. [COBISS.SI-ID 11303963]
10.	NOE, Dragica, PERME, Tomaž, HERAKOVIČ, Niko. Orodja za načrtovanje in analizo delovanja proizvodnih sistemov LASIMCO - simulacija v montaži, SIMPLE++ - simulacija poslovnih in proizvodnih sistemov, DSHplus – simulacija delovanja hidravličnih sistemov. V: KUZMAN, Karl (ur.). Dnevi slovenskega proizvodnega inženirstva, Celje, 3.-5. junij 1998. Zbornik posvetovanja. Ljubljana: Fakulteta za strojništvo, 1998, str. 111-116. ISBN 961-90401-3-9. [COBISS-ID 2658331]

Summary data for teacher's scientific or art and professional activity:

Quotation total :	11			
Total of SCI(SSCI) list papers :	13			
Current projects :	Domestic :	1	International :	3

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Ivandić I. Željko
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	2002	Faculty of Mechanical Engineering and Naval Architecture - Zagreb	Mechanical Engineering
Magister thesis	1996	Faculty of Mechanical Engineering and Naval Architecture - Zagreb	Mechanical Engineering
Bachelor's thesis	1990	Mechanical Engineering Faculty - Slavonski Brod - Slavonski Brod	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H102	Fundamentals in Product Development	(H00) Mechatronics, Undergraduate Academic Studies
2.	H105	Fundamentals in Computer science	(H00) Mechatronics, Undergraduate Academic Studies
3.	H109	Fundamentals in Programming	(H00) Mechatronics, Undergraduate Academic Studies
4.	H1409	Intelligent Systems	(H00) Mechatronics, Undergraduate Academic Studies
5.	H1410	Programming and application of programmable logic controllers	(H00) Mechatronics, Undergraduate Academic Studies
6.	H1501A	Systems for Survailance and Visualisation of Process	(H00) Mechatronics, Undergraduate Academic Studies
7.	H308	Industrial Robotics	(H00) Mechatronics, Undergraduate Academic Studies
8.	II1015	Programmable Logic Controllers (PLC)	(I10) Industrial Engineering, Undergraduate Academic Studies
9.	II1048	Artificial intelligence in engineering	(I10) Industrial Engineering, Undergraduate Academic Studies
10.	H301	System Modeling and Symulation	(H00) Mechatronics, Master Academic Studies
11.	HDOS12	Research in the area of automatic identification technology	(I12) Industrial Engineering, Specialised Academic Studies
12.	HDOS13	Motion control and application of MEMS	(I12) Industrial Engineering, Specialised Academic Studies
13.	HDOS14	Nonindustrial automation	(I12) Industrial Engineering, Specialised Academic Studies
14.	PLM09	Systems and Devices for Tracking Products Through Life Cycle	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
15.	NIT06	Advanced Technologies for Manufacturing Support	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
16.	H845	Motion control	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
17.	I903	Application of microelectromechanical systems	(I10) Industrial Engineering, Master Academic Studies
18.	IIDS6	Selected chapters in automation	(I12) Industrial Engineering, Specialised Academic Studies
19.	IM2516	Artificial Intelligence in Engineering	(I20) Engineering Management, Master Academic Studies
20.	IM2721	Systems for detection, alarming and warning	(I20) Engineering Management, Master Academic Studies
21.	HDOK12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
22.	HDOK13	Motion control and the application of MEMS	(H00) Mechatronics, Doctoral Academic Studies
23.	HDOK14	Non-industrial Automation	(H00) Mechatronics, Doctoral Academic Studies
24.	HDOK-3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
25.	HDOKL3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
26.	HDOL12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
27.	HDOL13	Motion controla and application of MEMS	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
28.	HDOL14	Nonindustrial automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Brillová, K., Ohlídal, M., Valíček, J., Hloch, S., Kozak, D., Ivandić, Z. Evaluation of abrasive waterjet produced titan surfaces topography by spectral analysis techniques (2012) Metalurgija, 51 (1), pp. 39-42.
2.	Kozak, D., Ivandić, Z., Kontajić, P. Determination of the critical pressure for a hot-water pipe with a corrosion defect [Določitev kritičnega pritiska v vročevodni cevi s korozijsko poškodbo] (2010) Materiali in Tehnologije, 44 (6), pp. 385-390.
3.	Balicević, P., Ivandić, Z., Kraljević, D. Temperature transitional phenomena in spherical reservoir wall (2010) Tehnicki Vjesnik, 17 (1), pp. 31-34.
4.	Ivandić, Z., Ergić, T., Kljajin, M. Welding robots kinematic structures evaluation of based on conceptual models using the potential method (2009) Tehnicki Vjesnik, 16 (4), pp. 35-45.
5.	Ergić, T., Ivandić, Ž. Ultra-light telescopic crane/platform mechanisms feature analysis (2009) Tehnicki Vjesnik, 16 (4), pp. 87-91.
6.	Ivandić, Ž., Ergić, T., Kokanović, M. Conceptual model and evaluation of design characteristics in product development (2009) Strojarstvo, 51 (4), pp. 281-291.
7.	Hlaváček, P., Valíček, J., Hloch, S., Greger, M., Foldyna, J., Ivandić, Z., Sitek, L., Kušnerová, M., Zeleňák, M. Measurement of fine grain copper surface texture created by abrasive water jet cutting (2009) Strojarstvo, 51 (4), pp. 273-279.
8.	Radvanská, A., Ergić, T., Ivandić, Ž., Hloch, S., Valicek, J., Mullerova, J. Technical possibilities of noise reduction in material cutting by abrasive water-jet (2009) Strojarstvo, 51 (4), pp. 347-354.
9.	Kušnerová, M., Valíček, J., Hloch, S., Ergić, T., Ivandić, Z. Derivation and measurement of the velocity parameters of hydrodynamics oscillating system (2008) Strojarstvo, 50 (6), pp. 375-379.
10.	Dunder, M., Ivandić, Ž., Samardžić, I. Selection of arc welding parameters of micro alloyed HSLA steel (2008) Metalurgija, 47 (4), pp. 325-330.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	14			
Total of SCI(SSCI) list papers :	13			
Current projects :	Domestic :	1	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Ivanišević V. Andrea
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2005
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2011	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Magister thesis	2008	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	2005	Faculty of Economics - Subotica	Economic Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	F108	Sociology of Culture	(F00) Graphic Engineering and Design, Undergraduate Academic Studies
2.	M317	Economy	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
3.	S002A	Economics	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
4.	II121	Principles of economics	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
5.	II1047	Analysis and calculation of production costs	(I10) Industrial Engineering, Undergraduate Academic Studies
6.	IM1004	Principles of economics	(I20) Engineering Management, Undergraduate Academic Studies (ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
7.	IM1014	Company Economics	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
8.	IM1047	Planning and enterprises performance analysis	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1422	Managing the cost of production	(I20) Engineering Management, Undergraduate Academic Studies
10.	IMDS88	Planning and implementing cost structure of the investment cycle	(I22) Engineering Management, Specialised Academic Studies
11.	Z513A	Economics and the environmental protection	(Z20) Environmental Engineering, Master Academic Studies
12.	Z513	Ekonomija i zaštita životne sredine(uneti naziv na engleskom)	(Z20) Environmental Engineering, Master Academic Studies
13.	IM2122	The rating company profitability	(I20) Engineering Management, Master Academic Studies
14.	IM2415	Investment Environment	(M50) Energy Management, Master Academic Studies (OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
15.	IM2417	Managing individual property	(I20) Engineering Management, Master Academic Studies
16.	IM2421	Manage the budget for development investment	(I20) Engineering Management, Master Academic Studies
17.	IM2425	Economics of the Firm	(M50) Energy Management, Master Academic Studies
18.	IMDR88	Planning and implementing cost structure of the investment cycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

Representative refferences (minimum 5, not more than 10)

1.	Leković B., Ivanišević A., Marić B., Demko-Rihter J.: ASSESSMENT OF THE MOST SIGNIFICANT IMPACTS OF ENVIRONMENT ON THE CHANGES IN COMPANY COST STRUCTURE, Economic Research, 2013
2.	Milovanović Z.N., Knežević D., Ivanišević A., Jocanović M., Mitrović S.: ECONOMICAL EVALUATION OF THE PROJECT ON REPLACEMENT OF HEATING PLANT WITH CO-GENERATION HEAT AND POWER PLANT BY THE END OF 2030., Metalurgia International, 2013, No.4
3.	Marić B., Ivanišević A.: THE EFFECT OF PERMANENT WORKING CAPITAL ON THE QUALITY OF INVESTMENT PROJECTS, Metalurgia International, 2013
4.	Marić B., Ivanišević A., Mitrović S., Sreto A., Mihailo R.: Analysis of internal rate of return on investments: Dynamic and static approach, African Journal of Business Management, 2011, Vol. 5, No 8, pp. 3269-3273, ISSN 1993-8233
5.	Katić I, Ivanišević A., Penezić N., Lalić G., Tasić N.: EFFECTS OF FATIGUE TO OPERATIONAL PRODUCTIVITY WITH EMPLOYEES, Metalurgia International, 2013
6.	Mitrović S., Milisavljević S., Ćosić I., Leković B., Grubić-Nešić L., Ivanišević A.: Change in leadership styles in a transitional economy: A serbian case study, African Journal of Business Management, 2011, Vol. 5, No 9, pp. 3563-3569, ISSN 1993-8233
7.	Alpar Lošonc, Andrea Ivanišević, Slavica Mitrović „ Globalizacija-rešenja i dileme“ Monografija, Fakultet tehničkih nauka, Novic Sad, 2009. (ISBN 978-86-7892-207-7, COBISS.SR-ID 244134407. (1-263)
8.	Lošonc (Losoncz) A., Ivanišević A., Mitrović S.: Strukturalna kriza: forme i uzroci, Novi Sad, Fakultet tehnickih nauka, , 2012, str. 1-232, ISBN 978-86-7892-375-3, UDK: 268964871
9.	Razvoj sistema za planiranje praćenje i uskalđivanje ključnih segmenata poslovanja industrijskog distema u skaldu sa promena u okruženju, Fakultet tehničkih nauka Novi Sad, 2011
10.	Lošonc A., Radivojević R., Ivanišević A., Pejić S.: TOYOTISM AS A BASIS FOR CORPORATE CULTURE AND WORK ORGANIZATIONS, 1st International Scientific Conference on Lean Tehnologies, Novi Sad, Sertember 2012., pp. 100-106

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	3	International :	0

Science, arts and professional qualifications

Name and last name:	Jocanović T. Mitar
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.03.1999
Scientific or art field:	Quality, Effectiveness and Logistics

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
PhD thesis	2010	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
Magister thesis	2006	Faculty of Technical Sciences - Novi Sad	Mechanical Engineering
Bachelor's thesis	1999	Faculty of Technical Sciences - Novi Sad	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H1403	Automation of work processes	(H00) Mechatronics, Undergraduate Academic Studies
2.	H310	Components of technological systems	(H00) Mechatronics, Undergraduate Academic Studies
3.	I401	Tribology	(M30) Energy and Process Engineering, Undergraduate Academic Studies
4.	URZP17	Devices and systems in fire protection	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
5.	URZP40	Stationary Systems for Fire Extinguishing	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
6.	URZP45	Mobile Equipment and Fire Extinguishing Equipment	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
7.	II1011	Automation of work processes 1	(I10) Industrial Engineering, Undergraduate Academic Studies
8.	II1038	Automation of work processes 2	(I10) Industrial Engineering, Undergraduate Academic Studies
9.	II1050	TRIBOLOGY AND LUBRICATION	(I10) Industrial Engineering, Undergraduate Academic Studies
10.	IM1008	Processes and Work Equipment	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
11.	IMDS58	Selected Chapters in Hydraulic Systems	(I12) Industrial Engineering, Specialised Academic Studies
12.	IMDS95	Trends in Customer Relationship Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
13.	ZP507	Design and Maintenance of Stationary Fire Extinguishing Systems	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
14.	ZP512	Protection and Rescue Plans	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
15.	IIDS12	Quality and organizational performance	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
16.	IIDS30	Trends in the environmental management systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
17.	IIDS7	Selected topics in quality engineering and logistics	(I12) Industrial Engineering, Specialised Academic Studies
18.	IMDS74	Selected Topics in Quality Management and Logistics	(I22) Engineering Management, Specialised Academic Studies
19.	IMDR58	Selected Chapters in Hydraulic Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
20.	IMDR94	Trends in the environmental management systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
21.	IMDR95	Trends in Customer Relationship Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
22.	IMDR74	Selected Topics in Quality Management and Logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
23.	IMDR79	Selected topics in quality engineering and logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	IMDR83	Quality abd organisational performance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	V. Savić, D. Knežević, D. Lovrec, M. Jocanović, Velibor Karanović: Determination of Pressure Losses in Hydraulic Pipeline Systems by Considering Temperature and Pressuer, Strojšnik Vestnik-Journal of Mechanical Engineering, 2009, Vol. 55, No. 4, str.237-243, UDK: 621.643, ISSn 0039-2480
2.	M. Jocanović, D. Šević, V. Karanović, I. Beker, S. Dudić: Increased efficiency of hydraulic systems through reliability theory and monitoring of system operating parameters,Strojšnik Vestnik-Journal of Mechanical Engineering, 2012, Vol. 58, No. 4, str.281-288, UDK: 621.643, ISSN 0039-2480
3.	Z.Milovanović, D. Knežević,A. Ivanišević, M. Jocanović, S. Mitrović:ECONOMICAL EVALUATION OF THE PROJECT ON REPLACEMENT OF HEATING PLANT WITH CO-GENERATION HEAT AND POWER PLANT BY THE END OF 2030 , Metalurgia International, 2013, No4,
4.	M. Jocanović, V. Savić, V. Karanović,: MODEL FOR TRANSLATION OF CLASSES OF PURITY OF OILS BETWEEN ISO 4406/99, NAS 1638-01 AND SAE AS 4059: D STANDARDS, 14. Međunarodna naučna konferencija INDUSTRIJSKI SISTEMI - IS"08, Novi Sad: Fakultet tehničkuh nauka - Novi Sad, 2-3 Oktobar, 2008, str. 391- 396, UDK: 685.5 (082), ISBN 978-86-7892-135-3.
5.	M. Jocanović; PRILAZ ISTRAŽIVANJU I DEFINISANJU MODELA ZA PRORAČUN PROTICANJA ČVRSTIH ČESTICA SA ULJNOM MASOM KROZ ZAZORE U FUNKCIJI KONSTRUKCIONO RADNIH PARAMETARA HIDRAULIČNIH KOMPONENATA, Doktorska disertacija
6.	M.Jocanović; RAZVOJ INTEGRALNOG MODELA ZA IZBOR I DIJAGNOSTIKU MINERALNIH HIDRAULIČKIH ULJA; Magistrarski rad iz oblasti problematike vezane za izbor i dijagnostikovanje mineralnih hidrauličkih ulja u hidrauličkim sistemima
7.	M.Jocanović, D.Babić, V.Karanović, R.Geaverts: Industrial Aplication of Automatic Lubrication Systems, Fluid Power 2011, str. 409-418, Mašinski fakultet univerziteta u Mariboru, Slovenija: 2011, UDK 621.51/54 (082), ISBN 978-961-248-290-9
8.	V. Savić, V. Karanović, M. Jocanović, D. Knežević: Pressure drop in hydraulic pipeline system - Identification of real basis for calculation of mineral hydraulic oil flow, Fluid Power 2009, str. 133-148, Mašinski fakultet univerziteta u Mariboru, Slovenija: 2009, UDK 621.51/54 (063)(082), ISBN 978-961-248-176-6
9.	V. Savić, M. Jocanović, D.Knežević, M.Kraišnik; KINEMATICS OF DISTRIBUTION OF PRESSURE WITHIN PIPELINE OF TWO'LINE SYSTEMS FOR LUBRICATION, VII TH INTERNATIONAL SYMPOSIUM INTERTRIBO 2002, str. 141 – 143, Stara Lesna, Slovak Republic (2002),
10.	V.Savić, M. Jocanović, V. Karanović: BASIC CONSTRUCTION MODEL OF THE SYSTEM FOR PROTECTION OF FRUIT TREES FROM FROST BY ICE PROTECTIVE CRUST, 14. Međunarodna naučna konferencija INDUSTRIJSKI SISTEMI - IS"08, Novi Sad: Fakultet tehničkuh nauka - Novi Sad, 2-3 Oktobar, 2008, str. 129- 134, UDK: 685.5 (082), ISBN 978-86-7892-135-3.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	2			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	2	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Jovanović M. Vukica
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	2010	Purdue University - West Lafayette	Mechatronics, Robotics and Automation and Intelligent Systems
Magister thesis	2006	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Intelligent Systems
Bachelor's thesis	2001	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H105	Fundamentals in Computer science	(H00) Mechatronics, Undergraduate Academic Studies
2.	H109	Fundamentals in Programming	(H00) Mechatronics, Undergraduate Academic Studies
3.	H1409	Intelligent Systems	(H00) Mechatronics, Undergraduate Academic Studies
4.	H1410	Programming and application of programmable logic controllers	(H00) Mechatronics, Undergraduate Academic Studies
5.	BMI110	Sensors and actuators in medicine	(BM0) Biomedical Engineering, Undergraduate Academic Studies
6.	II1009	Automatic identification systems	(I10) Industrial Engineering, Undergraduate Academic Studies
7.	II1010	Control of technical systems	(I10) Industrial Engineering, Undergraduate Academic Studies
8.	II1015	Programmable Logic Controllers (PLC)	(I10) Industrial Engineering, Undergraduate Academic Studies
9.	II1029	Computer integrated manufacturing	(I10) Industrial Engineering, Undergraduate Academic Studies
10.	II1045	Systems for measurement, surveillance and control	(I10) Industrial Engineering, Undergraduate Academic Studies
11.	II1048	Artificial intelligence in engineering	(I10) Industrial Engineering, Undergraduate Academic Studies
12.	IM1001	Fundamentals of industrial engineering	(I20) Engineering Management, Undergraduate Academic Studies
13.	IM1022	Fundamentals of technical systems control	(I20) Engineering Management, Undergraduate Academic Studies (M20) Mechanization and Construction Engineering, Undergraduate Academic Studies
14.	IM1035	Identification technologies in enterprises	(I20) Engineering Management, Undergraduate Academic Studies
15.	IM1117	Computer integrated manufacturing (CIM)	(I20) Engineering Management, Undergraduate Academic Studies
16.	IM1719	Implementation of information systems in insurance	(I20) Engineering Management, Undergraduate Academic Studies
17.	HDOK2S	Selected topics in non-industrial robotics	(I12) Industrial Engineering, Specialised Academic Studies
18.	HDOS12	Research in the area of automatic identification technology	(I12) Industrial Engineering, Specialised Academic Studies
19.	HDOS13	Motion control and application of MEMS	(I12) Industrial Engineering, Specialised Academic Studies
20.	HDOS14	Nonindustrial automation	(I12) Industrial Engineering, Specialised Academic Studies
21.	NIT08	Fundamentals of Computer Science and Informatics	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
22.	H799	Fieldbuses and protocols	(H00) Mechatronics, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
23.	I907	Automated Assembly Systems for High Accuracy	(H00) Mechatronics, Master Academic Studies (PM0) Production Engineering, Master Academic Studies
24.	IM2516	Artificial Intelligence in Engineering	(I20) Engineering Management, Master Academic Studies
25.	IM2716	Automation systems in insurance	(I20) Engineering Management, Master Academic Studies
26.	IM2721	Systems for detection, alarming and warning	(I20) Engineering Management, Master Academic Studies
27.	HDOK12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
28.	HDOK13	Motion control and the application of MEMS	(H00) Mechatronics, Doctoral Academic Studies
29.	HDOK14	Non-industrial Automation	(H00) Mechatronics, Doctoral Academic Studies
30.	HDOK-3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
31.	HDOKL3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
32.	HDOL12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
33.	HDOL13	Motion controla and application of MEMS	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
34.	HDOL14	Nonindustrial automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Ostojić G., Stankovski S., Tarjan L., Šenk I., Jovanović V.: Development and Implementation of Didactic Sets in Mechatronics and Industrial Engineering Courses, International Journal of Engeneering Education, 2010, Vol. 26, No 1, pp. 2-8, ISSN 0949-149X
2.	Jovanović V., Filipović S., Ostojić G., Stankovski S., Lazarević M.: Analysis of Possible Use of Identification Technologies in Disassembly, Facta universitatis - series: Mechanical Engineering, 2009, Vol. 7, No 1, pp. 81-82, ISSN 0354–2025, UDK: 658.515
3.	Ostojić G., Lazarević M., Jovanović V., Stankovski S., Ćosić I.: Design Process in the Assembly and Disassembly Systems Using RFID Technology , Journal for Fluid Power, Automation and Mechatronics – Ventil, 2006, Vol. 6, pp. 385-389, ISSN 1318-7279
4.	Stankovski S., Ostojić G., Jovanović V., Stevanov B.: Using RFID Technology in Collaborative Design, Facta universitatis - series: Mechanical Engineering, 2006, Vol. 4, No 1, pp. 75-82, ISSN 0354-2025, UDK: 681.518:65.011.56
5.	Ostojić G., Lazarević M., Jovanović V., Stankovski S., Ćosić I.: RFID Tehnology Use In Assembly and Disassembly Processes , Journal for Fluid Power, Automation and Mechatronics – Ventil, 2006, Vol. 6, No 12, pp. 385-389, ISSN 1318-7279, UDK: 62-82 62-85 62-31/33 681.523
6.	Jovanovic, V., DeAgostino, T.H., Thomas, M.B., Trusty II, R.T. Educating engineering students to succeed in a global workplace, 2012, ASEE Annual Conference and Exposition, Conference Proceedings
7.	Ostojić G., Jovanović V., Stankovski S., Lazarević M.: RFID Product and Part Tracking for the Preventive Maintenance, 4. ASME International Manufacturing Science and Engineering Conference (MSEC), West Lafayette: American Society of Mechanical Engineeris (ASME), 4-7 Oktobar, 2009, ISBN 978-0-7918-3859-4
8.	Jovanović V., Savić B.: Determining the Optimal Interval for the Technical Diagnostics of Bearings, 4. ASME International Manufacturing Science and Engineering Conference (MSEC), West Lafayette: American Society of Mechanical Engineers (ASME), 4-7 Oktobar, 2009, ISBN 9780791843611
9.	Jovanović V.: An Overview of Possible Integration of Green Design Principles into Mechatronic Product Development through Product Lifecycle Management, 4. ASME International Manufacturing Science and Engineering Conference (MSEC), West Lafayette: American Society of Mechanical Engineers (ASME), 4-7 Oktobar, 2009, ISBN 9780791843611
10.	Jovanović V., Ncube L.: The Curriculum as a Product: The Application of PLM to the Comprehension Collaborative Design Education Project, 7. Annual ASEE Global Colloquium in Engineering Education, Cape Town: American Society of Engineering Education (ASEE), 1 Januar, 2008

Summary data for teacher's scientific or art and professional activity:

Quotation total :	9			
Total of SCI(SSCI) list papers :	1			
Current projects :	Domestic :	1	International :	2

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Kamberović L. Bato
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.03.1979
Scientific or art field:	Quality, Effectiveness and Logistics

Academic carieer	Year	Institution	Field
Academic title election:	2007	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
PhD thesis	1996	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	1985	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1978	Faculty of Technical Sciences - Novi Sad	Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II1014	Product measurement and control techniques	(I10) Industrial Engineering, Undergraduate Academic Studies
2.	II1036	Methods and techniques of quality improvement	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	II1050	TRIBOLOGY AND LUBRICATION	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	IM1020	Quality Management System	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
5.	IM1606	Designing, Auditing and Analyses of Quality Management System	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
6.	IM1612	Methods and techniques of quality system improvements	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1613	Product measurement and control techniques	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1616	Quality planning	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1617	Quality Managament System in Service Provision	(I20) Engineering Management, Undergraduate Academic Studies
10.	IM1619	Quality and Procurement	(I20) Engineering Management, Undergraduate Academic Studies
11.	I503	Models of Excellence in Quality Management Systems	(I10) Industrial Engineering, Master Academic Studies
12.	I504	Integrated Management Systems	(I10) Industrial Engineering, Master Academic Studies
13.	IMDS95	Trends in Customer Relationship Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
14.	I309	Quality Management System	(LIM) Logistic Engineering and Management, Master Academic Studies
15.	LIM18	Life Cycle Costs and Supply	(LIM) Logistic Engineering and Management, Master Academic Studies
16.	LIM21	Total Quality Management and Logistics	(LIM) Logistic Engineering and Management, Master Academic Studies
17.	I843	Maintenance effectiveness	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
18.	I912	Process approach and quality	(I10) Industrial Engineering, Master Academic Studies
19.	IIDS12	Quality and organizational performance	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
20.	IIDS30	Trends in the environmental management systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
21.	IIDS7	Selected topics in quality engineering and logistics	(I12) Industrial Engineering, Specialised Academic Studies
22.	IM2613	Models of Excellence in Quality Management Systems	(I20) Engineering Management, Master Academic Studies
23.	IM2614	Integrated Management Systems	(I20) Engineering Management, Master Academic Studies
24.	IM2616	Product and service quality improvement - lean six sigma	(I20) Engineering Management, Master Academic Studies
25.	IM2623	Total Quality Management	(I20) Engineering Management, Master Academic Studies
26.	IMDS74	Selected Topics in Quality Management and Logistics	(I22) Engineering Management, Specialised Academic Studies
27.	IMDS76	Selected topics in industrial marketing and media engineering	(I22) Engineering Management, Specialised Academic Studies
28.	IMDR94	Trends in the environmental management systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
29.	IMDR95	Trends in Customer Relationship Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
30.	IMDR74	Selected Topics in Quality Management and Logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
31.	IMDR76	Selected topics in industrial marketing and media engineering	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
32.	IMDR79	Selected topics in quality engineering and logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
33.	IMDR83	Quality abd organisational performance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
34.	ZRD212	Integrating occupational health and safety requirements into management systems	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Delić M., Radlovački V., Kamberović B., Vulanović S., Hadžistević M., Tasić N.: ESTIMATES OF QUALITY MANAGEMENT SYSTEMS IN SERBIA , Metalurgia international, 2013, No 4, ISSN 1582-2214
2. Jovanović R., Radlovački V., Pečujlija M., Kamberović B., Delić M., Grujić J.: Assessment of blood donors' satisfaction and measures to be taken to improve quality in transfusion service establishments, Medicinski glasnik (BiH), 2012, Vol. 9, No 2, pp. 231-237
3. Radlovački V., Pečujlija M., Kamberović B., Jovanović R., Delić M., Beker I.: SATISFACTION OF HIGH SCHOOL STUDENTS WITH THE APPLICABILITY OF THEIR KNOWLEDGE, TTEM. Tehnics tehnologies education management, 2012, Vol. 7, No 2, pp. 777-785, ISSN 1840-1503
4. Radlovački V., Beker I., Majstorović V., Pečujlija M., Stanivuković D., Kamberović B.: Quality Managers' Estimates of Quality Management Principles Application in Certified Organisations in Transitional Conditions - Is Serbia Close to TQM, Strojniški vestnik - Journal of Mechanical Engineering, 2011, Vol. 57, No 11, pp. 851-861, ISSN 0039-2480
5. B. Kamebrović: MODEL INTEGRALNOG SISTEMA ZA UPRAVLjANjE KVALITETOM, Univerzitet u Novom Sadu, Institut za industrijske sisteme i IIS - Istraživački i tehnološki centar, Novi Sad, 199 strana, 1998.
6. Kamberović B., Kecojević S.: ISO 9000 I ODRŽAVANjE , Novi Sad, Fakultet tehničkih nauka - Institut za industrijske sisteme
7. Kamberović B., Radaković N.: QFD METODA , Novi Sad, Fakultet tehničkih nauka - Institut za industrijske sisteme
8. Kamberović B., Radlovački V.: SISTEM UPRAVLJANJA KVALITETOM - ZAHTEVI u knjizi: Dr Vojislav Vulanović, Dragutin Stanivuković, Bato Kamberović, R. Maksimović, Nikola Radaković, V. Radovački, M. Šilobad: SISTEM KVALITETA ISO 9001:2000, Novi Sad, Fakultet tehničkih nauka - Institut za industrijske sisteme i IIS-Istraživački i tehnološki centar, 2007, str. 39-50, ISBN 978-86-907041-3-2, UDK: 005.336.3 006.83
9. Vojislav V., Kamberović B.: KONTROLNE KARTE u knjizi: Dr Vojislav Vulanović, Dragutin Stanivuković, Bato Kamberović, R. Maksimović, Nikola Radaković, V. Radovački, M. Šilobad: METODE I TEHNIKE UNAPREĐENJA PROCESA RADA - STATISTIČKE * INŽENJERSKE * MENADŽERSKE, Novi Sad, Fakultet tehničkih nauka - Institut za industrijske sisteme i IIS-Istraživački i tehnološki centar, 2003, str. 60-120, UDK: 658.5
10. Marić B., Kamberović B., Radlovački V., Delić M., Zubanov V.: Observing the dependence between dynamic indicators of investment profitability - Relative net present value and internal rate of return, African Journal of Business Management, 2011, Vol. 5, No 26, pp. 331-337, ISSN 1993-8233

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	0	International :	0

Science, arts and professional qualifications

Name and last name:	Katalinić -. Branko
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	1983	Faculty of Mechanical Engineering and Naval Architecture - Zagreb	Mechanical Engineering
Magister thesis	1979	Faculty of Mechanical Engineering and Naval Architecture - Zagreb	Mechanical Engineering
Bachelor's thesis	1976	Faculty of Mechanical Engineering and Naval Architecture - Zagreb	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IM1213	Globalization and new business models	(I20) Engineering Management, Undergraduate Academic Studies
2.	HDOK4S	Selected chapters from automation of work processes	(I12) Industrial Engineering, Specialised Academic Studies
3.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
4.	IIDR5S	Advanced Engineering Technologies	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (M50) Energy Management, Master Academic Studies
5.	IIDS9	Effective Production and Service Systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
6.	IM2103	New technologies in engineering and management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
7.	HDOK-4	Selected Chapters in Production Process Automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
8.	HDOKL4	Selected chapters from automation of work processes	(H00) Mechatronics, Doctoral Academic Studies
9.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
10.	IMDR31	Effective Production and Service Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
11.	IMDR57	Strategic Planning and Designing Procedures and Systems at the End of Product Lifecycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	B. Katalinic, J. Balic, I. Pahole: “Scheduling of Complex Flexible Manufacturing Systems-Methodology Design”; STROJNISKI VESTNIK-JOURNAL OF MECHANICAL ENGINEERING Volume: 44 Issue: 5-6 Pages: 168-174, Published: MAY-JUN 1998
2.	B. Katalinic: "Bionic Assembly Systems: Selforganizing Complex Flexible Assembly System"; Acta Mechanica Slovaca, Vol. 6 (2002), No. 2/2002; pp. 15 - 20.
3.	B. Ljoljic, B. Katalinic, K. Stuja, V. Kordic: "Simulation of Complex Flexible Assembly System"; Acta Mechanica Slovaca, Vol.6 (2002), 2/2002; pp. 117 - 122
4.	B. Ljoljic, B. Katalinic, K. Stuja: "Optimisation of Flexible Assembly System Using Simulation"; International Journal of Simulation Modelling, Vol. 1 (2002), No 1/2002; pp. 16 - 22.
5.	A. Lazinica, B. Katalinic: “Bionic assembly system: new concept of self-organising multirobot system”; International Journal of Automation and Control, Volume 1, Number 1 / 2007; Pages: 16 – 27.
6.	B. Katalinic, V. Kordic: "Integration of Subordination and Self Organisation in Working Scenarios of Bionic Assembly System"; in: "DAAAM International Scientific Book 2003", B. Katalinic (Hrg.); herausgegeben von: DAAAM International Vienna; DAAAM International Vienna, Wien, 2003, (eingeladen), ISBN: 3-901-509-36-4, pp. 319 - 330.

Representative refferences (minimum 5, not more than 10)

7.	B. Katalinic, A. Lazinica: "Autonomous mobile robots in assembly applications"; in: "DAAAM International Scientific Book 2005", DAAAM Internaitonal Vienna; DAAAM International Vienna, Vienna, 2005, (eingeladen), ISBN: 3-901509-43-7, pp. 323 - 332.
8.	V. Malisa, B. Katalinic: "Next Generation of Production Systems: Original Concept of Selforganizing Production Systems"; Vortrag: Eight International Conference on Manufacturing & Management, Gold Coast, Queensland, Australia (eingeladen); 08.12.2004 - 10.12.2004; in: "Eight International Conference on Manufacturing Management Proceedings", (2004), ISBN: 0-9578296-1-2; pp. 1 - 14.
9.	A. Lazinica, B. Katalinic: "Design of Transport Mobile Robot Behavior in Self-Organising Assembly System"; IEEE/ASME International Conference on Advanced Intelligent Mechatronics - AIM 2005, Monterey, California, USA (eingeladen); 24.07.2005 - 28.07.2005; in: "Proceedings of 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics - AIM 2005", (2005), ISBN: 0-7803-9046-6; S. 100 - 105.
10.	B. Katalinic, V. Kordic: "Bionic Assembly System: Concept, Structure and Function"; 5th International Conference on Integrated Design and Manufacturing in Mechanical Engineering, Bath, United Kingdom (eingeladen); 05.04.2004 - 07.04.2004; in: "Proceedings of 5th International Conference on Integrated Design and Manufacturing in Mechanical Engineering", (2004).

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Katić A. Vladimir
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.1978
Scientific or art field:	Power Electronics, Machines and Facilities

Academic carieer	Year	Institution	Field
Academic title election:	2002	Faculty of Technical Sciences - Novi Sad	Power Electronics, Machines and Facilities
PhD thesis	1991	School of Electrical Engineering - Beograd	Electrical and Computer Engineering
Magister thesis	1981	School of Electrical Engineering - Beograd	Electrical and Computer Engineering
Bachelor's thesis	1978	Faculty of Technical Sciences - Novi Sad	Electrical and Computer Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EE305	Power Electronics 1	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
2.	EE308	Power Electronics 2	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
3.	Z107	Electrical Engineering, Environment and Protection	(Z01) Safety at Work, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
4.	EE0406	Electric Power Quality	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
5.	EE431	Renewable Sources and Small Power Plants	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
6.	EZ300	Clean Electrical Energy Sources	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
7.	EZ400	Clean Energy Sources Design	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
8.	DE209S	Energy Converters in Renewable Energy Sources	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
9.	DE413S	Integration of Distributed Energy Resources	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
10.	DE505S	Power Quality in Distribution Networks	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
11.	DE506S	Renewable Electrical Energy Sources	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
12.	DE509S	Effects of Power Converters on Network and Environment	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
13.	EE406	Electric Power Quality	(E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
14.	EE509	Market and Deregulation in Electric Power Industry	(E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
15.	S0I51Ž	Electrical Substation and Electric Traction	(S00) Traffic and Transport Engineering, Master Academic Studies
16.	EE544	Renewable energy sources	(E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
17.	EE564	Distributed Energy Resources	(E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
18.	ZCM02	Clean technologies for electrical vehicles	(ZC0) Clean Energy Technologies, Master Academic Studies
19.	ZCM08	Renewable and Distributed Electrical Energy Sources	(ZC0) Clean Energy Technologies, Master Academic Studies
20.	DE108	FACTS Devices and Electric Power Quality	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
21.	DE113	Application of Power Electronics in Power Systems	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
22.	DE209	Energy Converters in Renewable Power Sources	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
23.	DE413	Integration of Distributed Energy Resources	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
24.	DE505	Power Quality in Distribution Networks	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
25.	DE506	Renewable Electrical Energy Sources	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
26.	DE509	Effects of Power Converters on Network and Environment	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
27.	SID04	Current State in the Field	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies
28.	MSID04	Present State in the Field	(M40) Technical Mechanics, Doctoral Academic Studies
29.	SID04	Present State in the Field	(A00) Architecture, Doctoral Academic Studies (AS0) Scenic Design, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Vladimir Katić: "Kvalitet električne energije – viši harmonici", Univerzitet u Novom Sadu - Fakultet tehničkih nauka, Edicija Tehničke nauke - Monografije, Br. 6, Novi Sad, 2002., ISBN 86-80249-57-2.
2. Vladimir Katić: "Energetska elektronika - Zbirka rešenih zadataka", Univerzitet u Novom Sadu-Fakultet tehničkih nauka, Edicija Univerzitetski udžbenik, Broj 66, Novi Sad, 1998, tiraž 500 primeraka, strana 430, Pomoćni udžbenik, ISBN 86-499-0017-8.
3. Vladimir Katić, Darko Marčetić, Dušan Graovac: "Energetska elektronika – Praktikum laboratorijskih vežbi", Univerzitet u Novom Sadu-Fakultet tehničkih nauka, Edicija Univerzitetski udžbenik, Broj 124, Novi Sad, 2000, tiraž 300 primeraka, strana 85, Pomoćni udžbenik, ISBN 86-499-0081-X.
4. Vladimir Katić, Vlado Porobić, Darko Marčetić: “Primena mikroprocesora u energetici – Praktikum laboratorijskih vežbi“, Univerzitet u Novom Sadu-Fakultet tehničkih nauka, Edicija: Tehničke nauke - Udžbenici, Broj 149, Novi Sad, Dec. 2006, tiraž 300 primeraka, strana 122, Pomoćni udžbenik, ISBN 86-7892-013-0.
5. Vladimir Katić: „Upravljanje energetskim pretvaračima“, Fakultet tehničkih nauka – WUS, Novi Sad, 2006, tiraž 20 primeraka, str.175, Skripta.
6. Dušan Graovac, Vladimir Katić, Alfred Rufer: "Power Quality Problems Compensation with Universal Power Quality Conditioning System", IEEE Transaction on Power Delivery, USA, ISSN 0885-8977, Vol.22, No.2, April 2007, pp.968-976.
7. Vladimir Katić, Jovan Knežević, Dušan Graovac: "Application-Oriented Comparison of the Methods for AC/DC Converter Harmonics Analysis", IEEE Transaction on Industrial Electronics, USA, ISSN 0278-0046, Vol.50, No.6, December 2003, pp.1100-1108.
8. Vladimir Katić, Dušan Graovac: “A Method for PWM Rectifier Line Side Filter Optimization in Transient and Steady States”, IEEE Transaction on Power Electronics, USA, ISSN 0885-8993, Vol.17, No.3, May 2002, pp.342-352.
9. Dušan Graovac, Vladimir Katić: “On-Line Control Of Current Source Type Active Rectifier Using Transfer Function Approach”, IEEE Transaction on Industrial Electronics, USA, ISSN 0278-0046, Vol.48, No.3, June 2001, pp.526-535.
10. Vladimir Katić: ”Modern Power Electronics Technologies for Wind Power Plants”, Invited Paper, Electronics/Elektronika, Banja Luka (BIH-R.Srpska), Vol.10, No.2, Dec.2006, YU ISSN 1450-5843, pp.3-9.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	122
Total of SCI(SSCI) list papers :	19

Current projects :	Domestic :	5	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Katić R. Ivana
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	31.10.2007
Scientific or art field:	Engineering Management - Human Resource Management

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Engineering Management - Human Resource Management
PhD thesis	2012	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2008	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2007	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	2004	Faculty of Philosophy - Novi Sad	Psychological Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II205	Menadžment ljudskih resursa	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
2.	II934	Psichology of Work	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
3.	IM1914	Career Management	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1916	Industrial psychology	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1921	Managerial competence	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1923	Interpersonal intelligence in business	(I20) Engineering Management, Undergraduate Academic Studies
7.	S0I322	Human Resources Management	(S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
8.	HR005	PR Plan Development and Application	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
9.	I076/S	Leadership and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
10.	IMDS98	Modern concepts, methods and tools of human resource management	(I22) Engineering Management, Specialised Academic Studies
11.	MBA308	Business communication	(IB0) Engineering Management - MBA, Specialised Professional Studies
12.	MBA513	leadership development and teamworking	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
13.	MBA515	decision macing and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
14.	MBA522	Lobbying, presentation and negotiation skills	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
15.	MBA605	Online Public Relations	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
16.	IM2916	Professional portfolio managers	(I20) Engineering Management, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
17.	IM2921	Talent Management	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
18.	IMDS77	Selected Chapters from Human Resource Management	(I22) Engineering Management, Specialised Academic Studies
19.	IMDR98	Modern concepts, methods and tools of human resource management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
20.	IMDR77	Selected Chapters from Human Resource Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Katić (Drezgić), I.: Preduzetna inteligencija i menadžment projekata, magistarska teza, Fakultet tehničkih nauka, Univerzitet u Novom Sadu, 2007.
2.	Katić (Drezgić) I., Borocki J., Zekić S., Penezić N.: Entrepreneurship significance in restructuring process, TTEM. Tehnics tehnologies education management, 2011, Vol. 6, No 4, pp. 902-907, ISSN 1840-1503
3.	Katić (Drezgić),I. Significance of psychological factors in mass customization and personalization process, 5th International Conference on Mass Customization and Personalization in Central Europe (MCP-CE 2012), September 19-21, 2012, Novi Sad, Serbia, Proceedings, University of Novi Sad, Faculty of Technical Sciences
4.	Katić (Drezgić),I.,Pavlović,J., Lalić, D., The role of Human resources in organisational change, XIV International Scientific Conference on Industrial Systems , October 2-3, 2008, Novi Sad, Serbia, Proceedings, University of Novi Sad, Faculty of Technical Sciences, ISBN 978-86-7892-135-3, pp. 543-546.
5.	Pavlović,J., Katić(Drezgić),I., The HR Scorecard, XIV International Scientific Conference on Industrial Systems, October 2-3, 2008, Novi Sad, Serbia, Proceedings, University of Novi Sad, Faculty of Technical Sciences, ISBN 978-86-7892-135-3, pp. 571-574.
6.	Lalić, D., Katić (Drezgić), I., Vujanac.,J. The influence of the information communicational technologies on the relationships among the employees and on their success in job, XIV International Scientific Conference on Industrial Systems , October 2-3, 2008, Novi Sad, Serbia, Proceedings, University of Novi Sad, Faculty of Technical Sciences, ISBN 978-86-7892-135-3, pp. 537-542.
7.	Katić (Drezgić), I., Pavlović,J., Lalić,D., Distribucija kao faza logističkog toka sa aspekta marketing miksa, XIII Internacionalni naučni skup, Strategijski menadžment i sistemi podrške odlučivanju u strategijskom menadžmentu, Subotica, 2008, CD ROM, ISBN 86-7233-193-1,pp.124-129.
8.	Penezić, N., Katić (Drezgić), I., Lalić, B. Sindrom izgaranja kod MBA studenata, XIV Skup Trendovi razvoja: Efikasnost i kvalitet bolonjskih studija, Trend, Kopaonik, 2008, CD ROM, ISBN 978-86-7892-096-7, pp.178-181.
9.	Katić (Drezgić), I., Došen, L.,Jovanović-Boka,D. Da li je moguća psihoterapija odnosa u organizaciji?,Drugi Kongres psihoterapeuta Srbije: Odnosi u psihoterapiji, Beograd, 2012.
10.	Katić (Drezgić), I.,Došen, L.,Jovanović-Boka,D. Napredovanje u karijeri-pretnja ili izazov, Prvi Kongres psihoterapeuta Srbije: Mentalitet i psihoterapija, Beograd, 2011.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	1			
Current projects :	Domestic :	1	International :	0

Science, arts and professional qualifications

Name and last name:	Kostić Z. Marko
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.10.1999
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	2004	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	2001	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1999	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E121	Mathematical Analysis 2	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
2.	E135B	Mathematical Analysis 2	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
3.	E212	Mathematical Analysis 1	(E20) Computing and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
4.	EOS07	Mathematics 2	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
5.	F101	Mathematics	(F00) Graphic Engineering and Design, Undergraduate Academic Studies
6.	GI107	Mathematical Analysis 1	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
7.	M106	Mathematics 2	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
8.	M4202	Applied Mathematical Analysis	(M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
9.	ISIT06	Matematika 2	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
10.	0M501	Functional Analysis	(OM1) Mathematics in Engineering, Master Academic Studies
11.	0ML501	Functional Analysis	(OM1) Mathematics in Engineering, Master Academic Studies
12.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
13.	Z506	20BAdvanced Course in Mathematics 1	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies (Z20) Environmental Engineering, Master Academic Studies
14.	Z506	Viši kurs matematike 1(uneti naziv na engleskom)	(Z20) Environmental Engineering, Master Academic Studies
15.	D0M01	Functional Analysis 1	(OM1) Mathematics in Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
16.	D0M19	Functional Analysis 2	(OM1) Mathematics in Engineering, Doctoral Academic Studies
17.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Kostić, Marko, Distribution cosine functions. Taiwanese J. Math. 10 (2006), no. 3, 739--775.
2.	Kostić Marko,On analytic integrated semigroups. Novi Sad J. Math. 35 (2005), no. 1, 127--135.
3.	Kostić Marko,Convoluted C-cosine functions and convoluted C-semigroups. Bull. Cl. Sci. Math. Nat. Sci. Math. No. 28 (2003), 75--92.
4.	Kostić Marko, On a class of quasi-distribution semigroups, Novi Sad J. Math 36 (2), 137-152
5.	M. Kostić, P. J. Miana, Relations between distribution cosine functions and almost-distribution cosine functions, Taiwanese Journal of Mathematics 11 (2007), 531--543.
6.	M. Kostić, S. Pilipović, Global convoluted semigroups, accepted in Math. Nachr.
7.	M. Kostić, S. Pilipović: Convoluted C-cosine functions and semigroups. Relations with ultradistribution and hyperfunction sines, accepted in J. Math. Anal. Appl.
8.	M. Kostić: Complex powers of operators, accepted in Publications De"l Institute Mathematique
9.	M. Kostić: C-Distribution semigroups, Studia Math. 185 (2008), 201--217.
10.	M. Kostić: Convoluted operator families and abstract Cauchy problems, accepted in Kragujevac Journal of Mathematics

Summary data for teacher's scientific or art and professional activity:

Quotation total :	32			
Total of SCI(SSCI) list papers :	15			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Kovačević M. Ilija
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.09.1972
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	1990	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	1979	Faculty of Mathematics - Beograd	Mathematical Sciences
Magister thesis	1975	Faculty of Mathematics - Beograd	Mathematical Sciences
Bachelor's thesis	1971	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E212	Mathematical Analysis 1	(E20) Computing and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
2.	EE204	Selected Chapters in Mathematics	(MR0) Measurement and Control Engineering, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
3.	E102	Mathematical Analysis 1	(ES0) Power Software Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
4.	E102A	Mathematical Analysis 1	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
5.	IM1423	Financial Mathematics	(I20) Engineering Management, Undergraduate Academic Studies
6.	0M501	Functional Analysis	(OM1) Mathematics in Engineering, Master Academic Studies
7.	0ML501	Functional Analysis	(OM1) Mathematics in Engineering, Master Academic Studies
8.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
9.	I004/S	Statistical Quantitative Methods	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
10.	GS012	Selected Chapters in Mathematics	(G10) Energy Efficiency in Buildings, Specialised Academic Studies
11.	MPK001	Statistical and Numerical Methods	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies
12.	SDOM30	Probability, Statistics and Theory of Engineering Experiment	(Z00) Environmental Engineering, Specialised Academic Studies
13.	D0M01	Functional Analysis 1	(OM1) Mathematics in Engineering, Doctoral Academic Studies
14.	D0M19	Functional Analysis 2	(OM1) Mathematics in Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
15.	DOM30	Probability, Statistics and Theory of Engineering Experiment	(M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
16.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	I.Kovačević, Some properties of Mn subsets and almost closed mappings, Indian J.pure appl. Math., 27(9), 1996., 875-881.
2.	I.Kovačević, On almost closed mapping, paracompactness and partial equivalence relatuions, Indian Journal of Pure and Applied mathematics,25(9), 1994., 949-954.
3.	I.Kovačević, On alfa-Hausdorff subsets, almost closed mappings and almost upper semicontinuous decomposition, Indian Jurnal of Pure and Applied mathematics 20 (4) 1989., 334-340.
4.	Kiurski J., Oros I., Ralević N., Kovačević I., Adamović (Majkić) S., Krstić J., Čomić L.: Cluster and principal component analysis in the assessment of fountain solution quality, Carpathian Journal of Earth and Environmental Sciences, 2013, Vol. 8, No 1, pp. 19-23, ISSN 1842-4090
5.	N. Adžić, I. Kovačević, V. Marić, V. Ungar, Matematička analiza 2, FTN (Edicija tehničke nauke-udžbenici), Novi Sad, 1996., 1-299.
6.	I. Kovačević, N. Ralević, Funkcionalna analiza,FTN (Edicija tehničke nauke-udžbenici), Novi Sad, (Ponovljeno i dopunjeno izdanje)2004., 1-203.
7.	I. Kovačević, N. Ralević, B.Carić,V.Marić,M.Novković,S.Medić,Matematička analiza 1- uvodni pojmovi i granični procesi ,(Ponovljeno i dopunjeno izdanje), FTN (Edicija tehničke nauke-udžbenici) Novi Sad, 2012,1-155.
8.	I.Kovačević, V.Marić, M.Novković, B.Carić, N.Ralević,S.Medić, Matematička analiza 1 - diferencijalni i integralni račun, obične diferencijalne jednačine (Ponovljeno i dopunjeno izdanje),FTN (Edicija tehničke nauke-udžbenici), Novi Sad,2012., 1-280.
9.	I. Kovačević, Algebra, Naučna knjiga, Beograd, 1990., 1-116.
10.	M.Novković,B.Carić,I.Kovačević, Zbirka rešenih zadataka iz verovatnoće i statistike, FTN (Edicija tehničke nauke-udžbenici), Novi Sad, (Ponovljeno i dopunjeno izdanje) 2012., 1-169.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	28			
Total of SCI(SSCI) list papers :	7			
Current projects :	Domestic :	3	International :	2

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Kozak V. Dražen
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	2012		Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	2001	Faculty of Mechanical Engineering and Naval Architecture - Zagreb	Mechanical Engineering
Magister thesis	1995	Faculty of Mechanical Engineering and Naval Architecture - Zagreb	Mechanical Engineering
Bachelor's thesis	1991	Mechanical Engineering Faculty - Slavonski Brod - Slavonski Brod	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H102	Fundamentals in Product Development	(H00) Mechatronics, Undergraduate Academic Studies
2.	H105	Fundamentals in Computer science	(H00) Mechatronics, Undergraduate Academic Studies
3.	H109	Fundamentals in Programming	(H00) Mechatronics, Undergraduate Academic Studies
4.	H1410	Programming and application of programmable logic controllers	(H00) Mechatronics, Undergraduate Academic Studies
5.	H1501A	Systems for Survailance and Visualisation of Process	(H00) Mechatronics, Undergraduate Academic Studies
6.	H308	Industrial Robotics	(H00) Mechatronics, Undergraduate Academic Studies
7.	BMI106	Rehabilitation devices and systems	(BM0) Biomedical Engineering, Undergraduate Academic Studies
8.	H301	System Modeling and Symulation	(H00) Mechatronics, Master Academic Studies
9.	HDOS12	Research in the area of automatic identification technology	(I12) Industrial Engineering, Specialised Academic Studies
10.	HDOS13	Motion control and application of MEMS	(I12) Industrial Engineering, Specialised Academic Studies
11.	HDOS14	Nonindustrial automation	(I12) Industrial Engineering, Specialised Academic Studies
12.	NIT06	Advanced Technologies for Manufacturing Support	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
13.	NIT08	Fundamentals of Computer Science and Informatics	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
14.	H828	Advanced robotics	(H00) Mechatronics, Master Academic Studies
15.	IIDS6	Selected chapters in automation	(I12) Industrial Engineering, Specialised Academic Studies
16.	IM2516	Artificial Intelligence in Engineering	(I20) Engineering Management, Master Academic Studies
17.	IM2721	Systems for detection, alarming and warning	(I20) Engineering Management, Master Academic Studies
18.	HDOK12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
19.	HDOK13	Motion control and the application of MEMS	(H00) Mechatronics, Doctoral Academic Studies
20.	HDOK14	Non-industrial Automation	(H00) Mechatronics, Doctoral Academic Studies
21.	HDOK-3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
22.	HDOKL3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
23.	HDOL12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
24.	HDOL13	Motion controla and application of MEMS	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	HDOL14	Nonindustrial automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Kozak, D., Gubeljak, N., Konjatić, P., Sertić, J. Yield load solutions of heterogeneous welded joints (2009) International Journal of Pressure Vessels and Piping, 86 (12), pp. 807-812.

Representative refferences (minimum 5, not more than 10)

2.	Hloch, S., Valíček, J., Kozak, D., Tozan, H., Chattopadhyaya, S., Adamčík, P. Analysis of acoustic emission emerging during hydroabrasive cutting and options for indirect quality control (2012) International Journal of Advanced Manufacturing Technology, pp. 1-14.
3.	Valíček, J., Hloch, S., Kozak, D. Surface geometric parameters proposal for the advanced control of abrasive waterjet technology (2009) International Journal of Advanced Manufacturing Technology, 41 (3-4), pp. 323-328.
4.	Kladaric, I., Kozak, D., Krumes, D. The effect of aging parameters on properties of maraging steel (2009) Materials and Manufacturing Processes, 24 (7-8), pp. 747-749.
5.	Valíček, J., Čep, R., Rokosz, K., Łukianowicz, C., Kozak, D., Zeleåák, M., Koštial, P., Hloch, S., Harničárová, M., Hlaváček, P., Haluzíková, B. New way to take control of a structural grain size in the formation of nanomaterials by extrusion (2012) Materialwissenschaft und Werkstofftechnik, 43 (5), pp. 405-411.
6.	Brillová, K., Ohlídal, M., Valíček, J., Kozak, D., Hloch, S., Zeleňák, M., Harničárová, M., Hlaváček, P. Spectral analysis of metallic surfaces topography generated by abrasive waterjet (2012) Tehnicki Vjesnik, 19 (1), pp. 1-9.
7.	Neslušan, M., Mrkvica, I., Čep, R., Kozak, D., Konderla, R. Deformations after heat treatment and their influence on cutting process (2011) Tehnicki Vjesnik, 18 (4), pp. 601-608.
8.	Younise, B., Rakin, M., Medjo, B., Gubeljak, N., Kozak, D., Sedmak, A. Numerical analysis of constraint effect on ductile tearing in strength mismatched welded CCT specimens using micromechanical approach (2011) Tehnicki Vjesnik, 18 (3), pp. 333-340.
9.	Vojvodić, D., Kozak, D., Sertić, J., Mehulić, K., Celebic, A., Komar, D. Influence of depth alignment of E-glass fiber reinforcements on dental base polymer flexural strength (2011) Materialpruefung/Materials Testing, 53 (9), pp. 528-535.
10.	Kozak, D., Ivandić, Z., Kontajić, P. Determination of the critical pressure for a hot-water pipe with a corrosion defect (2010) Materiali in Tehnologije, 44 (6), pp. 385-390.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	39			
Total of SCI(SSCI) list papers :	36			
Current projects :	Domestic :	1	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Kozmidis-Luburić F. Uranija
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.09.1975
Scientific or art field:	Physics

Academic carieer	Year	Institution	Field
Academic title election:	2000	Faculty of Technical Sciences - Novi Sad	Physics
PhD thesis	1988	Faculty of Sciences - Novi Sad	Physical Science
Magister thesis	1986	Faculty of Physics - Beograd	Physical Science
Bachelor's thesis	1974	Faculty of Sciences - Novi Sad	Physical Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E103	Physics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
2.	EOS06	Physics	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
3.	S014	Physics	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
4.	A401	Architectural Physics	(A00) Architecture, Undergraduate Academic Studies
5.	DZ01FS	Selected Chapters in Physics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
6.	DZ01F	Selected Chapters in Physics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. U.F.Kozmidis-Luburić and B.S.Tošić, "NON-LINEAR OPTICAL EFFECTS AND THE DIELECTRIC PROPERTIES OF CRYSTALS", Physica B 112, 331(1982)
2. D.Mirjanić, U.F.Kozmidis-Luburić, M.M.Marinković and B.S.Tosić, "COMBINED EFFECT OF EXCITION-EXCITION AND EXCITION-PHONON INTERACTION ON CRYSTALS DIELECTIC PROPERTIES", Can. J. Phys. 60, 1838(1982)
3. U.F. Kozmidis-Luburić and B.S. Tošić, "KINEMATICAL INTERACTION OF OPTICAL EXCITATION AND CONSEQUENCES", Physica A 153, 266(1988)

Representative refferences (minimum 5, not more than 10)

4.	LJ. Budinski-Petković and U.Kozmidis-Luburić, "J AMING CONFIGURATIONS FOR IRREVERSIBLE DEPOSITION ON A SQUARE LATTICE", Psysica A 236, 211(1997)
5.	Lj. Budinski-Petković and U. Kozmidis-Luburić, "RANDOM SEQUENTIAL ADSORPTION ON A TRIANGULAR LATTICE", Psysical Review E 56, 6904(1997)
6.	V.Sajfert,B.S.Tošić,M.Marinković and U.F.KOZMIDIS-LUBURIĆ,"SURFACE DEFORMATION IN FILMS AND EXCITON CONCETRATION", Physica A 166, 430(1990)
7.	B.S.Tošić, Lj.Mašković, U. F. KOZMIDIS-LUBURIĆ, V.Jovovic and G. Davidovic, "Transition FROM THE DEFORMED STRUCTURE TO THE STATISTICALLY EQUIVALENT IDEAL STRUCTURE AND AN ESTIMATE OF THE BASIS PHYSICAL CHARACTERISTICS OF THE DEFORMED STRUCTURE", Physica A 216, 478(1995)
8.	V.Jovović, G.Davidović, B.S.Tošić,Lj.Mašković, U.F.KOZMIDIS-LUBURIĆ and D.Ćirić,"MASS DISTRIBUTION IN HETEROGENEOUS STRUCTURES", Physica A 223,263(1996)
9.	Lj. Budinski-Petković and U. KOZMIDIS-LUBURIĆ, "IRREVERSIBLE DEPOSITION ON DISORDERED SUBSTRATES: LINE SEGMENTS ON A SQUARE LATTICE", Physica A 245,261(1997)
10.	Lj. Budinski-Petković and U. KOZMIDIS-LUBURIĆ, "IRREVERSIBLE DEPOSITION OF DIRECTED SELF-AVOIDING RANDOM WALKS ON A SQUARE LATTICE", Physica A 262,388(1999)

Summary data for teacher's scientific or art and professional activity:

Quotation total :	68			
Total of SCI(SSCI) list papers :	23			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Kozmidis-Petrović F. Ana
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.09.1975
Scientific or art field:	Physics

Academic carieer	Year	Institution	Field
Academic title election:	1997	Faculty of Technical Sciences - Novi Sad	Physics
PhD thesis	1984	Faculty of Sciences - Novi Sad	Physics
Magister thesis	1980	Faculty of Mathematics - Beograd	Physical Science
Bachelor's thesis	1972	Faculty of Sciences - Novi Sad	Physical Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E103	Physics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
2.	GG06	Civil Engineering Physics	(G00) Civil Engineering, Undergraduate Academic Studies
3.	M101	Technical Physics	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies (ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
4.	ZR440	Influence of radiation on health and occupational safety	(Z01) Safety at Work, Undergraduate Academic Studies
5.	ZC008	Technical physics	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
6.	DZ01FS	Selected Chapters in Physics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
7.	SZD017	Solid Materials in the Environment	(Z00) Environmental Engineering, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
8.	DZ01F	Selected Chapters in Physics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
9.	FDS141	Selected Chapters in Colour Management	(F00) Graphic Engineering and Design, Doctoral Academic Studies
10.	ZD017	Solid Materials in the Environment	(Z00) Environmental Engineering, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	D. M. Petrović, A. F. Petrović, V. M. Leovac, S. R. Lukić: Thermal decomposition of Cu(II) complexes with salicyladehyde S-methylthiosemicarbazone, Journal of Thermal Analysis, 42, 1165-1170, 1994.
2.	S.R. Lukić, D. M. Petrović, A. F. Petrović, F. Skuban, I.I. Turyanitsa: Tendency towards crystallization of Ge-As-Te system glasses, Journal of Materials Science Lett., 15,.
3.	A. F. Petrović, S. R. Lukić, D. M. Petrović, E. Z. Ivegeš, V. M. Leovac: Metal complex with pyrazole derived ligands. Part IV. Thermal decomposition of Cobalt(II) complexes with 3(5)-amino-4-acetyl 5(3) mathylpyrazole, Journal of Thermal Analysis, 47, 879-886,
4.	S. R. Lukić, D. M. Petrović, A. F. Petrović: Effect of copper on conductivity of amorphous AsSeyIz, Journal of Non-Crystalline Solids, 241, 74-77, 1998.
5.	S. R. Lukić, V. M. Leovac, A. F. Petrović, S. J. Skuban, V. I. Češljević, M. M.Garić: Metal Complexes with Pyrazole-derived Ligands. XIII. Synthesis and Thermal Studies of Zn(II) Complexes with 3-amino-4-acetyl-5-methylpyrazole, Synth.React.Inorg. Met.-Org.Chem.,2002
6.	S. R. Lukić, S. J. Skuban, D. M. Petrović, A. F. Petrović, M. Garić, Characteristics of complex non-crystalline chalcogenides from the Ge-As-S-Se-I system, Journal of Optoelectronics & Advanced Materials, 6(3), 755-768, 2004.
7.	A. F. Petrović, S.R. Lukić, D.D. Štrbac: Critical rate of cooling glassy melts under conditions of continuous nucleation.The application to some chalcogenide glasses, Journal of Optoelectronics & Advanced Materials, 6(4) 1167-1177, 2004.
8.	S. R. Lukić, D. M. Petrović, Ž. N. Cvejić,.A F. Petrović, F. Skuban: Thermally-induced Structural Changes in Copper-containing Chalcogenide Thin Films, Journal of Optoelectronics & Advanced Materials, 3(2), 337-340, 2001.
9.	S.R. Lukić, D.M. Petrović, G.R.Štrbac, A.F.Petrović, M Šiljegović : Effect of sulfur atom substitute with selenium on stability of glassy Ge20As14SxSe52-xI14, Journal of Physics and Chemistry of Solids 66, 1683-1686 (2005)
10.	A.F.Kozmidis-Petrovic, G.R.Strbac, D.D.Strbac, Kinetics of non-isothermal crystallization of chalcogenide, J.Non-Cyst.Solids, 2014–2019, 353(2007)2014

Summary data for teacher's scientific or art and professional activity:

Quotation total :	153			
Total of SCI(SSCI) list papers :	25			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Krsmanović B. Cvijan
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.05.1981
Scientific or art field:	Information-Communication Systems

Academic carieer	Year	Institution	Field
Academic title election:	2004	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
PhD thesis	1994	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
Magister thesis	1986	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
Bachelor's thesis	1981	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II1003	Product development and design	(I10) Industrial Engineering, Undergraduate Academic Studies
2.	II1005	Computer Aided Product Design and Analysis	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	II1018	Design of Information Systems	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	II1039	Resource planning systems in manufacturing	(I10) Industrial Engineering, Undergraduate Academic Studies
5.	II1049	Manufacturing documentation management (DMS)	(I10) Industrial Engineering, Undergraduate Academic Studies
6.	IM1029	Information and communication systems	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1048	Enterprise resource planning systems	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1513	Management of information systems development	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1521	Business document management systems	(I20) Engineering Management, Undergraduate Academic Studies
10.	ZC014	Information technologies in energetic management	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
11.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
12.	IMDS33	Structures of Modern Information and Communication Systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
13.	IMDS34	Raster and Image Processing Technologies in Engineering and Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
14.	IMDS37	CAE/CAD/CAM and CIM Concepts and Systems	(I12) Industrial Engineering, Specialised Academic Studies
15.	MUO004	Information Systems in Education	(I20) Engineering Management, Specialised Professional Studies
16.	IIDS8	Selected chapters from Information, management and communication systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies
17.	IM2507	Automation of production systems management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
18.	IM2514	Software Quality Assurance	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
19.	IM2521	Distance Learning and Remote Work	(I20) Engineering Management, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
20.	IMDS73	Selected chapters from Information management	(I22) Engineering Management, Specialised Academic Studies
21.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
22.	IMDR33	Structures of Modern Information and Communication Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
23.	IMDR34	Raster and Image Processing Technologies in Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	IMDR37	CAE/CAD/CAM and CIM Concepts and Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	IMDR73	Selected chapters from Information management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	IMDR81	Selected chapters from Information, management and communication systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Bojanič, P. O., Maneski, T., Krsmanovič, C.: Moduljnij princip proektirovanija s pomošćju EVM, štampani izvod, Referativnij žurnal vesesojuznogo komiteta po nauki i informatiki SSSR, strana 17., UDK 621.9.06.001.63.681.142, broj 12A129, Moskva, 1985.
2.	Mogin, P., Krsmanović, C., Luković, I., Brkić, M. : Basic Elements of the IIS* Approach to Information Systems and Database Design, International Journal of INDUSTRIAL SYSTEMS, Vol. 1, Institute of Industrial Systems Engineering, Novi Sad, Yugoslavia, December 1998.
3.	Bojanić, P., Krsmanović, C. : Paths and Crossroads in CAx Technologies Implementation in Engineering at the End of 2nd Millennium, International Journal of Industrial Systems Engineering, Novi Sad, Yugoslavia, October 1999., p.p. 69 - 76
4.	Radaković, N., Maksimović, R., Krsmanović, C. : Struktura radova u procesu industrijskog projektovanja - podloge za uvođenje automatizacije (CAD/CAM tehnologije), stručni časopis PROIZVODNJA, broj 12, p.p. 18 - 31, Beograd, 1980.
5.	Krsmanović, C., Radović, B., Govedarica, M., Baloš, D. : Industrial Engineering and Informatics - Does There Boundary Exist ?, III International Szmposium INTERDISCIPLINARY REGIONAL RESEARCH (Hungary, Romania, Yugoslavia), Proceedings, Novi Sad, September 24 - 25, 1998.
6.	Navalušić, S., Gatalo, R., Krsmanović, C., Hodolič, J., Milojević, Z.: Automated Product Design as a Segment of the Computer Integrated Manufacturing, III International Symposium INTERDISCIPLINARY REGIONAL RESEARCH (Hungary, Romania, Yugoslavia), Proceedings, Novi Sad, September 24 - 25, 1998.
7.	Krsmanović, C., Stefanović, D.: Startegic Planning of Data Protection and Data Access After Catastrophic Events, VI International Symposium INTERDISCIPLINARY REGIONAL RESEARCH (Hungary, Romania, Yugoslavia), Proceedings, Novi Sad, September 2002
8.	Krsmanović, C., Simić, M.: Osnove razvoja i projektovanja multifunkcijskih i inteligentnih tehničkih sistema, XII međunarodna konferencija Industrijski sistemi - IS 2002., Zbornik radova, p.p. 354 - 359, Vrnjačka Banja, Novembar 2002.
9.	Anišić, Z., Krsmanović, C.: Assembly Initiated Production as a Prerequisite for Mass Customization and Effective Manufacturing, Journal of Mechanical Engineering, Vol. 54, No. 9, Ljubljana, September 2008., pp. 607. - 618. (Thomson Scientific Journal List)
10.	Anderla, A.,Brkljac, B., Stefanovic, D., Krsmanović, C., Sladojevic, S., Culibrk, D.: 3D Reconstruction from MRI Images, Metalurgia International, Bucharest, Romania (accepted for publication in Journal number 4, April 2013) - Thomson Scientific Journal List

Summary data for teacher's scientific or art and professional activity:

Quotation total :	7			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	1	International :	2

Science, arts and professional qualifications

Name and last name:	Kulić J. Filip
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.09.1994
Scientific or art field:	Automatic Control and System Engineering

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Automatic Control and System Engineering
PhD thesis	2003	Faculty of Technical Sciences - Novi Sad	Automatic Control and System Engineering
Magister thesis	1999	Faculty of Technical Sciences - Novi Sad	Automatic Control and System Engineering
Bachelor's thesis	1994	Faculty of Technical Sciences - Novi Sad	Electroenergetics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	AU44	Control Systems Design	(E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
2.	E226	Automatic Control Systems	(E20) Computing and Control Engineering, Undergraduate Academic Studies (H00) Mechatronics, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
3.	E238A	Control Systems Technology	(BM0) Biomedical Engineering, Undergraduate Academic Studies (E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
4.	EEI302	Systems of Automatic Control in Power Engineering	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
5.	H1405	Optimization Methods	(H00) Mechatronics, Undergraduate Academic Studies
6.	H302	Control Systems 2	(H00) Mechatronics, Undergraduate Academic Studies
7.	M325	Automatic Control Systems	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies
8.	BMI125	Biological Control Systems	(BM0) Biomedical Engineering, Undergraduate Academic Studies
9.	E2315	Electrical Machines in Automatic Control Systems	(E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
10.	EMSAU 1	Automatic Control Systems in Electronics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
11.	SEAU01	Nonlinear programming and evolutionary computations	(SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies
12.	SEAU03	Real-time control algorithms	(SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies
13.	DE410S	Selected Topics in the Field of Automatic Control	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
14.	E2515	Intelligent Control Systems	(E20) Computing and Control Engineering, Master Academic Studies (MR0) Measurement and Control Engineering, Master Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
15.	M2550	Automatic Control Systems in Motor Vehicles	(M22) Mechanization and Construction Engineering, Master Academic Studies
16.	E2532	Automatic Control Systems Project Management	(E20) Computing and Control Engineering, Master Academic Studies
17.	SEAM01	Intelligent Control Systems	(SE0) Software Engineering and Information Technologies, Master Academic Studies
18.	DAU007	Selected Topics in Artificial Intelligence in Control and Signal Processing	(E20) Computing and Control Engineering, Doctoral Academic Studies
19.	DE410	Selected Topics in the Field of Automatic Control	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
20.	SID04	Current State in the Field	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies
21.	DAU017	Selected Topics from Totally Integrated Automatic Control Systems	(E20) Computing and Control Engineering, Doctoral Academic Studies
22.	SID04	Present State in the Field	(A00) Architecture, Doctoral Academic Studies (AS0) Scenic Design, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Dragan Kukolj, Vesna Bengin, Filip Kulić: Osnovi klasične teorije automatskog upravljanja kroz rešene probleme, Sombor, Somel, 1995. 241str., UDK: 681.5(075.8),
2.	Dragan Kukolj, Filip Kulić: Projektovanje sistema automatskog upravljanja u prostoru stanja, Novi Sad, Fakulet tehničkih nauka, 1995. 232str., UDK: 681.5(075.8),
3.	D.Kukolj, F.Kulić, E.Levi: Design Of The Speed Controller For Sensorless Electric Drives Based On AI Techniques: A Comparative Study, Artificial Intelligence in Engineering, 2000, Vol. 14, str. 165- 174
4.	D.Kukolj, S.Kuzmanović, E.Levi, F.Kulić: Design of Near Optimal, Wide Range Fuzzy Logic Controller, Fuzzy Sets and Systems, 2001, Vol. 120, No. 1, str. 17- 34
5.	D.Kukolj, F.Kulić, D.Popović, Z.Gorečan: Determining Topological Changes and Critical Load Levels of a Power System by Means of Artificial Neural Network, Electric Machines and Power Systems, 1997, Vol. 25, No. 8, str. 917- 926, ISSN 0731-356x.
6.	D.Kukolj, D.Popović, F.Kulić, Z.Gorečan: Fast Dynamic Stability Analysis of a Power System Using Artificial Neural Networks, European Transactions on Electrical Power (ETEP), 1998, Vol. 8, No. 3, str. 207- 212, ISSN 1430-144X.
7.	D.Popović, D.Kukolj, F.Kulić: Monitoring and Assessment of Voltage Stability Margins Using Artificial Neural Networks with a Reduced Input Set, IEE Proc. -Gener. Transm. Distrib, 1998, Vol. 145, No. 4, str. 355- 362, ISSN 1350-2360.

Representative refferences (minimum 5, not more than 10)

8.	Matić Dragan, Kulić Filip, Pineda-Sanchez Manuel, Kamenko Ilija: “Support vector machine classifier for diagnosis in electrical machines: Application to broken bar”, Expert Systems With Applications, vol.39 br.10, str. 8681-8689, 2012.
9.	Čongradac Velimir, Kulić Filip: “Recognition of the importance of using artificial neural networks and genetic algorithms to optimize chiller operation”, Energy and Buildings, vol. 47, str. 651-658; April 2012.
10.	Ilić Slobodan; Vukmirović Srđan; Erdeljan Aleksandar; Kulić Filip: “Hybrid Artificial Neural Network System for Short-Term Load Forecasting, Thermal Science, vol.16, br. , str. S215-S224, 2012

Summary data for teacher's scientific or art and professional activity:

Quotation total :	32			
Total of SCI(SSCI) list papers :	12			
Current projects :	Domestic :	2	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Lalić P. Bojan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	17.06.2002
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2011		Production Systems, Organization and Management
PhD thesis	2011	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2004	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	2001	Faculty of Technical Sciences - Novi Sad	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EOS39	Projektni menadžment	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
2.	II1017	Production System Design	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	II1019	Project Management	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	IM1019	Commercial Processes	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1026	E-Business	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1027	Production systems	(I20) Engineering Management, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
7.	IM1046	Structural and Development Projects	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1104	Strategic Management	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1106	Business Process Simulation	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
10.	IM1319	Platforms and systems for knowledge transfer	(I20) Engineering Management, Undergraduate Academic Studies
11.	IM2123	Operations management	(M50) Energy Management, Master Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
12.	IS001	Effective management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
13.	MBA304	Business Strategies	(IB0) Engineering Management - MBA, Specialised Professional Studies
14.	MBA413	Knowledge Systems and Project Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
15.	MBA601	Applied use of IT and Internet in business	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
16.	PLM05	Management of PLM Projects	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
17.	SZP003	Selected Chapters in Applied Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
18.	RPR005	Project Cycle Management	(RPR) Regional Development Planning and Management, Master Academic Studies
19.	IM2101	Intelligent Enterprising and Effective Management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
20.	IM2123	Operations management	(M50) Energy Management, Master Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
21.	IM2124	Production and Service Systems	(H00) Mechatronics, Master Academic Studies (M50) Energy Management, Master Academic Studies
22.	IM2307	Strategic Project Management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies (Z20) Environmental Engineering, Master Academic Studies
23.	IM2314	Program and Portfolio management	(I20) Engineering Management, Master Academic Studies
24.	IM2316	Theory of Constraints	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
25.	IM2319	Project evaluation	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
26.	IM2922	eHRM	(I20) Engineering Management, Master Academic Studies
27.	IMDS71	Selected topics of project management	(I22) Engineering Management, Specialised Academic Studies
28.	S1I594	E-Business	(S01) Postal Traffic and Telecommunications, Master Academic Studies
29.	UP002	Applied Project Cycle Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
30.	IMDR71	Selected topics of project management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
31.	ZRD27A	Operations management in the security and occupational safety	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Lalić, B., Ćosić I., Anišić, Z.: SIMULATION BASED DESIGN AND RECONFIGURATION OF PRODUCTION SYSTEMS, International journal of Simulation Modelling, IJSIMM, issn 1726-4529, Volume 4, Number 4, pp. 173-183, Vienna, Austria, December 2005.
2.	R. Maksimovic, B.Lalić; Flexibility and Complexity of Effective Enterprises, Strojniski Vesnik, 2008.
3.	Lalić B., Marjanović U.: Organizational Readiness/Preparedness. In: M.M. Cruz-Cunha and J. Varajao, ed. E-business issues, challenges and opportunities for SMEs: driving competitiveness., New York, Business Science Reference (IGI Global), 2011, str. 101-116, ISBN 978-1-61692-880-3
4.	Simeunović N., Ćosić I., Radaković N., Lalić B.: The General Work Procedure Model for the Service Product, Beč, DAAAM International Scientific Book, 2009, str. 281-288, ISBN 987-3-901509-71-1, UDK: ISSN 1726-9687
5.	Lalić B., Palčič I.: Analytical Hierarchy Process as a Tool for Selecting and Evaluating Projects, International journal of Simulation Modelling-IJSIMM, 2009, Vol. 8, No 1, pp. 16-26, ISSN 1726-4529
6.	Lalić B., Ćosić I., Anišić Z.: SIMULATION BASED DESIGN AND RECONFIGURATION OF PRODUCTION SYSTEMS , International journal of Simulation Modelling-IJSIMM, 2005, Vol. 4, No 4, pp. 173-183, ISSN 1726-4529
7.	Jovanovic M., Moreno Perez J., Lalić B., Todorovic V., Jovanović M.: Use of cost analysis, estimation and risk management in making project management decisions in construction, Projektna mreza Slovenije - Project Management Review, 2010, Vol. 8, No 3, pp. 4-9, ISSN 1580-0229
8.	Lalić D., Marjanović U., Lalić B.: The influence of social networks on communication satisfaction within the organizations. In: M.M. Cruz-Cunha, P. Goncalves, N. Lopes, E.M. Miranda and G.D. Putnik, ed. Handbook of Research on Business Social Networking: Organizational, Managerial, and Technological Dimensions., New York, Business Science Reference (IGI Global), 2011, str. 545-566, ISBN 978-1-61350-168-9
9.	Lalić B., Ćosić I., Poli M.: Project Strategy Matching Project Structure to Project Type to Achieve Better Success, International Journal of Industrial Engineering and Management - IJIEM, 2010, Vol. 1, No 1, pp. 29-40, ISSN 2217-2661

Representative refferences (minimum 5, not more than 10)	
10.	Poli M., Mithiborwala H., Maksimović R., Lalić B.: PROJECT STRATEGY: SELECTING THE BEST PROJECT STRUCTURE, 9. PICMET Conference, Portland: Portland International Center for Management of Engineering and Technology, 2-6 Avgust, 2009, pp. 1276-1281, ISBN 978-1-890843-20/5

Summary data for teacher's scientific or art and professional activity:				
Quotation total :	4			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	2	International :	2

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Lalić S. Danijela
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	30.06.2004
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2010	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2007	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	2004	Faculty of Technical Sciences - Novi Sad	Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EOS39	Projektni menadžment	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
2.	II202	Marketing	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
3.	II205	Menadžment ljudskih resursa	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
4.	IM1019	Commercial Processes	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1023	Business Communication	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1817	Public Relations	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1919	Employee Relations	(I20) Engineering Management, Undergraduate Academic Studies
8.	S0I322	Human Resources Management	(S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
9.	HR005	PR Plan Development and Application	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
10.	HR017	Corporate Communication Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	I076/S	Leadership and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
12.	IMDS68	Business communication in efective sistems	(I22) Engineering Management, Specialised Academic Studies
13.	MBA304	Business Strategies	(IB0) Engineering Management - MBA, Specialised Professional Studies
14.	MBA308	Business communication	(IB0) Engineering Management - MBA, Specialised Professional Studies
15.	MBA513	leadership development and teamworking	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
16.	MBA515	decision macing and change	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
17.	MBA522	Lobbying, presentation and negotiation skills	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
18.	MBA524	interculture business communications	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
19.	MBA605	Online Public Relations	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
20.	PLM01	PLM Platform	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
21.	NIT04	Communication Skills	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
22.	RPR005	Project Cycle Management	(RPR) Regional Development Planning and Management, Master Academic Studies
23.	RPR013	Management of Human Resources	(RPR) Regional Development Planning and Management, Master Academic Studies
24.	IM2817	Internet and Social Media Communication	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
25.	IM2820	Event Marketing	(I20) Engineering Management, Master Academic Studies
26.	IM2907	Leadership	(I20) Engineering Management, Master Academic Studies
27.	IM2914	Corporate Communications Management	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
28.	IMDS76	Selected topics in industrial marketing and media engineering	(I22) Engineering Management, Specialised Academic Studies
29.	IMDS77	Selected Chapters from Human Resource Management	(I22) Engineering Management, Specialised Academic Studies
30.	IMDR68	Business Communication in Effective Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
31.	IMDR76	Selected topics in industrial marketing and media engineering	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
32.	IMDR77	Selected Chapters from Human Resource Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
33.	ZRD27A	Operations management in the security and occupational safety	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Vlastelica Bakić, T., Lalić, D., Verčić, D. “Employee Engagement: The case of Coca-Cola Hellenic Serbia”, BledCom 2011, 18th International Public Relations Research Symposium BledCom, 1-2. jul 2011, Bled, Slovenija, ISBN 978-961-90484-8-1, str. 32-41.
2. Lalić D., Popovski K., Gecevska V., Popovska Vasilevska S., Tešić Z.: Analysis of the opportunities and challenges for renewable energy market in the Western Balkan countries, Renewable and Sustainable Energy Reviews, 2011, Vol. 15, No Issue 6, pp. 3187-3195, ISSN 1364-0321, UDK: doi: 10.1016/j.rser. 2011.04.11, Elsevier
3. Tešić Z., Lalić D., Ćosić I., Mitrović V.: Integration of information for manufacturing shop control, Strojniski vestnik = Journal of Mechanical Engineering, 2010, Vol. 56, No 3, pp. 217-223, ISSN 0039-2480
4. Grubic-Nesic, L., Konja, V., & Lalic, D. (in press, 2012). Leadership in Learning Organizations. Metalurgia international, 17(12)
5. Konja, V., Grubic-Nesic, L., & Lalic, D. (in press, 2012). Leader-member Exchange Influence on Organizational Commitment among Serbian Hospital Workers. Healthmed, 6(11)
6. Lalić D., Marjanović U., Lalić B.: The influence of social networks on communication satisfaction within the organizations. In: M.M. Cruz-Cunha, P. Goncalves, N. Lopes, E.M. Miranda and G.D. Putnik, ed. Handbook of Research on Business Social Networking: Organizational, Managerial, and Technological Dimensions., New York, Business Science Reference (IGI Global), 2012, str. 545-566, ISBN 978-1-61350-168-9
7. Lalic, D., Gajic, S., & Konja, V. (2012). Social Media influence on Mass Customization and Personalization process. 5th International conference on Mass Customization and Personalization in Central Europe (MCP - CE 2012), 19-21 Sept., Novi Sad, Serbia

Representative refferences (minimum 5, not more than 10)

8.	Danijela Lalic, REACHING FURTHER WITH ONLINE COMMUNICATION STRATEGIES OF ORGANIZATIONS , CASE STUDY: “SECOND LIFE “ - SUCCESSFUL EXAMPLES OF ORGANIZATION’S ONLINE COMMUNICATION STRATEGIES, (Online proceedings: Web strana: http://www.onlinecommunicators.org/Seminars/IAOC-Conference-Agenda.cfm), IAOC Conference in Washington, DC, International Association of Online Communicators, 1-2 October, 2009, Washington, DC, USA.
9.	Danijela Lalic, Danijela Gracanin, Dragan Varagic: PERSONALIZATION OF INTERNET CONTENT, (str. 125-131), ISBN 978- 86-7892-114-8, 3th International Conference on Mass Customization and Personalization in Central Europe, (MCP-CE 2008), Palic, Serbia, 3-6 June 2008.
10.	Lalić D.: A HOLISTIC VIEW OF TECHNOLOGY AND MANAGEMENT DEVELOPMENT TOWARDS SUSTAINABLE BUSINESSES, 14. International Research/Expert Conference "Trends in the Development of Machinery and Associated Technology" - TMT, Mediterranean Cruise, 11-18 Septembar, 2010

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	2	International :	3

Science, arts and professional qualifications

Name and last name:	Lazarević M. Milovan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	11.11.2000
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2009	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2006	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	2000	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	EOS19	Dismantling and recycling technologies	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
2.	M316	Production Systems	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
3.	II1012	Assembly Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	II1017	Production System Design	(I10) Industrial Engineering, Undergraduate Academic Studies
5.	II1037	Disassembly and recycling technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
6.	II1053	Production Systems	(F00) Graphic Engineering and Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
7.	IM1027	Production systems	(I20) Engineering Management, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
8.	IM1114	Energy Flows in the Enterprise	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1119	Product management at end of life	(I20) Engineering Management, Undergraduate Academic Studies
10.	EI504	Management of Small and Medium Enterprises	(MR0) Measurement and Control Engineering, Master Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
11.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
12.	IMDS56	Product traceability during the lifetime	(I12) Industrial Engineering, Specialised Academic Studies
13.	IMDS57	Strategic Planning and Designing Procedures and Systems at the End of Product Lifecycle	(I12) Industrial Engineering, Specialised Academic Studies
14.	IMDS93	Virtual Enterprises and Collaborative Systems	(I22) Engineering Management, Specialised Academic Studies
15.	MBA411	Business intelligence concepts	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
16.	PLM02	Product Development and Management in PLM	(I10) Industrial Engineering, Master Academic Studies (I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
17.	PLM06	Technologies for Disposal at the Products End-Of-Life	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
18.	I907	Automated Assembly Systems for High Accuracy	(H00) Mechatronics, Master Academic Studies (PM0) Production Engineering, Master Academic Studies
19.	IIDR5S	Advanced Engineering Technologies	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (M50) Energy Management, Master Academic Studies
20.	IIDS10	Effective technological and production structures	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
21.	IM2102	Manufacturing strategy (KAIZEN, LEAN, KANBAN, EFPS)	(I10) Industrial Engineering, Master Academic Studies (M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
22.	IM2120	Virtual Enterprises	(I20) Engineering Management, Master Academic Studies
23.	IM2124	Production and Service Systems	(H00) Mechatronics, Master Academic Studies (M50) Energy Management, Master Academic Studies
24.	PLM02	Applied Product Development	(I20) Engineering Management, Specialised Professional Studies
25.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	IMDR56	Traceability of Product Lifecycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
27.	IMDR57	Strategic Planning and Designing Procedures and Systems at the End of Product Lifecycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
28.	IMDR93	Virtual Enterprises and Collaborative Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
29.	IMDR85	Effective technological and production structures	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Vukelić Đ., Ostojić G., Stankovski S., Lazarević M., Tadić B., Hodolič J., Simeunović N.: Machining fixture assembly/disassembly in RFID environment, Assembly Automation, 2011, Vol. 31, No 1, pp. 62-68, ISSN 0144-5154
2.	Stankovski S., Ostojić G., Tarjan L., Škrinjar D., Lazarević M. : IML Robot Grasping Process Improvement (Article in press, Date of acceptance 14. March 2010), Iranian Journal of Science & Technology, Transactions B, 2011, ISSN 1028-6284
3.	Ostojić G., Lazarević M., Stankovski S., Ćosić I. : RFID Technology Application in Disassembly Systems , Strojniski vestnik = Journal of Mechanical Engineering, 2008, Vol. 54, Broj 11, str. 759-767, ISSN 0039- 2480, UDK: 658.5
4.	Stankovski S., Lazarević M., Ostojić G., Ćosić I., Purić R. : RFID Technology in Product/Part Tracking During the Whole Life Cycle , Assembly Automation, 2009, Vol. 29, Broj 4, str. 364-370, ISSN 0144-5154
5.	Lazarević M., Ostojić G., Ćosić I., Stankovski S., Vukelić Đ., Zečević I.: Product lifecycle management (PLM) methodology for product tracking based on radio-frequency identification (RFID) technology, Scientific Research and Essays, 2011, Vol. 6, No 22, pp. 4776-4787, ISSN 1992-2248
6.	Ostojić G., Stankovski S., Vukelić Đ., Lazarević M., Hodolič J., Tadić B., Odri S.: Implementation of automatic identification technology in a process of fixture assembly/disassembly, Strojniški vestnik - Journal of Mechanical Engineering, 2011, Vol. 57, No 11, pp. 819-825, ISSN 0039-2480
7.	Lazarević M., Ostojić G., Stankovski S., Ćosić I.: Postupak upravljanja proizvodom u celokupnom životnom veku korišćenjem RFID taga, Broj priznatog patenta: 51796, datum priznavanja 24.10.2011. godine., 2011
8.	Milovan Lazarević, Gordana Ostojić, Stevan Stankovski, Marija Rakić-Skoković: Implementation of RFID Tecnology In Disassembly and Recycling Systems, Infoteh Jahorina 2007, Srpsko Sarajevo, Republika Srpska: Mašinski fakultet, Srpsko Sarajevo, 28-30 mart, 2007, str. 151- 155, ISBN 99938-624-2-8.
9.	Ostojić G., Stankovski S., Vukelić Đ., Lazarević M., Križan P.: Maintenance with the usage of RFID technology, Journal ERIN, 2010, Vol. 3, No 2, pp. 2-7, ISSN 1337-9089
10.	Stankovski S., Ostojić G., Lazarević M., Popović B., Mijić D.: RFID TECHNOLOGY, PRIVACY AND SECURITY, Facta universitatis - series: Mechanical Engineering, 2010, Vol. 8, No 1, pp. 57-62, ISSN 0354–2025

Summary data for teacher's scientific or art and professional activity:

Quotation total :	11			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	4	International :	3

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Lisov R. Milimir
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2012		Production Systems, Organization and Management
PhD thesis	2006	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	1978	Faculty of Economics - Beograd	Mathematics
Bachelor's thesis	1975	Faculty of Mathematics - Beograd	Mathematics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IM1011	Applied Operational Research	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
2.	IM1024	Risk Management and insurance	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1706	Actuerial Mathematics	(I20) Engineering Management, Undergraduate Academic Studies
4.	URZP80	Basic principals of insurance	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
5.	IMDS53	Selected Chapters in Life Insurance	(I22) Engineering Management, Specialised Academic Studies
6.	OIR001	Basic insurance	(I20) Engineering Management, Specialised Professional Studies
7.	OIR005	Tehničke osnove osiguranja	(I20) Engineering Management, Specialised Professional Studies
8.	IM2707	Methods for the analysis of insurance risk	(I20) Engineering Management, Master Academic Studies
9.	IM2713	Rates of Insurance Premiums	(I20) Engineering Management, Master Academic Studies
10.	IM2717	Management of strategic and operational risks of insurance companies	(OM1) Mathematics in Engineering, Master Academic Studies
11.	IM2719	Loss Assessment	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
12.	IMDR53	Selected Chapters in Life Insurance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Lisov, M: Zarkovic, N; Mrksic, D; SITUATION AND POSSIBILITIES OF IMPROVEMENT OF VOLUNTARY PENSION INSURANCE IN SERBIA AS A DEVELOPING COUNTRY, African Journal of Business Management, Vol. 4 (10), August 2010, pp 2075-2086
2. Zarkovic, N., Lisov, M., Mrksic, D., Investmants of Serbian Insurance companies, Economic Research, (2012), vol. 25, No 3
3. Zarkovic, N; Lisov, M; Mrksic, D: NATIONAL BANK AS INSURANCE SUPERVISOR IN SERBIA AS A DEVELOPING COUNTRY, African journal of business management, (2012), Vol. 6, No 8, pp. 2816-2824
4. Rakonjac-Antic, T; Lisov,M; Rajic, V: Sustainabality problems of the public pension and disability system, Part II, Chapter 13 in monograph "Achieved Results and Prospects of Insurance Market Development in Modern World", Faculty of Economy of the University of Belgrade, 2012, pp. 213-228, ISBN: 978-86403-1222-6
5. Lisov, M: PRIVATNO PENZIJSKO OSIGURANJE, Novi Sad, 2006, 223 str, CIP 368.914.2, ISBN 86 – 907827-2-9
6. Lisov, M: OSIGURANJE ŽIVOTA – DINAMIČKI SISTEM RENTNIH OSIGURANJA , Osiguranje i privreda – časopis za teoriju i praksu osiguranja, Zagreb, 1980, 28 - 34 str.
7. Lisov, M: OCENA KRITERIJUMA ZA IZBOR MEHANIČKIH METODA IZRAVNANJA SIROVIH VEROVATNOĆA SMRTNOSTI, Osiguranje i privreda – časopis za teoriju i praksu osiguranja, Zagreb, 1989, 76 - 81 str.
8. Lisov M.: EKONOMSKE I TEHNIČKE OSNOVE OSIGURANJA, Novi Sad, Fakultet tehničkih nauka, 2010, str. 52-261, ISBN 978-86-7892-234-3, UDK: 368(075.8)

Representative refferences (minimum 5, not more than 10)	
9.	Lisov, M; Bukumirić ,G: POSLOVANJE OSIGURAVAJUĆIH KOMPANIJA ZA ŽIVOTNO OSIGURANJE U USLOVIMA KRIZE, Osmi međunarodni simpozijum iz osiguranja: "Problemi poslovanja osiguravajućih kompanija u uslovima krize", Zlatibor, maj 2010, 165-179 str, ISBN: 978-86-84309-26-8
10.	Lisov, M: METODE REZERVACIJE NASTALIH NEPRIJAVLJENIH ŠTETA, Sedmi međunarodni simpozijum iz osiguranja: "Osiguranje i globalna finansijska kriza", Zlatibor, 2009, 505-518 str, ISBN 978-86-84309-22-0

Summary data for teacher's scientific or art and professional activity:				
Quotation total :	22			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	0	International :	0

Science, arts and professional qualifications

Name and last name:	Maksimović M. Rado
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	12.06.1979
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2008	University of Novi Sad - Novi Sad	Production Systems, Organization and Management
PhD thesis	1998	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	1989	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1978	Faculty of Technical Sciences - Novi Sad	Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	Z421	Operacioni menadžment(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
2.	BM118C	Medical management	(BM0) Biomedical Engineering, Undergraduate Academic Studies
3.	IM1021	Developmental Processes in Company	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1031	Enterprise's organization	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
5.	IM1113	Improvement of products and processes	(I20) Engineering Management, Undergraduate Academic Studies
6.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
7.	IMDS60	Enterprise Complexity and Flexibility	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
8.	IMDS63	Intelligent Organisation	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
9.	IMDS65	Entrepreneurship and Organizational Development	(I22) Engineering Management, Specialised Academic Studies
10.	I901	Manufacturing performace measurement	(I10) Industrial Engineering, Master Academic Studies
11.	I907	Automated Assembly Systems for High Accuracy	(H00) Mechatronics, Master Academic Studies (PM0) Production Engineering, Master Academic Studies
12.	IIDS10	Effective technological and production structures	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
13.	IIDS19	Organizational structures	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
14.	IIDS5	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies
15.	IIDS9	Effective Production and Service Systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
16.	IM2102	Manufacturing strategy (KAIZEN, LEAN, KANBAN, EFPS)	(I10) Industrial Engineering, Master Academic Studies (M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
17.	IM2103	New technologies in engineering and management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
18.	IM2113	Design of enterprise's organization	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
19.	IM2114	Enterprise's performances	(I20) Engineering Management, Master Academic Studies
20.	IM2119	Layout and location of the enterprise	(I20) Engineering Management, Master Academic Studies
21.	IM2321	Management of project oriented enterprises	(I20) Engineering Management, Master Academic Studies
22.	IMDS69	Selected chapters in enterprise's design, organization and control	(I22) Engineering Management, Specialised Academic Studies
23.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	IMDR12	Organizational structures	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	IMDR31	Effective Production and Service Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	IMDR60	Enterprise Complexity and Flexibility	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
27.	IMDR63	Intelligent Organisation	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
28.	IMDR65	Entrepreneurship and Organizational Development	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
29.	IMDR5	Selected chapters in enterprise's design, organization and control	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
30.	IMDR69	Selected chapters of enterprise's management and control	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
31.	IMDR85	Effective technological and production structures	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
32.	ZRD27A	Operations management in the security and occupational safety	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Njegomir V., Maksimović R.: The overview of some basic issues in insurance matket - the case of Serbian insurance risk transfer market, Transformations in Business & Economics (TIBE), 2012, Vol. 11, No 2, pp. 51-69, ISSN 1648-4460
2. Marković V., Maksimović R.: A contribution to continual software service improvement based on the six-step service improvement method, INTERNATIONAL JOURNAL OF SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 2012, Vol. 22, No 4, pp. 549-569, ISSN 0218-1940
3. Zelenović, D., Ćosić, I., Maksimović, R.: IISE - APPROACH IN DEVELOPMENT OF EFFECTIVE MANUFACTURING SYSTEMS - COMPANIES, U: Suresh, N.C, Kay, M.J.: GROUP TECHNOLOGY & CELLULAR MANAGEMENT - A state of-The-Art Synthesis of Research & Practice, New York: Cluwer Pres, Buffalo - New York, 1998, ISBN 0-7923-8080-0. pp. 517- 536.
4. Maksimović, R, Lalić, B: Flexibility and Complexity of Effective Enterprises, Strojniški vestnik - Journal of mechanical engineering, 2008, Vol. 54, No. 11, pp. 768- 782, UDK: 658.51, ISSN 0039-2480
5. Maksimović, R., Stankovski, S., Ostojić, G., Petrović, S, Ratković, Ž.: Complexity and Flexibility of Production Structures, Journal of Scientific and Industrial Research, 2009, 101-105, ISSN 0022-4456
6. Borocki J., Ćosić I., Lalić B., Maksimović R.: Analysis of Company Development Factors in Manufacturing and Service Company: a Strategic Approach, Strojniski vestnik = Journal of Mechanical Engineering, 2011, Vol. 57, No 1, pp. 55-68, ISSN 0039-2480, UDK: DOI:10.5545/sv-jme.2010.030
7. Marović, B., Njegomir, V., Maksimović, R.: The implications of the financial crisis to the insurance industry - Global and regional perspective, Economic research, 2010, Vol. 23, No. 2, 127-141, ISSN 1331-677X.
8. Obadović M., Maksimović R., Obadović M.: The estimate of the market risk by the application of historical simulation method in the period of growth of stock exchange indices on Belgrade stock exchange, Economic research, 2010, Vol. 23, No 3, pp. 82-95, ISSN 1331-677X, UDK: UDK 330.322:336.76
9. Djuric, Ž., Maksimović, R., Adamović, Ž.: Key performance indicators in a joint-stock company, AFRICAN JOURNAL OF BUSINESS MANAGEMENT, 4 (6): 890-902, 2010
10. Radišić, O., Radišić, M., Maksimović, R. et al. 2012. Industrial Cogeneration Appliance--An Example of a Drilling Rig. J Can Pet Technol 51 (6): 487-492. SPE-157689-PA. http://dx.doi.org/10.2118/157689-PA.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	8			
Total of SCI(SSCI) list papers :	11			
Current projects :	Domestic :	2	International :	1

Science, arts and professional qualifications

Name and last name:	Marić B. Branislav
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2009
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2011	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	1995	Faculty of Technical Sciences "Mihajlo Pupin" in Zrenjanin - Zrenjanin	Organization Science
Magister thesis	1992	Faculty of Technical Sciences - Novi Sad	Organization Science
Bachelor's thesis	1977	Faculty of Technical Sciences - Novi Sad	Organization Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	I914	Project Management	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies
2.	M317	Economy	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
3.	II121	Principles of economics	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
4.	IM1014	Company Economics	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
5.	IM1027	Production systems	(I20) Engineering Management, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
6.	IM1102	Investment Management	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1419	Strategic resource allocation and planning	(I20) Engineering Management, Undergraduate Academic Studies
8.	IMDS63	Intelligent Organisation	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
9.	IMDS88	Planning and implementing cost structure of the investment cycle	(I22) Engineering Management, Specialised Academic Studies
10.	MBA303	Economics for Managers	(IB0) Engineering Management - MBA, Specialised Professional Studies
11.	LIM33	Logistic Economics	(LIM) Logistic Engineering and Management, Master Academic Studies
12.	IM2102	Manufacturing strategy (KAIZEN, LEAN, KANBAN, EFPS)	(I10) Industrial Engineering, Master Academic Studies (M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
13.	IM2103	New technologies in engineering and management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
14.	IM2122	The rating company profitability	(I20) Engineering Management, Master Academic Studies
15.	IM2414	Technical Analyses and the Trading Systems	(I20) Engineering Management, Master Academic Studies
16.	IM2418	Support to management decision making	(I20) Engineering Management, Master Academic Studies
17.	IM2424	Investment management	(M50) Energy Management, Master Academic Studies
18.	IM2425	Economics of the Firm	(M50) Energy Management, Master Academic Studies
19.	IMDR63	Intelligent Organisation	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
20.	IMDR88	Planning and implementing cost structure of the investment cycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Kiurski J., Marić B., Adamović D., Mihailović A., Grujić S., Oros I., Krstić J.: Register of hazardous materials in printing industry as a tool for sustainable development management, Renewable and Sustainable Energy Reviews, 2012, Vol. 16, No 1, pp. 660-667, ISSN 1364-0321, UDK: doi:10.1016/j.rser.2011.08.030
2.	Marić B., Dobromirov D., Radišić M.: Researching the dependence between the dynamic indicators of investment profitability, African Journal of Business Management, 2011, Vol. 5, No 13, pp. 5076-5082, ISSN 1993-8233
3.	Radišić M., Marić B., Dobromirov D.: SMEs and entrepreneurs investments' profitability effects within the transition period in the Republic of Serbia, African Journal of Business Management, 2011, Vol. 5, No 7, pp. 2654-2659, ISSN 1993-8233
4.	Marić B., Demko-Rihter J., Mitrović V., Rovčanin M.: Functional correlations between the efficiency indicators of investments, African Journal of Business Management, 2011, Vol. 5, No 7, pp. 2979-2984, ISSN 1993-8233
5.	Marić B., Kamberović B., Radlovački V., Delić M., Zubanov V.: Observing the dependence between dynamic indicators of investment profitability - Relative net present value and internal rate of return, African Journal of Business Management, 2011, Vol. 5, No 26, pp. 331-337, ISSN 1993-8233
6.	Marić B., Ivanišević A., Mitrović S., Sreto A., Mihailo R.: Analysis of internal rate of return on investments: Dynamic and static approach, African Journal of Business Management, 2011, Vol. 5, No 8, pp. 3269-3273, ISSN 1993-8233
7.	Organizacija preduzeća, Fakultet za preduzetni menadžment, Novi Sad, 2006.
8.	Upravljanje projektima, Fakultet za preduzetni menadžment, Novi Sad, 2000.
9.	Upravljanje investicijama, Fakultet tehničkih nauka, 2010.
10.	Osnove organizacije rada, Fakultet tehničkih nauka, 1982.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	1	International :	0

Science, arts and professional qualifications

Name and last name:	Mihailović P. Biljana
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.03.1999
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	2009	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	2003	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1998	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E135	Probability, Statistics and Stochastic Processes	(MR0) Measurement and Control Engineering, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
2.	E212	Mathematical Analysis 1	(E20) Computing and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
3.	E213	Discrete Mathematics and Linear Algebra	(E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
4.	E224A	Probability and Stochastic Processes	(E20) Computing and Control Engineering, Undergraduate Academic Studies (ES0) Power Software Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
5.	EOS07	Mathematics 2	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
6.	M102	Mathematics 1	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
7.	E102	Mathematical Analysis 1	(ES0) Power Software Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
8.	BMI91	Mathematics 1	(BM0) Biomedical Engineering, Undergraduate Academic Studies
9.	BMI92	Mathematics 2	(BM0) Biomedical Engineering, Undergraduate Academic Studies
10.	E102A	Mathematical Analysis 1	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
11.	IM1423	Financial Mathematics	(I20) Engineering Management, Undergraduate Academic Studies
12.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
13.	I004/S	Statistical Quantitative Methods	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
14.	OIR009	Primenjena aktuarska matematika	(I20) Engineering Management, Specialised Professional Studies
15.	ZR503	Statistical Advanced Models	(Z01) Safety at Work, Master Academic Studies
16.	D0M07	Mathematical Foundations of Fuzzy Systems	(OM1) Mathematics in Engineering, Doctoral Academic Studies
17.	D0M21	Fuzzy Systems and Their Applications	(OM1) Mathematics in Engineering, Doctoral Academic Studies
18.	D0M49	Aggregation Functions	(OM1) Mathematics in Engineering, Doctoral Academic Studies
19.	D0M50	Fuzzy Measures and Integrals	(OM1) Mathematics in Engineering, Doctoral Academic Studies
20.	D0M51	Large Deviations Principles	(OM1) Mathematics in Engineering, Doctoral Academic Studies
21.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. E. Pap, B. Mihailović: A representatation of a comonotone-v-additive and monotone functional by two Sugeno integrals, Fuzzy Sets and Systems 155, (2005) 77-88
2. B. Mihailović, E. Pap: Sugeno integral based on absolutely monotone real set functions, Fuzzy Sets and Systems, Vol 161, Issue 22, (2010) 2857-2869
3. B. Mihailović, E. Pap: Asymmetric integral as a limit of generated Choquet integrals based on absolutely monotone real set functions, Fuzzy Sets and Systems 181, (2011) 39-49.
4. B. Mihailović, E. Pap: Asymmetric general Choquet integrals, Acta Polytechnica Hungarica, Volume 6, Issue Number 1, (2009) 161-173.
5. Kalina M., Manzi M., Mihailović B.: Choquet integrals and T-supermodularity, E. Pap (Ed.): Intelligent Systems: Models and Applications, TIEI 3, DOI: 10.1007/978-3-642-33959-2 4 c Springer-Verlag Berlin Heidelberg , (2013) 61-75.

Representative refferences (minimum 5, not more than 10)

6.	B. Mihailović, Lj. Nedović, T. Grbić : The induced Sugeno integral-based operator w.r.t bi-fuzzy measures, Journal of Electrical Engineering, Vol.54, No. 12/s, (2003) 76-79.
7.	B. Mihailović, E. Pap: Non-monotonic set functions and general fuzzy integrals, Proceedings of SISY 2008, Subotica, (2008) 371-374.
8.	B. Mihailović: On the class of symmetric S-separable aggregation functions Proceedings of AGOP 2007, Ghent, Belgium, (2007) 187-191.
9.	B. Mihailović, E. Pap: Decomposable signed fuzzy measures, Proceedings of EUSFLAT 2007, Ostrava, Czech Republic, (2007) 265-269.
10.	B. Mihailović, M. Manzi: On the asymmetric Shilket-like integral, Proceedings of AGOP2011, Benevento, Italy, (2011) 73-77.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	10			
Total of SCI(SSCI) list papers :	4			
Current projects :	Domestic :	2	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Milisavljević M. Stevan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.02.2007
Scientific or art field:	Quality, Effectiveness and Logistics

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
PhD thesis	2012	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
Master's thesis	2006	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
Bachelor's thesis	2006	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II1016	Reliability of technical systems and Maintenance	(I10) Industrial Engineering, Undergraduate Academic Studies
2.	IM1030	Integral Systems Support - Logistic	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
3.	IM1036	Reliability Theory	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1049	Supply chain Management	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1614	Organization and Management of Logistic	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1814	Industrial Customer Relationship Management	(I20) Engineering Management, Undergraduate Academic Studies
7.	I501	Risk Management	(I10) Industrial Engineering, Master Academic Studies
8.	IMDS95	Trends in Customer Relationship Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
9.	LIM05	Fundamentals of Logistic Management	(LIM) Logistic Engineering and Management, Master Academic Studies
10.	LIM16	Production Logistics	(LIM) Logistic Engineering and Management, Master Academic Studies
11.	LIM19	Customer Relationship Management	(LIM) Logistic Engineering and Management, Master Academic Studies
12.	LIM30	Inventory Planning and Management	(LIM) Logistic Engineering and Management, Master Academic Studies
13.	LIM31	Reverse and Green Logistics	(LIM) Logistic Engineering and Management, Master Academic Studies
14.	IIDS12	Quality and organizational performance	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
15.	IIDS30	Trends in the environmental management systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
16.	IIDS7	Selected topics in quality engineering and logistics	(I12) Industrial Engineering, Specialised Academic Studies
17.	IM2607	Risk management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
18.	IM2615	Lean Logistics	(I20) Engineering Management, Master Academic Studies
19.	IM2618	Transportation management	(I20) Engineering Management, Master Academic Studies
20.	IM2619	Stock planning and management	(I20) Engineering Management, Master Academic Studies
21.	IM2621	Castumer Relationship Management	(I20) Engineering Management, Master Academic Studies
22.	IM2815	Logistics in Engineering Marketing	(I20) Engineering Management, Master Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
23.	IMDS74	Selected Topics in Quality Management and Logistics	(I22) Engineering Management, Specialised Academic Studies
24.	IMDR94	Trends in the environmental management systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	IMDR95	Trends in Customer Relationship Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	IMDR74	Selected Topics in Quality Management and Logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
27.	IMDR79	Selected topics in quality engineering and logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
28.	IMDR83	Quality abd organisational performance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Brkljač N., Šević D., Beker I., Kesić I., Milisavljević S.: Procedure for treatment of hazardous waste by MID-MIX procedure in Serbia, International Journal of the Physical Sciences, 2012, Vol. 7, No 18, pp. 2639-2646, ISSN 1992-1950
2.	Mitrović S., Grubić-Nešić L., Milisavljević S., Melović B., Babinkova Z.: Manager's Assessment of Organizational Culture, E M Ekonomie a Management, ISSN 1212-3609.
3.	Mitrović S., Milisavljević S., Ćosić I., Leković B., Grubić-Nešić L., Ivanišević A.: Change in leadership styles in a transitional economy: A serbian case study, African Journal of Business Management, 2011, Vol. 5, No 9, pp. 3563-3569, ISSN 1993-8233
4.	Melović B., Mitrović S., Milisavljević S., Pejanović R., Ćelić Đ.: Research of consumption and competitiveness of homemade products for manufacturing improvements: A case study from Montenegro, African Journal of Agricultural Research, 2012, Vol. 7, pp. 3757-3764, ISSN 1991-637X
5.	Milisavljević S.: Razvoj modela sistema upravljanja odnosima sa korisnicima u organizacijama u Srbiji, Novi Sad, Fakultet tehnickih nauka, 2012
6.	Grubić-Nešić L., Mitrović S., Melović B., Milisavljević S.: Research among Employees in the Agricultural Sector, HealthMED, 2013, ISSN 1840-2991
7.	Milisavljević S., Grubić-Nešić L.: Doprinos sistema kvaliteta pozicioniranju preduzetništva, 2. Preduzetnička konferencija "Zapošljavanje kroz prizmu preduzetništva", Podgorica: Ekonomski fakultet Podgorica, 18 Maj, 2012, ISBN 978-86-80133-56-0
8.	Mirković M., Ćulibrk D., Anderla A., Stefanović D., Milisavljević S.: A framework for obtaining publicly available geo-referenced video meta-data, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Fakultet tehničkih nauka, 14-16 Septembar, 2011, pp. 223-228, ISBN 978-86-7892-341-8
9.	Grubić-Nešić L., Mitrović S., Milisavljević S.: Personal prerequisities of entrpreneurial engagement, 2. Preduzetnička konferencija "Zapošljavanje kroz prizmu preduzetništva", Podgorica: Ekonomski fakultet Podgorica, 18 Maj, 2012, ISBN 978-86-80133-56-0
10.	I. Beker, D.Šević, S. Milisavljević "Uporedna analiza zahteva standarda ISO 14001:2004 i standarda ISO 14001:1996

Summary data for teacher's scientific or art and professional activity:

Quotation total :	2			
Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	2	International :	2

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Mirković R. Milan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.01.2007
Scientific or art field:	Information-Communication Systems

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
PhD thesis	2012	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
Master's thesis	2005	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
Bachelor's thesis	2005	Faculty of Technical Sciences - Novi Sad	Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	Z201	Fundamentals of Computer Technologies	(Z20) Environmental Engineering, Undergraduate Academic Studies
2.	Z201A	Fundamentals of Computer Technologies	(Z01) Safety at Work, Undergraduate Academic Studies
3.	II1002	Computer Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	IM1010	Fundamentals of Information Technologies	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1038	Introduction to Business Intelligence Systems	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1514	Web-oriented Technologies and Systems	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1515	Mobile information technologies	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1813	Multimedia and global media	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1815	Industrial Internet marketing	(I20) Engineering Management, Undergraduate Academic Studies
10.	HR013	Knowledge Economy	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	IMDS55	Data Mining	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
12.	MBA309	Human Resource Management in Knowledge Economy	(IB0) Engineering Management - MBA, Specialised Professional Studies
13.	MBA411	Business intelligence concepts	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
14.	MBA415	Development of services, products and marketing of technological innovation	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
15.	LIM02	Business Information Systems	(LIM) Logistic Engineering and Management, Master Academic Studies
16.	I835	Data mining methods	(I10) Industrial Engineering, Master Academic Studies
17.	I913	Expert systems and tools for knowledge management	(I10) Industrial Engineering, Master Academic Studies
18.	IIDS8	Selected chapters from Information, management and communication systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies
19.	IM2518	Captology - procedures and methods	(I20) Engineering Management, Master Academic Studies
20.	IM2519	Advanced Information Technology	(I20) Engineering Management, Master Academic Studies
21.	IM2520	E-commerce Procedures and Methods	(I20) Engineering Management, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
22.	IM2816	Data mining in industrial marketing	(I20) Engineering Management, Master Academic Studies
23.	IM2821	Digital products design and Human-Computer Interaction	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
24.	IMDS73	Selected chapters from Information management	(I22) Engineering Management, Specialised Academic Studies
25.	IMDR34	Raster and Image Processing Technologies in Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	IMDR55	Data Research	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
27.	IMDR73	Selected chapters from Information management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
28.	IMDR81	Selected chapters from Information, management and communication systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Mirković M., Ćulibrk D., Crnojević V.: Computational Social Networks (Chapter: Mining Geo-Referenced Community-Contributed Multimedia Data), London, Springer, 2012, str. 81-102, ISBN 978-1-4471-4053-5
2. Ćulibrk D., Mirković M., Zlokolica V., Pokrić M., Crnojević V., Kukolj D.: Salient Motion Features for Video Quality Assessment, IEEE Transactions on Image Processing, 2011, Vol. 20, No 4, pp. 948-958, ISSN 1057-7149
3. Mirković M., Ćulibrk D., Papadopoulos S., Zigkolis C., Kompatsiaris Y., McArdle G., Crnojević V.: A Comparative Study of Spatial, Temporal and Content-based Patterns Emerging in YouTube and Flickr
4. Ćulibrk D., Mirković M., Lugonja P., Crnojević V.: Mining Web Videos for Video Quality Assessment, 2. International Conference of Soft Computing and Pattern Recognition - SocPar, Pariz, 7-10 Decembar, 2010
5. Mirković M., Ćulibrk D., Anderla A., Stefanović D., Milisavljević S.: A framework for obtaining publicly available geo-referenced video meta-data, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Fakultet tehničkih nauka, 14-16 Septembar, 2011, pp. 223-228, ISBN 978-86-7892-341-8
6. Stefanović D., Mirković M., Anderla A., Drapšin M., Drid P., Radjo I.: Investigating erp systems success from the end user perspective, TTEM. Tehnics tehnologies education management, 2011, Vol. 6, No 4, pp. 1089-1099, ISSN 1840-1503
7. Stefanović D., Rakić-Skoković M., Mirković M., Anderla A., Rašić D.: Contemporary Software Business Suites as a Company's Competitive Advantage, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Faculty of Technical Sciences; Department of Industrial Engineering and Management; University of Novi Sad, 14-16 Septembar, 2011, ISBN 978-86-7892-341-8
8. Ćulibrk D., Žunić I., Mirković M., Šetrajčić I.: PRIMENA ISTRAŽIVANJA PODATAKA NA PREDVIĐANJE PERFORMANSI PROFESIONALNIH KOŠARKAŠA, 10. Naučno-stručni simpozijum INFOTEH-JAHORINA, Jahorina: Infoteh, 16-18 Mart, 2011, pp. 539-542, ISBN 978-99938-624-6-8
9. Gavrić K., Lugonja P., Mirković M., Ćulibrk D., Crnojević V.: Detecting Attractive Locations and Tourist' Dynamics Using Geo-referenced Images, 10. TELSIKS - International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, Niš, 5-8 Oktobar, 2011, ISBN 978-1-4577-2017-8
10. Gavric, K., Culibrk, D., Mirkovic, M., & Crnojevic, V. (2011). Using YouTube Data to Analyze Human Contintent-level Mobility. CASoN 2011 (pp. 207–210). Salamanca: MIR Labs.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	12			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	2	International :	3

Science, arts and professional qualifications

Name and last name:	Mitrović M. Slavica
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2005
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2011	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2007	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	2004	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E2I41	Information System Engineering	(E20) Computing and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies
2.	EOS33	Entrepreneurial management	(E01) Power Engineering - Renewble Sources of Electrical Energy, Undergraduate Professional Studies
3.	S002A	Economics	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
4.	II121	Principles of economics	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
5.	I120	Principi menadžmenta(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
6.	I201	Preduzetništvo(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
7.	II1041	Innovation and Entrepreneurship	(I10) Industrial Engineering, Undergraduate Academic Studies
8.	IM1005	Entrepreneurship	(I20) Engineering Management, Undergraduate Academic Studies (Z01) Safety at Work, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
9.	IM1007	Principles of engineering management	(I20) Engineering Management, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies (ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
10.	IM1215	Management of small and medium size enterprises	(I20) Engineering Management, Undergraduate Academic Studies
11.	IM1218	Models of open innovations and corporate entrepreneurship	(I20) Engineering Management, Undergraduate Academic Studies
12.	IMDS97	Entrepreneurial Management	(I22) Engineering Management, Specialised Academic Studies
13.	MBA304	Business Strategies	(IB0) Engineering Management - MBA, Specialised Professional Studies
14.	NIT07	Management Skills	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
15.	IMDS66	Managerial decision-making	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
16.	IMDR97	Entrepreneurial Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
17.	IMDR66	Managerial decision-making	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Mitrović, S., Grubić-Nešić, L ., Milisavljević, S., Melović, B.,. Zuzana Babinkova (in press) Manager's Assessment of Organizational Culture. E+M Ekonomie a Management ISSN 1212-3609.
2.	Slavica MITROVIĆ, Bozidar LEKOVIĆ, Valentin KONJA, Ana NEŠIĆ (in press). EMPLOYEE TIME MANAGEMENT: A CASE STUDY FROM SERBIA.Metalurgia International, ISSN 1582 – 2214. Vol. (1).
3.	Valentin KONJA, Leposava GRUBIĆ-NEŠIĆ, Slavica MITROVIĆ (2012). LEADER-MEMBER EXCHANGE: A SHORT CASE STUDY FROM A SERBIAN COMPANY. Metalurgia International, ISSN 1582 – 2214. Vol.17 (11), pp. 146-153.
4.	Melović, B., Mitrović, S., Milisavljević, S., Pejanović, R., Ćelić, Đ. (2012). RESEARCH OF CONSUMPTION AND COMPETITIVENESS OF HOMEMADE PRODUCTS FOR MANUFACTURING IMPROVEMENT: CASE STUDY FROM MONTENEGRO. African Journal of Agricultural Research. ISSN 1991-637X .Vol. 7(26), pp. 3757-3764.
5.	S. Mitrovic, S. Milisavljevic, I. Cosic, B. Lekovic, L. Grubic-Nesic, A. Ivanisevic: Changes in leadership styles in a transitional economy: A Serbian case study, African Journal of Business Management, Vol. 5(9), pp. 3563-3569, 4 May 2011. ISSN 1993-8233 Academic Journals.
6.	Mitrović, S., Nikolić, J., Milisavljević, S., Ćosić, I. (2012). Factors influencing managerial decision-making in industrial systems, International symposium on industrisl enigneering-SIE, Belgrade. Proceeding page 67-73. ISBN 978-86-7083-758-4 (COBISS:SR-ID 191329292).
7.	Mitrović, S., Melović, B., Ćosić, I. (2012). ENTREPRENEURIAL EDUCATION AS AN EMPLOYMENT-INFLUENCING FACTOR. International entrepreneurship conference „Recruitment in the light of entrepreneurship“, organized by Faculty of Economics, Podgorica, Montenegro. ISBN 978-86-80133-56-0
8.	Mitrović, S., Milisavljević, S., Melović, B., Grubić-Nešić, L. (2012). Strategic management in the function of overcoming economical crizes, 17 th International Scientific Symposium Strategic management and Decision Support Systems in Strategic Management, Palic-Subotica. ISBN 978-86-7233-305-3 (COBISS.SR-ID 250924295).
9.	Leposava GRUBIC-NESIC, Sanja VRNJES, Biljana RATKOVIC-NJEGOVAN, Slavica MITROVIC (2012). ATTITUDES OF THE EMPLOYEES ABOUT THE ORGANIZATIONAL RESTRUCTURING: A SAMPLE OF ORGANIZATIONS IN SERBIA. Metalurgia International, ISSN 1582 – 2214. Vol.17 (12), pp. 153-160.
10.	Lošonc (Losoncz) A., Ivanišević A., Mitrović S.: Strukturalna kriza: forme i uzroci, Novi Sad, Fakultet tehnickih nauka, 2012, str. 1-232, ISBN 978-86-7892-375-3, UDK: 268964871

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	8			
Current projects :	Domestic :	2	International :	0

Science, arts and professional qualifications

Name and last name:	Morača D. Slobodan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2000
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Magister thesis	2005	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1999	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	URZP51	Strategy of Intervention	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
2.	ZR305	Risks and Hazards at Work and in the Working Environment	(Z01) Safety at Work, Undergraduate Academic Studies
3.	I201	Preduzetništvo(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
4.	II1019	Project Management	(I10) Industrial Engineering, Undergraduate Academic Studies
5.	IM1028	Fundamentals of Project Management	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1047	Planning and enterprises performance analysis	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1121	Industrial Clusters	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1306	Project Management	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1313	Project cost management	(I20) Engineering Management, Undergraduate Academic Studies
10.	IM1314	Computer aided project management	(I20) Engineering Management, Undergraduate Academic Studies
11.	IM1316	Project Cycle Management	(I20) Engineering Management, Undergraduate Academic Studies
12.	ZR402A	Protection System Design	(Z01) Safety at Work, Undergraduate Academic Studies
13.	IMDS96	Project portfolio management	(I22) Engineering Management, Specialised Academic Studies
14.	ZP512	Protection and Rescue Plans	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
15.	IM2313	Planning, guidance and control of the project	(I20) Engineering Management, Master Academic Studies
16.	IM2317	IT Project management	(I20) Engineering Management, Master Academic Studies
17.	IM2320	Project Auditing	(I20) Engineering Management, Master Academic Studies
18.	IMDS71	Selected topics of project management	(I22) Engineering Management, Specialised Academic Studies
19.	UP001	Computer Supported Project Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
20.	UP002	Applied Project Cycle Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
21.	UP004	Applied IT Project Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
22.	IMDR96	Project portfolio management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
23.	IMDR71	Selected topics of project management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	ZRD213	Current state and development tendencies of quality management of work environment	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Moraca Slobodan Hadzistevic Miodrag Drstvensek Igor Radakovic Nikola, Application of Group Technology in Complex Cluster Type Organizational Systems, STROJNISKI VESTNIK-JOURNAL OF MECHANICAL ENGINEERING, ISBN 0039-2480, (2010), vol. 56 br. 10, str. 663-675
2.	Hadžistević Miodrag; Morača Slobodan; Networks and Quality Improvement; International Journal for Quality Research ISSN: 1800-6450 Detalji Vol. 3, No. 4, Str. 353-361
3.	Demko-Rihter J., Gračanin D., Morača S.: The importance of the business environment for the liquidity of SMEs and entrepreneurs - case of Serbia, 4. International Conference for Entrepreneurship, Innovation and Regional Development ICEIRD, Ohrid: National Centre for Development of Innovation and Entrepreneurial Learning, 5-7 Maj, 2011, pp. 172-179, ISBN 978-608-65144-1-9
4.	Ćosić Ilija; Gračanin Danijela; Morača Slobodan; Ćirić Jelena; Project Approach in Deign of Complex Organizational Structures Vol. 13, No. 1, Str. 249-252, ISBN 1840-4944, University of Zenica, Faculty of Mechanical engineering in Zenica; International Research/Expert Conference "Trends in the Development of Machinery and Associated Technology" TMT (13 ; Hammamet ; 2009)
5.	Morača Slobodan; Maksimović Rado; HOLISTIC, MANAGEMENT, AND CHANGES IN ORGANIZATION; Str. 835-841, UDK 658.5(082), ISBN 86-7780-008-5, Izdavač: University of Novi Sad, Faculty of Technical Sciences; International Scientific Conference on Industrial Systems - IS (13 ; Herceg Novi ; 2005)
6.	Morača, S., Ćosić, I. Softver za podršku odlučivanju u strateškom upravljanju preduzećem, Naziv skupa: XLVI konferencija ETRAN-a, Banja Vrućica, Detalji Str. 63-66, ISBN 86-80509-43-4, Društvo za elektorniku, telekomunikacije, računarstvo, automatiku i nuklearnu tehniku;
7.	Etos - Moris, dr Božo Sovilj, mr Slobodan Morača: Udžbenik koji obrađuje probleme poslovne etike i morala
8.	Morača Slobodan, Katić Jasna, Vulanović Srđan, Proizvodnja bio dizela - pozitivni i negativni uticaji u odnosu na zahteve standarda ISO 14000 i OHSAS 18000 Tehnika - Kvalitet, standardizacija i metrologija, vol. 8, br. 3, str. 6-10, 2008
9.	Morača Slobodan; Gračanin Danijela; Ćirić Jelena; Change Management in modern organizations; International Conference for Entrepreneurship, Innovation and Regional Development ICEIRD (3 ; NoviSad ; 2010) pp. 547-552, ISBN 978-86-7892-250-3, Izdavač: Fakultet tehničkih nauka;
10.	Morača Slobodan; Hadžistević Miodrag; Šević Dragoljub; Value Creation in Business Networks; International Conference for Entrepreneurship, Innovation and Regional Development ICEIRD (3 ; Novi Sad ; 2010) Str. 553-558, ISBN 978-86-7892-250-3, Izdavač: Fakultet tehničkih nauk;

Summary data for teacher's scientific or art and professional activity:

Quotation total :	2			
Total of SCI(SSCI) list papers :	1			
Current projects :	Domestic :	4	International :	4

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Mrkšić Lj. Dragan
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	02.10.2006
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2007	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	1984	Faculty of Law - Beograd	Legal Science
Magister thesis	1981	Faculty of Law - Beograd	Legal Science
Bachelor's thesis	1977	Faculty of Law - Beograd	Legal Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IM1009	Business Law	(I20) Engineering Management, Undergraduate Academic Studies
2.	IM1712	Management of Life Insurance	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1717	Right insurance	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1720	Communications in Insurance	(I20) Engineering Management, Undergraduate Academic Studies
5.	IMDS53	Selected Chapters in Life Insurance	(I22) Engineering Management, Specialised Academic Studies
6.	OIR006	The basis of the rights in insurance	(I20) Engineering Management, Specialised Professional Studies
7.	IMDR53	Selected Chapters in Life Insurance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Zarković, N., Mrkšic, D. and Lisov, M.: SITUATION AND POSSIBILITIES OF IMPROVEMENT OF VOLUNTARY PENSION INSURANCE IN SERBIA AS A DEVELOPING COUNTRY, African Journal of Business Management, Vol.4 (10), 18 August 2010, pp 2075-2086.
2. Mrkšić, D., Carić, S., Vitez, M.:PRIVREDNO PRAVO, CENTAR ZA PRIVREDNI CONSALTING, Novi Sad, petnaesto izdanje, 2005., str. 500,
3. Mrkšić, D., Marović, B.: OSIGURANJE I REOSIGURANJE, FINANSING CENTAR, Novi Sad, 1996.
4. Mrkšić, D., Petrović, Z.: PRAVO OSIGURANJA, FAKULTET ZA POSLOVNO PRAVO Beograd, Beograd 2004.
5. Mrkšić, D.: OSIGURANJE U TEORIJI I PRAKSI, ALEF, Novi Sad, 1990.
6. Mrkšić, D., Kostadinović, S.: KOMPANIJSKO PRAVO, FAKULTET ZA USLUŽNI BIZNIS, Novi Sad, 2004.
7. Mrkšić, D., Petrović, Z.: ŽIVOTNO OSIGURANJE, DIS PUBLIK, Beograd, 2005.
8. Mrkšić, D., Šulejić, P., Vujović, R., Žarković, N., Rašeta, J., Miloradić, J.: OSNOVI OSIGURANJA, FAKULTET ZA FINANSIJSKI MENADŽMENT I OSIGURANJE, Beograd, 2006.
9. Mrkšić, D., Miloradić, J., Žarković, N.: UVOD U OSIGURANJE I ŽIVOTNA OSIGURANJA, IKP „ZASLON“ Šabac i Monart – Sremska Mitrovica, Novi Sad, 2006.
10. Mrkšić, D.: UPRAVLJANJE OSIGURAVAJUĆIM I REOSIGURAVAJUĆIM ORGANIZACIJAMA, FAKULTET ZA FINANSIJSKI MENADŽMENT I OSIGURANJE, Beograd, 260 str., 2006.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	122			
Total of SCI(SSCI) list papers :	1			
Current projects :	Domestic :	0	International :	0

Science, arts and professional qualifications

Name and last name:	Nerandžić B. Branislav
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	20.10.2006
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2011	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2006	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2004	Faculty of Technical Sciences - Novi Sad	Engineering Management
Education Specialist Thesis	2003	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1980	Faculty of Economics - Subotica	Economic Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	ETI41	Sociology of Technique	(E02) Electronics and Telecommunications, Undergraduate Professional Studies
2.	IM1018	Management Accounting and Financial Management	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1414	Analyses of business reports	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1415	Indicators of Business Performance	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1418	Operational Audit	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1718	Controlling and Auditing in Insurance	(I20) Engineering Management, Undergraduate Academic Studies
7.	IMDS89	Controlling and Internal Audit in Corporate Governance	(I22) Engineering Management, Specialised Academic Studies
8.	IMDS90	Selected Chapters of Strategic Management Accounting	(I22) Engineering Management, Specialised Academic Studies
9.	IR001	Professional Practice of Internal Auditing	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
10.	IR002	Implementation and Execution of Internal and Operational Audit.	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	KIR001	Internal and Operational Auditing	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
12.	MBA307	European and international business and trade law	(IB0) Engineering Management - MBA, Specialised Professional Studies
13.	MBA310	Financial management with the accounting elements	(IB0) Engineering Management - MBA, Specialised Professional Studies
14.	MBA521	The European Union-development process	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
15.	MUO002	Management Accounting, Auditing and Controlling	(I20) Engineering Management, Specialised Professional Studies
16.	SZP003	Selected Chapters in Applied Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
17.	IM2117	Calculation of costs and prices of products and services	(I20) Engineering Management, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
18.	IM2419	Business in Terms of Globalization	(I20) Engineering Management, Master Academic Studies
19.	IM2426	Operational Audit and Controlling	(M50) Energy Management, Master Academic Studies (OM1) Mathematics in Engineering, Master Academic Studies
20.	IMDR89	Controlling and Internal Audit in Corporate Governance.	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
21.	IMDR90	Selected Chapters of Strategic Managment Accounting	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Perović V., Nerandžić B., Bojanić R., Živkov E., Bulatović B.: INFLENCE OF CONTROLLING THE INVESTMENT PROJECTION ERP (M) WITH PRIMARY FOCUS ON THE CASHFLOW IN THE COMPANY - , Metalurgia international, 2013, No 3 - 2013, ISSN 1582-2214
2.	Nerandžić B., Perović V.: Personality and moral character traits and acnowledging the principles of management ethics,auditing and accounting ethics, African Journal of Business Management, 2011, ISSN 1993-8233
3.	Perović V., Nerandžić B.: Controlling as a useful managament instrument in crisis times, African Journal of Business Management, 2011, ISSN 1993-8233
4.	Perović V., Nerandžić B., Bulatović B.: The Transition Processin the Context of Privatization in the Republic of Serbia (2001-2010) , Actual Problems of Economics, 2013, No 02-2013, ISSN 1993-6788
5.	Pečujlija M., Perović V., Nerandžić B.: Initiating innovation in Serbian companies' organizational cultures, African Journal of Business Management, 2010, Vol. 4, No 18, pp. 3957-3967, ISSN 1993-8233
6.	Nerandžić B.: Interna i operativna revizija , Stylos, 2007, ISBN 978-86-7473-330-1
7.	Nerandžić B., Perović V.: Upravljačko računovodstvo, Novi Sad, Fakultet tehničkih nauka, 2009, ISBN 978-86-7892-210-7
8.	Vujičić D., Nerandžić B., Perović V.: Priručnik za investicije, Novi Sad, Stilos, 2008, ISBN 978-86-7892-210-7
9.	Nerandžić B.: Sistemi internih kontrola i operativna revizija , Privredna izgradnja, 2005, No 1-2, pp. 99-112, ISSN 0032-8979
10.	Nerandžić B.: Prilaz strateškim menadžment instrumentima primenom operativne revizije , Ekonomist - Savez ekonomista Srbije i Crne Gore, 2005, Vol. 43, No 2, pp. 131-137, ISSN 0354-5253

Summary data for teacher's scientific or art and professional activity:

Quotation total :	1			
Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Nikolić T. Slavka
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.01.2000
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2012		Production Systems, Organization and Management
PhD thesis	2002	Faculty of Organizational Sciences - Beograd	Management and Business
Magister thesis	1992	Faculty of Organizational Sciences - Beograd	Organization Science
Bachelor's thesis	1978	Faculty of Technology and Metallurgy - Beograd	Technological Processes, Techno-Economic Optimization and Virtual Design

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	F109	Marketing and Entrepreneurship	(F00) Graphic Engineering and Design, Undergraduate Academic Studies
2.	II202	Marketing	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
3.	IM1015	Industrial Marketing	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1051	Market Research	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1219	Analysis of entrepreneurial environment	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1806	Behavioral models of industrial customers	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1816	Industrial brand management	(I20) Engineering Management, Undergraduate Academic Studies
8.	S1I323	Market research and customer behavior	(S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
9.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
10.	MBA415	Development of services, products and marketing of technological innovation	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	RPR003	Marketing and Strategies for Regional Development	(RPR) Regional Development Planning and Management, Master Academic Studies
12.	IM2807	Strategic industrial marketing management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
13.	IM2819	Industrial eco-marketing	(I20) Engineering Management, Master Academic Studies
14.	IMDS76	Selected topics in industrial marketing and media engineering	(I22) Engineering Management, Specialised Academic Studies
15.	IMDS82	Industrial eco-marketing management	(I22) Engineering Management, Specialised Academic Studies
16.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
17.	IMDR76	Selected topics in industrial marketing and media engineering	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
18.	IMDR82	Industrial eco-marketing management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Nikolić, T.S, Pecujlija, M.: Customer behavior in the culture of fear and short attention, African Journal of Business Managemen, 2011., Vol. 6 (9), pp. 3147-3155, 7 March, 2012, ISSN 1993-8233
2. Nikolić S., Ćosić I., Miletić A., Pečujlija M.: The effect of the 'golden ratio' on consumer behaviour, African Journal of Business Management, 2011, Vol. 5, No 20, pp. 8347-8360, ISSN 1993-8233

Representative refferences (minimum 5, not more than 10)

3.	Nikolić, T.S.: Menadžment između mislećeg i osećajnog, monografija, Fakultet tehničkih nauka, Univerzitet u Novom Sadu, 2010.
4.	Nikolić, T.S.: Stretegijski menadžment u minskom polju savremenosti, STRATEGIJSKI MENADŽMENT, ISSN 0354-8414, ID= 215489031,Vol. 10 (3), 2-5;
5.	Nikolić S.: CUSTOMIZED' CONSUMER AND CONSUMER 'INNOVATOR' IN THE LIGHT OF SOCIAL CAPITAL AND DOMINANT CULTURAL PATTERN, 5. International Conference on Mass Customization and Personalization in Central Europe MCP-CE, Novi Sad: University of Novi Sad, 19-21 Septembar, 2012, pp. 170-174
6.	Nikolić, T.S.; Mujićić,V.; Anđelić,G.: Entrepreneurship and Crisis Management – Two Sides of the Same Coin, International Conference for Entrepreneurship, Innovation and Regional Development, ICEIRD2010, ISBN 978-86-7892-250-3, COBISS.SR-ID 252076295, CD ROM, str. 559-564.
7.	Nikolić, T.S., Stamatović, M., Miladinović, S.: Marketing Reflexion in Broken Transition Mirror, International Scientific Conference CRISIS OF TRANSITION AND TRANSITION OF CRISIS 2011, B. Luka, BiH
8.	Nikolić, T.S.; Strak, M.; Gujanica, I.:Business System Between "Liposuction" and "Bodybuilding"; International Journal of Strategic management and Decision Support Systems in Strategic Management, Vol.14, No4, p.33-38;
9.	Dimitrijević(Nikolić), T. S.: Marketing u industriji teške mašinogradnje; Međunarodna naučna konferencija TEŠKA MAŠINOGRADNJA TM96, Kraljevo 1996., str. 4.51
10.	Stark M., Nikolić S.: Implementation of Complex Projects Using Constraint Programming, The International Scientific Journal of Management Information Systems, 2012, Vol. 7, No 3, pp. 11-19, ISSN 1452-774X

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Ostojić M. Gordana
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	06.03.2000
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	2008	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Intelligent Systems
Magister thesis	2003	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Intelligent Systems
Bachelor's thesis	1999	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H105	Fundamentals in Computer science	(H00) Mechatronics, Undergraduate Academic Studies
2.	H109	Fundamentals in Programming	(H00) Mechatronics, Undergraduate Academic Studies
3.	H1403	Automation of work processes	(H00) Mechatronics, Undergraduate Academic Studies
4.	H1501A	Systems for Survailance and Visualisation of Process	(H00) Mechatronics, Undergraduate Academic Studies
5.	H1504	Computer Integration of Production Systems	(H00) Mechatronics, Undergraduate Academic Studies
6.	H310	Components of technological systems	(H00) Mechatronics, Undergraduate Academic Studies
7.	BM116B	Acquisition, analysis and monitoring of medical data	(BM0) Biomedical Engineering, Undergraduate Academic Studies
8.	BM116C	Motion control	(BM0) Biomedical Engineering, Undergraduate Academic Studies
9.	BM119C	Automatic identification in bioengineering	(BM0) Biomedical Engineering, Undergraduate Academic Studies
10.	BMI106	Rehabilitation devices and systems	(BM0) Biomedical Engineering, Undergraduate Academic Studies
11.	II1009	Automatic identification systems	(I10) Industrial Engineering, Undergraduate Academic Studies
12.	II1010	Control of technical systems	(I10) Industrial Engineering, Undergraduate Academic Studies
13.	II1015	Programmable Logic Controllers (PLC)	(I10) Industrial Engineering, Undergraduate Academic Studies
14.	II1029	Computer integrated manufacturing	(I10) Industrial Engineering, Undergraduate Academic Studies
15.	II1045	Systems for measurement, surveillance and control	(I10) Industrial Engineering, Undergraduate Academic Studies
16.	II1048	Artificial intelligence in engineering	(I10) Industrial Engineering, Undergraduate Academic Studies
17.	IM1022	Fundamentals of technical systems control	(I20) Engineering Management, Undergraduate Academic Studies (M20) Mechanization and Construction Engineering, Undergraduate Academic Studies
18.	IM1035	Identification technologies in enterprises	(I20) Engineering Management, Undergraduate Academic Studies
19.	IM1117	Computer integrated manufacturing (CIM)	(I20) Engineering Management, Undergraduate Academic Studies
20.	H1503	Non Industrial Robotics and Automation in Buildings	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
21.	HDOS12	Research in the area of automatic identification technology	(I12) Industrial Engineering, Specialised Academic Studies
22.	HDOS13	Motion control and application of MEMS	(I12) Industrial Engineering, Specialised Academic Studies
23.	HDOS14	Nonindustrial automation	(I12) Industrial Engineering, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
24.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
25.	PLM09	Systems and Devices for Tracking Products Through Life Cycle	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
26.	NIT06	Advanced Technologies for Manufacturing Support	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
27.	H845	Motion control	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
28.	I903	Application of microelectromechanical systems	(I10) Industrial Engineering, Master Academic Studies
29.	I907	Automated Assembly Systems for High Accuracy	(H00) Mechatronics, Master Academic Studies (PM0) Production Engineering, Master Academic Studies
30.	IIDS6	Selected chapters in automation	(I12) Industrial Engineering, Specialised Academic Studies
31.	IM2716	Automation systems in insurance	(I20) Engineering Management, Master Academic Studies
32.	HDOK12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
33.	HDOK13	Motion control and the application of MEMS	(H00) Mechatronics, Doctoral Academic Studies
34.	HDOK14	Non-industrial Automation	(H00) Mechatronics, Doctoral Academic Studies
35.	HDOK-3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
36.	HDOKL3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
37.	HDOL12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
38.	HDOL13	Motion controla and application of MEMS	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
39.	HDOL14	Nonindustrial automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
40.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
41.	IMDR80	Selected chapters in automation	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Stankovski S., Tarjan L., Škrinjar D., Ostojić G., Šenk I.: Using a Didactic Manipulator in Mechatronics and Industrial Engineering Courses, IEEE Transactions on Education, 2010, Vol. 53, No 4, pp. 572-579, ISSN 0018-9359
2. Gajić G., Stankovski S., Ostojić G., Tešić Z., Miladinović Lj.: Method of evaluating the impact of ERP implementation critical success factors – a case study in oil and gas industries (DOI:10.1080/17517575.2012.690105), Enterprise Information Systems, 2012, ISSN 1751-7575
3. Stankovski S., Ostojić G., Šenk I., Rakić-Skoković M., Trivunović S., Kučević D.: Dairy cow monitoring by RFID, Scientia Agricola, 2012, Vol. 69, No 1, pp. 75-80, ISSN 0103-9016
4. Janković J., Petrović N., Miladinović Lj., Popkonstantinović B., Stoimenov M., Petrović D., Ostojić G., Stankovski S.: Computer Simulation of Fast Hydraulic Actuators, Iranian Journal of Science and Technology - Transactions of Mechanical Engineering, Vol. 36, No. M1 , pp. 95-106, ISSN 2228-6187.
5. Stankovski S., Ostojić G., Tarjan L., Škrinjar D., Lazarević M.: IML Robot Grasping Process Improvement, Iranian Journal of Science and Technology - Transactions of Mechanical Engineering, Vol. 35, No. M1 , pp. 61-71, ISSN 2228-6187.
6. Popović B., Popović N., Mijić D., Stankovski S., Ostojić G.: Remote Control of Laboratory Equipment for Basic Electronics Courses: A LabVIEW-based Implementation DOI: 10.1002/cae.20531, Computer Applications in Engineering Education, 2011, ISSN 1061-3773
7. Vukelić Đ., Ostojić G., Stankovski S., Lazarević M., Tadić B., Hodolič J., Simeunović N.: Machining fixture assembly/disassembly in RFID environment, Assembly Automation, 2011, Vol. 31, No 1, pp. 62-68, ISSN 0144-5154
8. Ostojić, G., Stankovski, S.: Sistemi i uređaji za praćenje proizvoda tokom životnog ciklusa, Fakultet tehničkih nauka, 2012
9. Ostojić, G.,Stankovski, S., Tarjan, L., Šenk, I., Jovanović, V., DEVELOPMENT AND IMPLEMENTATION OF DIDACTIC SETS IN MECHATRONICS AND INDUSTRIAL ENGINEERING COURSES,International Journal of Engineering Education; 2010, Vol. 26, No. 1, pp. 2-8, ISSN 0949-149X

Representative refferences (minimum 5, not more than 10)

10.	Popkonstantinović B., Miladinović Lj., Stoimenov M., Petrović D., Ostojić G., Stankovski S.: DESIGN, MODELLING AND MOTION SIMULATION OF THE REMONTOIRE MECHANISM, Transactions of FAMENA, 2011, Vol. 35, No 2, pp. 79-93, ISSN 1333-1124.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	25			
Total of SCI(SSCI) list papers :	17			
Current projects :	Domestic :	3	International :	2

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Palčič -. Iztok
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2009	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2004	Faculty of Mechanical Engineering - Maribor	Production Systems, Organization and Management
Magister thesis	2002	Faculty of Mechanical Engineering - Maribor	Mechanical Engineering
Bachelor's thesis	1999	Faculty of Mechanical Engineering - Maribor	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IM1046	Structural and Development Projects	(I20) Engineering Management, Undergraduate Academic Studies
2.	IM1317	Project Procurement Management	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1821	Managing Media Projects	(I20) Engineering Management, Undergraduate Academic Studies
4.	HDOK4S	Selected chapters from automation of work processes	(I12) Industrial Engineering, Specialised Academic Studies
5.	IMDS59	Project approach in Effective Systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
6.	MBA413	Knowledge Systems and Project Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
7.	PLM05	Management of PLM Projects	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
8.	IM2101	Intelligent Enterprising and Effective Management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
9.	IM2107	SAP Enterprise systems	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
10.	IM2314	Program and Portfolio management	(I20) Engineering Management, Master Academic Studies
11.	HDOK-4	Selected Chapters in Production Process Automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
12.	HDOKL4	Selected chapters from automation of work processes	(H00) Mechatronics, Doctoral Academic Studies
13.	IMDR59	Project Approach in Effective Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. PALČIČ, Iztok, POLAJNAR, Andrej, VUJICA-HERZOG, Nataša. Upravljanje proizvodnje v večprojektnem okolju. Proj. mreža Slov., sep. 2002, letn. 5, št. 3, str. 20-28. [COBISS.SI-ID 7420182]
2. PANDŽA, Krsto, BUCHMEISTER, Borut, POLAJNAR, Andrej, PALČIČ, Iztok. Proizvodna strategija, podprta s teorijo proizvodnih virov : študij primera v podjetju Primat = An operations strategy supported with resource-based theory = a case study at the Primat company. Stroj. vestn., 2002, letn. 48, št. 7, str. 379-394. [COBISS.SI-ID 7601430] JCR IF: 0.05, SE (96/102), engineering, mechanical, x: 0.553
3. PALČIČ, Iztok, POLAJNAR, Andrej, PANDŽA, Krsto. Model za učinkovito upravljanje proizvodnje po naročilu = A model for the effective management of order-based production. Stroj. vestn., 2003, letn. 49, št. 7/8, str. 398-412. [COBISS.SI-ID 8491030] JCR IF: 0.048, SE (99/106), engineering, mechanical, x: 0.61
4. FULDER, Tatjana, PALČIČ, Iztok, POLAJNAR, Andrej, PIŽMOHT, Petja. Razvoj proizvodnih zmogljivosti v industrijskih grozdih - primer Slovenski avtomobilski grozd = The process of manufacturing capability development in industrial clusters - a Case study of the automotive cluster of Slovenia. Stroj. vestn., 2005, letn. 51, št. 12, str. 771-785. [COBISS.SI-ID 8782875] JCR IF: 0.116, SE (91/104), engineering, mechanical, x: 0.644

Representative refferences (minimum 5, not more than 10)

5.	PALČIČ, Iztok. Projektni management za večjo inovativnost v industrijskem grozdu. Proj. mreža Slov., sep. 2004, letn. 7, št. 3, str. 25-29. [COBISS.SI-ID 9007894]
6.	FULDER, Tatjana, PALČIČ, Iztok, POLAJNAR, Andrej. Razvoj proizvodnih sposobnosti in učinkovitost izvajanja projektov v industrijskih grozdih. Proj. mreža Slov., dec. 2005, letn. 8, št. 1/3, str. 13-20. [COBISS.SI-ID 10062614]
7.	PALČIČ, Iztok, LALIĆ, Bojan. Analytical hierarchy process as a tool for selecting and evaluating projects. Int. j. simul. model., Mar. 2009, vol. 8, no. 1, str. 16-26. http://dx.doi.org/10.2507/IJSIMM08(1)2.112, doi: 10.2507/IJSIMM08(1)2.112. [COBISS.SI-ID 13077782]
8.	PALČIČ, Iztok, BALAŽIC, Matej, MILFELNER, Matjaž, BUCHMEISTER, Borut. Potential of laser engineered net shaping (LENS) technology. Mater. manuf. process., 2009, vol. 24, no. 7/8, str. 750-753, doi: 10.1080/10426910902809776. [COBISS.SI-ID 13243670] JCR IF (2008): 0.706, SE (25/38), engineering, manufacturing, x: 0.905, SE (128/191), materials science, multidisciplinary, x: 1.953
9.	PALČIČ, Iztok, BUCHMEISTER, Borut, LALIĆ, Bojan. Analitični hierarhični proces kot orodje za ocenjevanje in izbiro projektov. Proj. mreža Slov., mar. 2009, letn. 12, št. 1, str. 4-10. [COBISS.SI-ID 13103126]
10.	PALČIČ, Iztok. Industrial clusters. Vienna: DAAAM International Publishing, 2007. VIII, 116 str., graf. prikazi. ISBN 3-901509-80-1. ISBN 978-3-901509-80-3. [COBISS.SI-ID 60180993] 2.02 Professional monograph

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	7			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Pantović B. Jovanka
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	13.06.1993
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2010		Mathematics
PhD thesis	2000	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	1996	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1991	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E145	Operations Research	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
2.	E213	Discrete Mathematics and Linear Algebra	(E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
3.	E221A	Mathematical Analysis 2	(E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
4.	GI101	Algebra	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
5.	H203	Mathematics 3	(H00) Mechatronics, Undergraduate Academic Studies
6.	IAM002	Discrete and Combinatorial Methods for Computer Graphics	(F10) Engineering Animation, Undergraduate Academic Studies
7.	S053N	Operations research	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
8.	0M512	Models of Computation	(OM1) Mathematics in Engineering, Master Academic Studies
9.	0ML512	Models of Computation	(OM1) Mathematics in Engineering, Master Academic Studies
10.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
11.	D0M08	Applied Abstract Algebra	(OM1) Mathematics in Engineering, Doctoral Academic Studies
12.	D0M13	Theory of Mobile Processes	(OM1) Mathematics in Engineering, Doctoral Academic Studies
13.	D0M14	Process Algebra	(OM1) Mathematics in Engineering, Doctoral Academic Studies
14.	D0M22	Multiple-Valued Logic	(OM1) Mathematics in Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
15.	D0M23	Clone Theory	(OM1) Mathematics in Engineering, Doctoral Academic Studies
16.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
17.	AID05	Theory of Mobile Processes	(F20) Engineering Animation, Doctoral Academic Studies
18.	AID06	Graph theory	(F20) Engineering Animation, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Gilezan S., Pantović J., Žunić J.: Partitioning Finite d-Dimensional Integer Grids with Applications, chapter in: Approximation Algorithms and Metaheuristics (editor: T. F. Gonzalez)., Chapman
2. Ghilezan S., Pantović J., Žunić J.,Separating points by parallel hyperplanes - characteization problem, IEEE Transactions on Neural Networks, 2007, Vol. 18, No. 5, 1356-1363.
3. Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Jovanka Pantovic, Daniele Varacca: Security types for dynamic web data. Theor. Comput. Sci, 2008, 402(2-3): 156-171
4. Pantović J., Vojvodić D., On the cardinality of nonfinitely based functionally complete algebras, Algebra Universalis, Vol. 43, No. 4, 2000, 369-374.
5. Pantović J., Tošić R., Vojvodić G., The cardinality of functionally complete algebras on a three element set, Algebra Universalis, Vol. 38, No.2, 1997, 136-140.
6. Pantović J., Machida H., Rosenberg I.: Regular sets of operations, Journal of Multiple Valued Logic and Soft Computing, 2012, Vol. 19, No 1-3, pp. 149-162, ISSN 1542-3980
7. Machida H., Pantović J.: Three classes of maximal hyperclones, Journal of Multiple Valued Logic and Soft Computing, 2012, Vol. 18, No 2, pp. 201-210, ISSN 1542-3980
8. Pantović J., Machida H.: Maximal hyperclones on E2 as hypercores , Journal of Multiple Valued Logic and Soft Computing, 2009, pp. 1-13, ISSN 1542-3980
9. Pantović J., Tošić R., Vojvodić G., Relative completeness with respect to two unary functions, Discrete Applied Mathematics, Vol.113 (2-3), 2001, 337-342.
10. Marinagiola Dezani-Ciancaglini, Silvia Ghilezan, Jovanka Pantović, Security types for dynamic web data, Proceedings of Trustworthy Global Computing, Lecture Notes in Computer Science, 2007, Vol. 4661, str. 263-280.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	30			
Total of SCI(SSCI) list papers :	13			
Current projects :	Domestic :	2	International :	3

Science, arts and professional qualifications

Name and last name:	Pečujlija D. Mladen
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.01.2007
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2011	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2010	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Magister thesis	2007	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1989	Faculty of Philosophy - Novi Sad	Psychological Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	URZP38	Selected Chapters in Psychology	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
2.	IM1052	Engineering Ethics	(I20) Engineering Management, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies
3.	IM1820	The theory and practice of organizational socialization	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1913	Research Methodology for Human Resources 1	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1920	Organizational socialization	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1922	Value management	(I20) Engineering Management, Undergraduate Academic Studies
7.	HR015	Ethical and legal aspects of human resources	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
8.	I077/S	Ethics in Education	(I20) Engineering Management, Specialised Professional Studies
9.	IMDS10	COGNITIVE MANAGEMENT	(I22) Engineering Management, Specialised Academic Studies
10.	IMDS99	Data ACQUISITION, ANALYSIS AND INTERPRETATION 2	(I22) Engineering Management, Specialised Academic Studies
11.	MM008	Audiovisual and media production	(I20) Engineering Management, Specialised Professional Studies
12.	ZP506	Crisis Management	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
13.	ZP515	Qualitative and quantitative methods of risk management	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
14.	IM2918	Human Resources Research Methodology 2	(I20) Engineering Management, Master Academic Studies
15.	IM2920	Personnel Management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
16.	IMDS77	Selected Chapters from Human Resource Management	(I22) Engineering Management, Specialised Academic Studies
17.	IMDS84	Data ACQUISITION, ANALYSIS AND INTERPRETATION 1	(I22) Engineering Management, Specialised Academic Studies
18.	IMDR10	COGNITIVE MANAGEMENT	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
19.	IMDR99	Data ACQUISITION, ANALYSIS AND INTERPRETATION 2	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
20.	IMDR77	Selected Chapters from Human Resource Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
21.	IMDR84	Data ACQUISITION, ANALYSIS AND INTERPRETATION 1	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Pecujlija, M., Cosic, D (2010). An Orthodox Christian Reflection: Genetic Enhencement Must Not Be the Creation Primacy Problem Between Man and God. American Journal of Bioethics, 4, 10, 78-80
2.	Pecujlija, M., Culibrk, D. (2012). Why we believe the computer when it lies. Computers in Human Behavior, 28, 143-152
3.	Pecujlija, M., Cosic, I., Ivanisevic, V. (2011). A Professor`s Moral Thinking at the Abstract Level vs The Professor`s Moral Thinking in the Real Life Situations. Science and Engineering Ethics, 17, 2, 299-320
4.	Pecujlija, M., Azemovic, N., Azemovic, R. (2011). Leadership and productivity in transition: employees' view in Serbia, Journal of East European Management Studies, 16, 3, 251-263
5.	Radlovacki, V., Beker, I., Majstorovic, V., Pecujlija, M., Stanivukovic, D., Kamberovic, B. (2011). Quality managers' estimates of quality management principles application in certified organisations in transitional conditions - is Serbia close to TQM? Journal of Mechanical Engineering, 57, 11, 851-861
6.	Jovanovic, R, Radlovacki, V, Pecujlija, M, Kamberovic, B, Delic, M, Grujic, J. (2012). Assessment of blood donors' satisfaction and measures to be taken to improve quality in transfusion service establishments. MEDICINSKI GLASNIK 9, 2, 231-238
7.	Pecujlija, M., Nerandzic, B., Perovic, V., Jevtic, A., Simic, N. (2010). Initating innovations in Serbian companies organizational cultures. African Journal of Business Management, 18, 4, 3957-3967
8.	Pecujlija, M. et al (2010). "Employees' Attitudes Toward Company Privatization as Possible Predictors of a High-Performance Work System", African Journal for Business and Management. 5, 5, 1663-1672
9.	Jokic, S, Cosic, I, Sajfert, Z, Pecujlija, M, Pardanjac, M. (2012) Schools as Learning Organizations: Empirical Study in Serbia. METALURGIA INTERNATIONAL, 17, 2, 83-89
10.	Radlovacki, V, Pecujlija, M, Kamberovic, B, Jovanovic, R, Delic, M, Beker, I. (2012). Satisfaction of high school students with the applicability of their knowledge TECHNICS TECHNOLOGIES EDUCATION MANAGEMENT-TTEM,7, 2, 777-785

Summary data for teacher's scientific or art and professional activity:

Quotation total :	7			
Total of SCI(SSCI) list papers :	11			
Current projects :	Domestic :	1	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Perović I. Veselin
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	24.10.2006
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2011		Production Systems, Organization and Management
PhD thesis	2006	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2004	Faculty of Technical Sciences - Novi Sad	Engineering Management
Education Specialist Thesis	2003	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1982	Faculty of Economics - Beograd	Economic Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	Z310	Social Ecology	(Z20) Environmental Engineering, Undergraduate Academic Studies
2.	A206	Sociology and Economy of the Built Enviroment	(A00) Architecture, Undergraduate Academic Studies
3.	ASO311	Sociology of Art and Culture	(AS0) Scenic Architecture, Technique and Design, Undergraduate Academic Studies
4.	ETI41	Sociology of Technique	(E02) Electronics and Telecommunications, Undergraduate Professional Studies
5.	IM1018	Management Accounting and Financial Management	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1414	Analyses of business reports	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1415	Indicators of Business Performance	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1417	Controlling	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1718	Controlling and Auditing in Insurance	(I20) Engineering Management, Undergraduate Academic Studies
10.	A005S	Urban sociology and economics: selected chapters	(A00) Architecture, Specialised Academic Studies
11.	GM502	Management in Construction	(G00) Civil Engineering, Master Academic Studies
12.	GM503	Management in a Construction Company	(G00) Civil Engineering, Master Academic Studies
13.	GM504	Selected Chapters in Construction Economy	(G00) Civil Engineering, Master Academic Studies
14.	IMDS89	Controlling and Internal Audit in Corporate Governance	(I22) Engineering Management, Specialised Academic Studies
15.	IMDS90	Selected Chapters of Strategic Management Accounting	(I22) Engineering Management, Specialised Academic Studies
16.	KIR002	Controlling	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
17.	KIR003	Financial Modeling	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
18.	KON01	Controlling Planning	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
19.	KON02	Controlling Data and Reporting	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
20.	MUO002	Management Accounting, Auditing and Controlling	(I20) Engineering Management, Specialised Professional Studies
21.	SZP003	Selected Chapters in Applied Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
22.	Z513A	Economics and the environmental protection	(Z20) Environmental Engineering, Master Academic Studies
23.	IM2319	Project evaluation	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
24.	IM2419	Business in Terms of Globalization	(I20) Engineering Management, Master Academic Studies
25.	IM2426	Operational Audit and Controlling	(M50) Energy Management, Master Academic Studies (OM1) Mathematics in Engineering, Master Academic Studies
26.	ZRMI3A	Sociological and Legal Aspects of Occupational Safety	(Z01) Safety at Work, Master Academic Studies
27.	A005	Urban Sociology and Economics – Selected Chapters	(A00) Architecture, Doctoral Academic Studies
28.	IMDR89	Controlling and Internal Audit in Corporate Governance.	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
29.	IMDR90	Selected Chapters of Strategic Managment Accounting	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Perović V., Nerandžić B., Bulatović B.: The Transition Processin the Context of Privatization in the Republic of Serbia (2001-2010) , Actual Problems of Economics, 2013, No 02-2013, ISSN 1993-6788
2. Perović V., Nerandžić B., Bojanić R., Živkov E., Bulatović B.: Inflence of Controlling the Investment Projection ERP (M) With Primary Focus on the Cash-flow in the Company, Metalurgia international, 2013, No 3 - 2013, ISSN 1582-2214
3. Nerandžić B., Perović V.: Personality and moral character traits and acnowledging the principles of management ethics,auditing and accounting ethics, African Journal of Business Management, 2011, ISSN 1993-8233
4. Perović V.: Controlling as a useuful managament instrument in crisis times, African Journal of Business Management, 2011, ISSN 1993-8233
5. Pečujlija M., Perović V., Nerandžić B.: Initiating innovation in Serbian companies organizational cultures, African Journal of Business Management, 2010, Vol. 4, No 18, pp. 3957-3967, ISSN 1993-8233
6. Perović V.: Controlling - a Chalange or necessity in time of crisis, 9. International Conference, Srečanje kontrolerjev: IZZivi in priložnosti kontrolinga, Ptuj, 24-25 Septembar, 2009
7. Demko-Rihter J., Perović V., Nerandžić B.: Harmonizacija finansijske i perspektive učenja i rasta u cilju povećanja vrednosti multidivizionalnog preduzeća, 15. Strategic Management and decision support systems in strategic Management, Subotica: Ekonomski fakultet Subotica, 22 April, 2010, ISBN 978-86-7233-252-0
8. Perović V., Nerandžić B., Bojanić R., Radišić S., Demko-Rihter J.: Controlling – as a Choice for Recent SME's, 3. International Conference for Entrepreneurship, Innovation and Regional Development ICEIRD, Novi Sad: Fakultet tehničkih nauka, 27-29 Maj, 2010, pp. 633-639
9. Nerandžić B., Perović V.: Internal audit, operational audit and corporative management, 4. Internacional Conference on Engineering Technologies - ICET, Novi Sad: Fakultet tehničkih nauka, 28-30 April, 2009, pp. 233-238, ISBN 978-86-7892-227-5, UDK: COBISS.SR-ID 245100807
10. Perović V., Nerandžić B., Todorović A., Bojanić R.: Controlling in a big company, 4. Internacional Conference on Engineering Technologies - ICET, Novi Sad: Fakultet tehničkih nauka, 28-30 April, 2009, pp. 239-242, ISBN 978-86-7892-227-5, UDK: COBISS.SR-ID 245100807

Summary data for teacher's scientific or art and professional activity:

Quotation total :	1			
Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Pilipović R. Stevan
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Sciences - Novi Sad
	01.01.1973
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	1987	Faculty of Sciences - Novi Sad	Mathematics
PhD thesis	1979	Faculty of Sciences - Novi Sad	Mathematics
Magister thesis	1977	Faculty of Mathematics - Beograd	Mathematics
Bachelor's thesis	1973	Faculty of Sciences - Novi Sad	Mathematics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	DAU004	Selected Chapters in Mathematics 2	(E20) Computing and Control Engineering, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies
2.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Atanacković TM, Oparnica L, Pilipović S: On a model of viscoelastic rod in unilateral contact with a rigid wall, IMA JOURNAL OF APPLIED MATHEMATICS, (2006) vol.71 br.1 str. 1-13.
2.	Atanackovic, TM Pilipovic, S Zorica, D: A diffusion wave equation with two fractional derivatives of different order, JOURNAL OF PHYSICS A-MATHEMATICAL AND THEORETICAL, (2007) vol.40 br.20 str. 5319-5333
3.	Pilipovic, S. Teofanov, N. : Multiresolution expansion, approximation order and quasiasymptotic behavior of tempered distributions, JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS, (2007) vol.331 br.1 str. 455-471
4.	Oberguggenberger, M. Pilipovic, S. Scarpalezos, D. : Positivity and positive definiteness in generalized function algebras, JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS, (2007) vol.328 br.2 str. 1321-1335
5.	Oberguggenberger, M. Pilipovic, S. Valmorin, V. : Global representatives of Colombeau holomorphic generalized functions, MONATSHEFTE FUR MATHEMATIK, (2007) vol.151 br.1 str. 67-74
6.	Pilipovic, S Scarpalezos, D : Divergent type quasilinear Dirichlet problem with singularities, ACTA APPLICANDAE MATHEMATICAE, (2006) vol.94 br.1 str. 67-82
7.	Pilipovic, Stevan Vuletic, Mirjana : Characterization of wave front sets by wavelet transforms, TOHOKU MATHEMATICAL JOURNAL, (2006) vol.58 br.3 str. 369-391
8.	Hormann, G Oberguggenberger, M Pilipovic, S : Microlocal hypoellipticity of linear partial differential operators with generalized functions as coefficients, TRANSACTIONS OF THE AMERICAN MATHEMATICAL SOCIETY, (2006) vol.358 br.8 str. 3363-3383
9.	Mitrovic, D Pilipovic, S : Approximations of linear Dirichlet problems with singularities, JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS, (2006) vol.313 br.1 str. 98-119
10.	Pilipovic, Stevan Scarpalezos, Dimitris Valmorin, Vincent : Equalities in algebras of generalized functions, FORUM MATHEMATICUM, (2006) vol.18 br.5 str. 789-801

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

Summary data for teacher's scientific or art and professional activity:				
Quotation total :	250			
Total of SCI(SSCI) list papers :	258			
Current projects :	Domestic :	0	International :	0

Science, arts and professional qualifications

Name and last name:	Popov B. Srđan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	05.09.2001
Scientific or art field:	Applied Computer Science and Informatics

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Applied Computer Science and Informatics
PhD thesis	2011	Faculty of Technical Sciences - Novi Sad	Electrical and Computer Engineering
Magister thesis	2007	Faculty of Technical Sciences - Novi Sad	Electrical and Computer Engineering
Bachelor's thesis	1999	Faculty of Technical Sciences - Novi Sad	Electrical and Computer Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E111	Programming Languages and Data Structures	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
2.	E214	Programming Languages and Data Structures	(E20) Computing and Control Engineering, Undergraduate Academic Studies (ES0) Power Software Engineering, Undergraduate Academic Studies
3.	URZP11	Fundamentals of Information Technologies	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
4.	URZP23	Applied Information Technologies	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
5.	URZP44	Application of geoinformation technology in risk management	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
6.	IMDS45	Application of information and satellite technology in risk management	(I22) Engineering Management, Specialised Academic Studies
7.	E2534	Data Compression	(E20) Computing and Control Engineering, Master Academic Studies (SE0) Software Engineering and Information Technologies, Master Academic Studies
8.	DRNI01	Selected Topics in Computer Programming	(E20) Computing and Control Engineering, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
9.	IMDR45	Application of Information and Satellite Technologies in Risk Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Jovčić N., Radonić (Jakšić) J., Turk Sekulić M., Vojinović-Miloradov M., Popov S.: Identification of emission sources of particle-bound polycyclic aromatic hydrocarbons in the vicinity of the industrial zone of the city of Novi Sad DOI: 10.2298/HEMIND120113062J, Hemijska industrija, 2012, ISSN 0367-598X
2.	Ćosić Đ., Popov S., Sakulski D., Pavlović A.: Geo-Information Technology for Disaster Risk Assessment, Acta Geotechnica Slovenica, 2011, Vol. 8, No 2011/1, pp. 64-74, ISSN 1854-0171
3.	Malbaški D., Kupusinac A., Popov S.: The Impact of Coding Style on the Readability of C Programs, TTEM. Tehnics tehnologies education management, 2011, Vol. 6, No 4, pp. 1073-1082, ISSN 1840-1503
4.	Sakulski D., Ćosić Đ., Popov S.: Implementation of Innovative Technologies for Disaster Risk Reduction, 1. International Conference Natural Hazards, Novi Sad: University of Novi Sad, Faculty of Science, 5 Maj, 2012, pp. 15-16, ISBN 978-86-7031-276-0
5.	Sakulski D., Ćosić Đ., Popov S., Pavlović A., Laban M.: Disaster risk management and fire safety, 1. International conference Protection, Ecology, Security, Bar: Fakultet za pomorstvo Kotor, 24-26 Maj, 2012, pp. 75-81
6.	Simić J., Popov S., Ćosić Đ., Sakulski D., Novaković T., Popović Lj., Pavlović A., Luhović A.: The aspect of bringing data in spatial relationship during the process of teaching at the subject “Disaster risk management” , UDK: 37.01:004 (082)
7.	Pavlović A., Ćosić Đ., Popov S., Kolaković S.: Indikatori praćenja hazardnih pojava poplave i suše u cilju poboljšanja planiranja melioracija, Tematski zbornik radova "Melioracije 07 - stanje i perspektive-", 2012, No 12, pp. 136-146, ISSN 978-86-7520-107-6, UDK: 626.8(082)

Representative refferences (minimum 5, not more than 10)

8.	Popović Lj., Popov S., Ćosić Đ., Sakulski D.: Impact of Visualization on Data Availability, UDK: CIP je dostupan u Univerzitetskoj biblioteci Rijeke pod brojem 121219001
9.	Alargić I., Badnjarević I., Vrtunski M., Popov S.: Setting the platform for testing the quality of DTM in the format of DTM-ASCII , 8. IEEE International Symposium on Intelligent Systems and Informatics (SISY), Subotica, , pp. 253-256, ISBN 978-1-4244-7395-3
10.	Popov S., Pavlović A., Ćosić Đ., Hlebjan M.: Interfacing Data Structures of Legacy Systems, 8. IEEE International Symposium on Intelligent Systems and Informatics (SISY), Subotica: 2010 IEEE , , pp. 409-411, ISBN 978-1-4244-7395-3

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	3			
Current projects :	Domestic :	2	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Radaković J. Nikola
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.11.1978
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2001	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Magister thesis	1989	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	1978	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	I914	Project Management	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies
2.	II1006	Processing Technology Products	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	II1008	Design methods of working procedures (CAPP, CAM)	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	II1019	Project Management	(I10) Industrial Engineering, Undergraduate Academic Studies
5.	IM1016	Production and Service Technologies	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1113	Improvement of products and processes	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1306	Project Management	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1315	Managing TQM projects	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1320	Project Risk Management	(I20) Engineering Management, Undergraduate Academic Studies
10.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
11.	IIDS10	Effective technological and production structures	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
12.	IIDS5	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies
13.	IM2116	Improvement of company flows	(I20) Engineering Management, Master Academic Studies
14.	IM2313	Planning, guidance and control of the project	(I20) Engineering Management, Master Academic Studies
15.	IMDS71	Selected topics of project management	(I22) Engineering Management, Specialised Academic Studies
16.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
17.	IMDR5	Selected chapters in enterprise's design, organization and control	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
18.	IMDR71	Selected topics of project management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
19.	IMDR85	Effective technological and production structures	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Morača, S., Hadžistević, M., Drstvenšek, I., Radaković, N.: "Application of Group Technology in Complex Cluster type Organizational Systems", Strojniski vestnik = Journal of Mechanical Engineering, University of Ljubljana, Faculty of Mechanical Engineering, Ljubljana, 2010., Vol. 56, No. 10, pp. 663-675, ISSN: 0039-2480

Representative refferences (minimum 5, not more than 10)

2.	Radišić, O., Radišić, M., Maksimović, R., Radaković, N.: "Industrial Cogeneration Appliance - An Example of Drilling Rig", Journal of Canadian Petroleum Technology, 2012, Vol. 51, No 6, pp. 487-492, ISSN 0021-9487
3.	Ćosić, I., Radaković, N., Simeunović, N.: "The Service Product Planning Work Plan Analysis", XIV International Scientific Conference on Industrial Systems - IS, Proceedings, str. 31-36, Fakultet tehničkih nauka - Departman za industrijsko inženjerstvo i menadžment, Novi Sad, 2008., UDK 658.5, ISBN 978-86-7892-135-3
4.	Morača, S., Radaković, N.: "The Group Approach Application In Complex Organizational Cluster Type Systems", XIV International Scientific Conference on Industrial Systems - IS, Proceedings, str. 427-431, Fakultet tehničkih nauka - Departman za industrijsko inženjerstvo i menadžment, Novi Sad, 2008., UDK 658.5, ISBN 978-86-7892-135-3
5.	Ćosić, I., Radaković, N., Simeunović, N., Lalić, B.: "Creating the Service Product by Applying the General Work Procedure Model", Annals of DAAAM for 2008 & Proceedings of the 19th International DAAAM Symposium, DAAAM International, Trnava, Slovakia, 2008., pp. 287-288, ISSN 1726-9679, ISBN: 978-3-901509-68-1, Published by DAAAM International Vienna, Vienna
6.	Radaković, N.: "Razvoj baze znanja za projektovanje tehnologije obrade", Edicija tehničke nauke - monografije br 23, Fakultet tehničkih nauka, Novi Sad, 2006, Recenzenti: Prof. dr Branko Ivković i Prof. dr Ilija Ćosić, UDK 658.5, ISBN 86-7892-025-4, str. 147
7.	Ćosić, I., Radaković, N., Lalić, B., Simeunović, N.: "The General Work Procedure Model for the Service Product", pp. 281-288, DAAAM International Scientific Book 2009, DAAAM International Vienna, 2009, ISSN 1726-9687, ISBN: 987-3-901509-71-1
8.	Vulanović, V., Stanivuković, D., Kamberović, B., Maksimović, R., Radaković, N., Radovački, V., Šilobad, M.: SISTEM KVALITETA ISO 9001:2000, Poglavlje 4: Sistem upravljanja kvalitetom, str. 51-74, Poglavlje 5: Odgovornost rukovodstva, str. 75-96, Poglavlje 7: Realizacija proizvoda, str. 127-208, Fakultet tehničkih nauka - Institut za industrijske sisteme i IIS - Istraživački i tehnološki centar, Novi Sad, 2007, ISBN 978-86-907041-3-2
9.	Radlovački, V., Kamberović, B., Radaković, N.: "Principi opšteg modela ocene efikasnosti i efektivnosti sistema menadžmenta kvalitetom podržane računarom", pregledni rad, Tehnika - Kvalitet, standardizacija i metrologija, Časopis saveza inženjera i tehničara Srbije, Beograd, ISSN 0040-2176, Godina 2008, Broj 6, str. 7-12
10.	Radišić, O., Radaković, N.: "Integration of Engineers in Project Management: An Example from Oil and Gas Industry", International Journal of Industrial Engineering and Management (IJIEM), Vol. 2 No 3, 2011, pp. 109-114, Fakultet tehničkih nauka, Departman za industrijsko inženjerstvo i menadžment, Novi Sad, ISSN 2217-2661

Summary data for teacher's scientific or art and professional activity:

Quotation total :	1			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	1	International :	1

Science, arts and professional qualifications

Name and last name:	Radenković B. Vladimir
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	30.03.2006
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2011		Production Systems, Organization and Management
PhD thesis	2005	Faculty of Technical Sciences - Novi Sad	Computer Science
Magister thesis	1996	Faculty of Technical Sciences - Novi Sad	Telecommunications and Signal Processing
Bachelor's thesis	1980	School of Electrical Engineering - Beograd	Telecommunications and Signal Processing

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IM1812	Basics of media and media technology	(I20) Engineering Management, Undergraduate Academic Studies
2.	IM1819	The use of media in the enterprise	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1822	Production of media content	(I20) Engineering Management, Undergraduate Academic Studies
4.	IMDS49	Media systems	(I22) Engineering Management, Specialised Academic Studies
5.	IMDS50	Media Research	(I22) Engineering Management, Specialised Academic Studies
6.	MM008	Audiovisual and media production	(I20) Engineering Management, Specialised Professional Studies
7.	MM014	Marketing and Public Relations	(I20) Engineering Management, Specialised Professional Studies
8.	SZP003	Selected Chapters in Applied Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
9.	IM2814	Processing of images in media	(I20) Engineering Management, Master Academic Studies
10.	IM2818	The organization of media production	(I20) Engineering Management, Master Academic Studies
11.	IMDS76	Selected topics in industrial marketing and media engineering	(I22) Engineering Management, Specialised Academic Studies
12.	IMDR49	Media Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
13.	IMDR50	Media Research	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
14.	IMDR76	Selected topics in industrial marketing and media engineering	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Radenković V., : Business practices in corporations of radio and television cable distribution programmes in Serbia, Journal for East European Management Studies (JEEMS), 2010, Vol.15, Issue 3, pp. 260-272, ISSN 0949-6181
2. Radenković, V., Radenković, M., Engus, K. (2010). Media and Social Responsible Business-A Serbian Model, African Journal of Business Management Vol.4 (15), November, 2010
3. PIP-A New Adaptive Filter for Noise Suppression in Still Images, V. Milošević, V. Crnojević, V. Radenković, V. Šenk, Facta Universitatis Series: Electronics and Energetics vol. 10, No. 1 (1997), 139-152.
4. V. Radenkovic, M. Temerinac, N. Teslic, M. Popovic: A Noise Reduction Algorithm Suitable for Hardware Implementation, JRE-Journal of Radio Electronics N10 Okt. 2003.
5. N. Teslić, V. Radenković, S. Crnogorac, Camera Object Tracking Fast Algorithm Advanced Concepts for Intelligent Vision Systems (ACIVS 2003) Ghent Belgium 2003., pp 188-193.
6. Camera Real-Time Human Tracking, N. Teslić,V. Radenković, D. Kukolj, M. Popović MIPRO 2004 Opatija, Croatia, pp.130-134
7. Dubravko Ćulibrk, Vladimir Radenković: Enhancing Video Object Segmentation Results Through Biologically Inspired Postprocessing, Daniel Socek TELSIKS 2007 Niš
8. Media Education – a Path for Acquiring Competences, Vladimir Radenković, Tehnologija, Informatika i Obrazovanje za društvo učenja i znanja, Peti međunarodni simpozijum TIO5, Novi Sad, 19.-20. jun 2009.

Representative refferences (minimum 5, not more than 10)	
9.	The Media Supporting an Integrated Economic Performance, Biljana Ratković Njegovan, Vladimir Radenković, International Scientific Conference on Industrial System-IS 08, Novi Sad: 02-03 Oktobar, 2008, str. 715- 722, UDK: 685.5(082), ISBN 978-86-7892-135-3.
10.	Radio i televizijska produkcija, Radenković Vladimir, Novi Sad, FTN-Izdavaštvo, 2008. 143str., UDK: 654.17/.19(075.8), ISBN 978-86-7892-139-1.

Summary data for teacher's scientific or art and professional activity:				
Quotation total :	0			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Radišić M. Mladen
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2008
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2012		Production Systems, Organization and Management
PhD thesis	2011	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	2008	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	-		Production Systems, Organization and Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IM1406	Investments Risk Management	(I20) Engineering Management, Undergraduate Academic Studies
2.	IM1412	Fundamentals of technology investments	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1420	Investments in innovation systems	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1421	Public sector management	(I20) Engineering Management, Undergraduate Academic Studies
5.	M3499	Energy markets	(M30) Energy and Process Engineering, Undergraduate Academic Studies
6.	I075/S	Selected chapters of portfolio management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
7.	IM001	Modern aspects of financial markets	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
8.	IM005	International financial transactions	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
9.	IMDS47	Behavioral Corporate Finance	(I22) Engineering Management, Specialised Academic Studies
10.	IMDS87	Financial engineering of public sector	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
11.	SZP003	Selected Chapters in Applied Management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
12.	IM007	Modern aspects of public sector systems	(I20) Engineering Management, Specialised Professional Studies
13.	IM2407	International business and finance	(I20) Engineering Management, Master Academic Studies
14.	IM2413	Enterprise portfolio management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
15.	IM2415	Investment Environment	(M50) Energy Management, Master Academic Studies (OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
16.	IM2416	Quantitative methods of risk management	(I20) Engineering Management, Master Academic Studies
17.	IM2422	Business case study solving	(I20) Engineering Management, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
18.	IM2423	Energy markets	(M50) Energy Management, Master Academic Studies
19.	IMDR87	Financial engineering of public sector	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Radišić M., Nedeljković A.: 5C Model - Business case study solving methodology, The New Educational Review, 2012, Vol. 27, No 1, pp. 19-30, ISSN 1732-6729
2.	Sando S., Radišić M., Dobromirov D.: Emerging markets - Galapagos for behavioral financial research (in print), Actual Problems of Economics, 2012, ISSN 1993-6788
3.	Dobromirov D., Radišić M., Kupusinac A.: Emerging markets arbitrages' perception: Risk versus growth potential, African Journal of Business Management, 2011, Vol. 5, No 3, pp. 713-721, ISSN 1993-8233
4.	Radišić O., Radišić M., Maksimović R., Radaković N.: Industrial Cogeneration Appliance - An Example of Drilling Rig, Journal of Canadian Petroleum Technology, 2012, Vol. 51, No 6, pp. 487-492, ISSN 0021-9487
5.	Marić B., Dobromirov D., Radišić M.: Researching the dependence between the dynamic indicators of investment profitability, African Journal of Business Management, 2011, Vol. 5, No 13, pp. 5076-5082, ISSN 1993-8233
6.	Radišić M., Marić B., Dobromirov D.: SMEs and entrepreneurs investments' profitability effects within the transition period in the Republic of Serbia, African Journal of Business Management, 2011, Vol. 5, No 7, pp. 2654-2659, ISSN 1993-8233
7.	Dobromirov D., Radišić M., Kupusinac A., Marić B.: Emerging Markets Unidirectional Sensitivity Coefficient as an Indicator in Portfolio Investors' Decision Making , International Journal of Industrial Engineering and Management - IJIEM, 2010, Vol. 1, No 2, pp. 63-68, ISSN 2217-2661
8.	Radišić M.: Uređivanje časopisa International Journal of Industrial Engineering and Management, International Journal of Industrial Engineering and Management - IJIEM, 2012, Vol. 3, No I - IV, ISSN 2217-2661
9.	Radišić M., Ferenčak M., Igor S., Stankovski S., Dobromirov D.: Harmonization of the Republic of Serbia tax system with the tax system of the European Union, 8. Augustin Cournot Doctoral Days, Strasbourg: University of Strasbourg, 13-15 April, 2011, pp. 15-15
10.	Dobromirov D., Radišić M., Šenk V.: Attractiveness of Serbia for venture capital, 3. International Conference for Entrepreneurship, Innovation and Regional Development ICEIRD, Novi Sad: University of Novi Sad, Faculty of Technical Sciences, IEM Department, 27-29 Maj, 2010, pp. 219-226, ISBN 978-86-7892-250-3

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	1	International :	2

Science, arts and professional qualifications

Name and last name:	Radlovački S. Vladan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.11.1992
Scientific or art field:	Quality, Effectiveness and Logistics

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
PhD thesis	2007	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	1999	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1992	Faculty of Technical Sciences - Novi Sad	Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II1014	Product measurement and control techniques	(I10) Industrial Engineering, Undergraduate Academic Studies
2.	II1036	Methods and techniques of quality improvement	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	IM1020	Quality Management System	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
4.	IM1037	Environmental Management System	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1606	Designing, Auditing and Analyses of Quality Management System	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
6.	IM1612	Methods and techniques of quality system improvements	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1613	Product measurement and control techniques	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1616	Quality planning	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1617	Quality Managament System in Service Provision	(I20) Engineering Management, Undergraduate Academic Studies
10.	IM1619	Quality and Procurement	(I20) Engineering Management, Undergraduate Academic Studies
11.	IM1622	Information Security Management System	(I20) Engineering Management, Undergraduate Academic Studies
12.	I503	Models of Excellence in Quality Management Systems	(I10) Industrial Engineering, Master Academic Studies
13.	I504	Integrated Management Systems	(I10) Industrial Engineering, Master Academic Studies
14.	IMDS95	Trends in Customer Relationship Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
15.	I309	Quality Management System	(LIM) Logistic Engineering and Management, Master Academic Studies
16.	LIM21	Total Quality Management and Logistics	(LIM) Logistic Engineering and Management, Master Academic Studies
17.	I912	Process approach and quality	(I10) Industrial Engineering, Master Academic Studies
18.	IIDS12	Quality and organizational performance	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
19.	IIDS30	Trends in the environmental management systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
20.	IIDS7	Selected topics in quality engineering and logistics	(I12) Industrial Engineering, Specialised Academic Studies
21.	IM2613	Models of Excellence in Quality Management Systems	(I20) Engineering Management, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
22.	IM2614	Integrated Management Systems	(I20) Engineering Management, Master Academic Studies
23.	IM2616	Product and service quality improvement - lean six sigma	(I20) Engineering Management, Master Academic Studies
24.	IM2617	Information Systems to Support Quality, Logistics and Maintenance	(I20) Engineering Management, Master Academic Studies
25.	IM2623	Total Quality Management	(I20) Engineering Management, Master Academic Studies
26.	IMDS74	Selected Topics in Quality Management and Logistics	(I22) Engineering Management, Specialised Academic Studies
27.	IMDS76	Selected topics in industrial marketing and media engineering	(I22) Engineering Management, Specialised Academic Studies
28.	IMDR94	Trends in the environmental management systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
29.	IMDR95	Trends in Customer Relationship Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
30.	IMDR74	Selected Topics in Quality Management and Logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
31.	IMDR76	Selected topics in industrial marketing and media engineering	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
32.	IMDR79	Selected topics in quality engineering and logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
33.	IMDR83	Quality abd organisational performance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
34.	ZRD212	Integrating occupational health and safety requirements into management systems	(Z01) Safety at Work, Doctoral Academic Studies
35.	ZRD213	Current state and development tendencies of quality management of work environment	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Radlovački V., Beker I., Majstorović V., Pečujlija M., Stanivuković D., Kamberović B.: Quality Managers' Estimates of Quality Management Principles Application in Certified Organisations in Transitional Conditions - Is Serbia Close to TQM, Strojniški vestnik - Journal of Mechanical Engineering, 2011, Vol. 57, No 11, pp. 851-861, ISSN 0039-2480
2. Delić M., Radlovački V., Kamberović B., Vulanović S., Hadžistević M., Tasić N.: ESTIMATES OF QUALITY MANAGEMENT SYSTEMS IN SERBIA , Metalurgia international, 2013, No 4, ISSN 1582-2214
3. Jovanović R., Radlovački V., Pečujlija M., Kamberović B., Delić M., Grujić J.: Assessment of blood donors' satisfaction and measures to be taken to improve quality in transfusion service establishments, Medicinski glasnik (BiH), 2012, Vol. 9, No 2, pp. 231-237
4. Radlovački V., Pečujlija M., Kamberović B., Jovanović R., Delić M., Beker I.: SATISFACTION OF HIGH SCHOOL STUDENTS WITH THE APPLICABILITY OF THEIR KNOWLEDGE, TTEM. Tehnics tehnologies education management, 2012, Vol. 7, No 2, pp. 777-785, ISSN 1840-1503
5. Radlovački V.: Opšti procesni model i ocenjivanje efikasnosti sistema menadžmenta kvalitetom u skladu sa zahtevima serije standarda ISO 9000, Novi Sad, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, FTN Izdavaštvo, 2011, ISBN 978-86-7892-346-3, UDK: 005.336.3 006.83
6. Kamberović B., Radlovački V.: RAZVOJ I STRUKTURA STANDARDA SISTEMA KVALITETA u knjizi: Dr Vojislav Vulanović, Dragutin Stanivuković, Bato Kamberović, R. Maksimović, Nikola Radaković, V. Radovački, M. Šilobad: SISTEM KVALITETA ISO 9001:2000, Novi Sad, Fakultet tehničkih nauka - Institut za industrijske sisteme i IIS-Istraživački i tehnološki centar, 2007, str. 7-38, ISBN 978-86-907041-3-2, UDK: 005.336.3 006.83
7. B. Kamberović, N. Radaković, V. Radlovački: ZNAČAJ UPRAVLJANJA DOKUMENTACIJOM SISTEMA KVALITETA ZA UNAPREĐENJE PROCESA RADA, Rad saopšten na VII međunarodnoj konferenciji "Fleksibilne tehnologije", Zbornik radova konferencije, str. 87-88, Novi Sad, jun 2000.
8. V. Radlovački, B. Kamberović, M. Brkić: SISTEM ZA UPRAVLJANJE ZAPISIMA KAO POGODNA OSNOVA ZA PROJEKTOVANJE INFORMACIONOG SISTEMA, 4. međunarodni kongres Kvalitet - Most ka Evropi, Beograd, 29 - 31. maj 2002., rad objavljen u zborniku radova u elektronskoj formi (CD), objavljen u časopisu Menadžment totalnim kvalitetom, YUSQ, Beograd, No 3-4, Vol 30, str. 145-150, UDK 658.5, YU ISSN 0354-9771
9. Štrbac B., Hadžistević M., Vrba I., Radlovački V., Hodolič J.: Analysis of Influencing Factors on Stylus Calibration of CMM, 22. DAAAM International Symposium, Vienna: DAAAM International Viena, 23-26 Novembar, 2011, pp. 1665-1666, ISBN 978-3-901509-83-4, UDK: 1726-9679
10. Marić B., Kamberović B., Radlovački V., Delić M., Zubanov V.: Observing the dependence between dynamic indicators of investment profitability - Relative net present value and internal rate of return, African Journal of Business Management, 2011, Vol. 5, No 26, pp. 331-337, ISSN 1993-8233

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	6			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Rajković R. Milan
Academic title:	Senior Science Associate
Name of the institution where the teacher works full time and starting date:	Vinča Institute of Nuclear Sciences - Vinča
	01.01.2000
Scientific or art field:	Physical Science

Academic carieer	Year	Institution	Field
Academic title election:	2005	Vinča Institute of Nuclear Sciences - Vinča	Physical Science
PhD thesis	1997	University of Belgrade - Beograd	Physics
Magister thesis	1983	University of Pennsylvania - Tennessee	Physics
Bachelor's thesis	1982	University of Pennsylvania - Tennessee	Physics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	D. Horak, S. Maletić, M. Rajković, Persistent Homology of Complex Networks, Journal of Statistical Mechanics and Applications (2009) P03034.
2.	Milan Rajković, M.M. Škorić, K. Sølna and G. Antar, Characetrization of Local Turbulence in Magnetic Confinement Devices, Nuclear Fusion 48 (2008) 1-13.
3.	Mladen Nikolić and Milan Rajković, A group theoretic approach to a class of third-order differential equations with two parameter symmetry group solvable by quadratures, Nonlinear Dynamics 48 (2007) 17-27.
4.	Mladen Nikolić and Milan Rajković, Bifurcations in Nonlinear Models of Fluid Conveying Pipes, Journal of Fluids and Structures, 22 (2006),
5.	Z. Mihailović and M. Rajković, Cooperative Parrondo's games on a two-dimensional lattice, Physica A 365 (2006) 244-251
6.	Milan Rajković, Tomo-hiko Watanabe and M.M. Škorić, Level crossing function in the Analysis of Confined Plasma Turbulence, Nuclear Fusion 49 (2009) 095016i
7.	Milan Rajković and M.M. Škorić, Characterization of Intermittency in Plasma Edge Turbulence; Contributions to Plasma Physics 48 (2008) L31-L35.
8.	M. Rajković, Nonextensive entropy as a measure of time series complexity, Physica A 340 (2004) 327-333
9.	M. Rajković and Z. Mihailović, Quantifying Complexity in the Minority Game, Physica A 325 (2003) 40 - 47
10.	Z. Mihailović and M. Rajković, One-dimensional Asynchronous Cooperative Parrondo's Games, Fluctuation and Noise Letters 3 (2003) L389 - 398

Summary data for teacher's scientific or art and professional activity:

Quotation total :	100			
Total of SCI(SSCI) list papers :	22			
Current projects :	Domestic :	1	International :	1

Science, arts and professional qualifications

Name and last name:	Ralević M. Nebojša
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.1990
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2010	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	1997	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	1994	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1990	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H103	Mathematics 1	(H00) Mechatronics, Undergraduate Academic Studies
2.	H107	Mathematics 2	(H00) Mechatronics, Undergraduate Academic Studies
3.	M4201	Mathematics 3	(M30) Energy and Process Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
4.	M4202	Applied Mathematical Analysis	(M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
5.	P216	Numerical Analysis	(P00) Production Engineering, Undergraduate Academic Studies
6.	0M502	Partial Differential Equations	(OM1) Mathematics in Engineering, Master Academic Studies
7.	0M508	Mathematical Foundations of Fuzzy Systems	(OM1) Mathematics in Engineering, Master Academic Studies
8.	0M517	Numerical Analysis	(OM1) Mathematics in Engineering, Master Academic Studies
9.	0ML502	Partial Differential Equations	(OM1) Mathematics in Engineering, Master Academic Studies
10.	0ML508	Mathematical Foundations of Fuzzy Systems	(OM1) Mathematics in Engineering, Master Academic Studies
11.	0ML517	Numerical Analysis	(OM1) Mathematics in Engineering, Master Academic Studies
12.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
13.	Z506	20BAdvanced Course in Mathematics 1	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies (Z20) Environmental Engineering, Master Academic Studies
14.	Z506	Viši kurs matematike 1(uneti naziv na engleskom)	(Z20) Environmental Engineering, Master Academic Studies
15.	D0M02	Partial Differential Equations	(OM1) Mathematics in Engineering, Doctoral Academic Studies
16.	D0M07	Mathematical Foundations of Fuzzy Systems	(OM1) Mathematics in Engineering, Doctoral Academic Studies
17.	D0M21	Fuzzy Systems and Their Applications	(OM1) Mathematics in Engineering, Doctoral Academic Studies
18.	D0M38	Non-linear Equations and Their Applications	(OM1) Mathematics in Engineering, Doctoral Academic Studies
19.	D0M39	Optimization Methods and Mathematical Modelling	(OM1) Mathematics in Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
20.	DOM54	Computational geometry	(F20) Engineering Animation, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
21.	DOM55	Pattern Recognition	(F20) Engineering Animation, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
22.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	E. Pap, N. Ralević, Pseudo-Laplace transform, Nonlinear Analysis: Theory Methods and Applications, 33 (1998), 533-550.
2.	N. M. Ralević, Lj. M. Nedović, T. Grbić, The pseudo-linear superposition principle for nonlinear partial differential equations and representation of their solution by the pseudo-integral, Fuzzy Sets and Systems 155 (2005) 89-101.
3.	Lj. M. Nedović, N. M. Ralević, T. Grbić,Large deviation principle with generated pseudo measures,Fuzzy Sets and Systems 155 (2005) 65-76.
4.	T. Lukić, N. M. Ralević, Geometric Mean Newton"s Method for Simple and Multiple Roots, Applied Mathematics Letters (accepted).
5.	N. M. Ralević,One characterization of Navier-Stokes equation, Acta Mechanica Slovaca, Košice, ročnik 8., č. 4/2004, str. 97-102.
6.	N. Ralević, Some new properties of g-calculus, Univ. u Novom Sadu Zb. Rad. Prirod.-Mat. Fak. Ser. Mat. 24, 1 (1994), 139-157.
7.	E. Pap, N. Ralević, Pseudo operations on finite intervals, Novi Sad J. Math. Vol. 29, No. 1, 1999, 1-6
8.	N. M. Ralević, A generalization of the Pseudo-Laplace transform, Novi Sad J. Math. Vol. (accepted).
9.	I. Kovačević, N. Ralević, Funkcionalna analiza, Edicija tehničke nauke, Novi Sad (2004), 203 str.
10.	I. Kovačević, N. Ralević, Matematička analiza I (uvodni pojmovi i granični procesi), Novi Sad (2000), 155 str.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	28			
Total of SCI(SSCI) list papers :	10			
Current projects :	Domestic :	2	International :	0

Science, arts and professional qualifications

Name and last name:	Ratković-NJegovan M. Biljana
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Media Engineering and Management

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Media Engineering and Management
PhD thesis	2003	University of Novi Sad - Novi Sad	Social Science
Magister thesis	1985	Essex university - Nepoznato	Social Science
Bachelor's thesis	1980	Faculty of Political Sciences - Beograd	Political Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	I409	Psychology in Management	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
2.	IM1820	The theory and practice of organizational socialization	(I20) Engineering Management, Undergraduate Academic Studies
3.	IM1920	Organizational socialization	(I20) Engineering Management, Undergraduate Academic Studies
4.	HR015	Ethical and legal aspects of human resources	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
5.	I077/S	Ethics in Education	(I20) Engineering Management, Specialised Professional Studies
6.	MM004	Theory and Practice of Media Communication	(I20) Engineering Management, Specialised Professional Studies
7.	URZP64	The role of media in reducing the risk	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
8.	IM2218	Entrepreneurship in creative industries	(I20) Engineering Management, Master Academic Studies
9.	IM2822	Mass Communications Research	(I20) Engineering Management, Master Academic Studies
10.	IMDS76	Selected topics in industrial marketing and media engineering	(I22) Engineering Management, Specialised Academic Studies
11.	MM016	MEDIA ORGANISATION AND MANAGEMENT	(I20) Engineering Management, Specialised Professional Studies
12.	IMDR76	Selected topics in industrial marketing and media engineering	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Ratković Njegovan, B. Teorija političke javnosti. (2004). Sremski Karlovci: Kairos.
2. Ratković Njegovan, B.. Merenje RTV auditorijuma i vrednovanje programa. (2005), Link, br. 32, Link – dodatak.
3. Ratković Njegovan, B. Mediji i auditorijum. (2007). Link, br. 58, god. VI, pp. 23–26.
4. Ratkov-NJegovan B.: Evropska javna sfera i mediji. (2008). Link, br. 65, god. VII, Link – dodatak.
5. Grubić-Nešić, L., Vranješ, S., Ratković Njegovan, B., Mitrović, S. (2012). Atitudes of the employees about the organizational restructuring: a sample of organizations in Serbia. Metalurgia international 12(17). ISSN: 1582-2214
6. Ratković Njegovan, B., Crnomarković, M.. (2012). School management in Serbia: Key Aspects of its Relation to School Success. Journal for East European Management Studies, 17(29, 184–205.
7. Ratković Njegovan, B., Vukadinović, M., Grubić Nešić, L. (2011). Characteristics and Types of Authority: the Attitudes of Young People. A Case Study. Sociológia / Slovak Sociological Review, 43, 657-673. ISSN: 0049-1225.
8. Ratković Njegovan, B., Radenković. V. (2010). Kablovski distribucioni sistemi u Srbiji: Izlazak iz sive zone poslovanja. Zbornik Matice srpske za društvene nauke, 131, 97–110. ISSN: 0352-5732/UDK 3(05).
9. Ratković Njegovan B., Šiđanin. I. (2011). Media and Creative Industries: The value of Creative Content In: XV International Scientific Conference on Industrial Systems – IS 11). Novi Sad: Faculty of Technical Sciences, Department of Industrial Engineering and Management, 583-587. ISBN: 978-86-7892-341-8.
10. Ratković Njegovan, B., Đurašković, D., Kostić, B. (2011). Creative Portfolio Strategy as a Model of Management in Media Company: An Example of Public Broadcasting. Journal of Engineering Management and Competitiveness (JEMC), 2(1), 6-10.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	4			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Ristić M. Sonja
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2006
Scientific or art field:	Information-Communication Systems

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
PhD thesis	2003	Faculty of Economics - Subotica	Information-Communication Systems
Magister thesis	1994	Faculty of Economics - Subotica	Information-Communication Systems
Bachelor's thesis	1989	Faculty of Economics - Subotica	Economics
Bachelor's thesis	1983	Faculty of Sciences - Novi Sad	Mathematics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	Z201	Fundamentals of Computer Technologies	(Z20) Environmental Engineering, Undergraduate Academic Studies
2.	Z201A	Fundamentals of Computer Technologies	(Z01) Safety at Work, Undergraduate Academic Studies
3.	ISIT3A	Metodologije i sistemi za upravljanje IT resursima	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
4.	H401	Object Oriented Technologies	(H00) Mechatronics, Undergraduate Academic Studies
5.	II1002	Computer Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
6.	IM1010	Fundamentals of Information Technologies	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1506	Database Design	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
8.	IM1512	Object-oriented Infromation Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
9.	IM1516	Database Systems	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
10.	IM1519	Information System Architecture and Computer Networks	(I20) Engineering Management, Undergraduate Academic Studies
11.	SE0016	Databases	(SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
12.	IMDS33	Structures of Modern Information and Communication Systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
13.	IMDS36	Advanced data models and database systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
14.	PLM11	Product Data Management	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
15.	LIM02	Business Information Systems	(LIM) Logistic Engineering and Management, Master Academic Studies
16.	E2537	IT Resources Management	(SE0) Software Engineering and Information Technologies, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
17.	IIDS8	Selected chapters from Information, management and communication systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies
18.	IM2513	Data Warehouse Design	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
19.	IMDS73	Selected chapters from Information management	(I22) Engineering Management, Specialised Academic Studies
20.	PLM04	Product Data Management	(I20) Engineering Management, Specialised Professional Studies
21.	IMDR33	Structures of Modern Information and Communication Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
22.	IMDR36	Advanced Data Models and Database Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
23.	IMDR73	Selected chapters from Information management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	IMDR81	Selected chapters from Information, management and communication systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Luković I., Popović A., Mostić J., Ristić S.: A Tool for Modeling Form Type Check Constraints and Complex Functionalities of Business Applications, Computer Science and Information Systems (ComSIS), 2010, Vol. 7, No 2, pp. 359-385, ISSN 1820-0214
2.	Lukovic I, Mogin P, Pavicevic J, Ristic S, An Approach to Developing Complex Database Schemas Using Form Types, Software: Practice and Experience, Volume 37, Issue 15, Pages 1621-1656, December 2007. Online ISSN: 1097-024X Print ISSN: 0038-0644 Copyright 2007 John Wiley & Sons, Ltd. Hoboken, USA, Published Online: May 29 2007 12:28PM DOI: 10.1002/spe.820
3.	Aleksić S., Ristić S., Luković I., Čeliković M.: A Design Specification and a Server Implementation of the Inverse Referential Integrity Constraints, Computer Science and Information Systems (ComSIS), 2013, Vol. 10, ISSN 1820-0214 (Accepted for publishing)
4.	Ristić S., Luković I., Pavićević J., Mogin P.: Resolving Database Constraint Collisions Using IIS*Case Tool, Journal of Information and Organizational Sciences (JIOS), 2007, Vol. 31, No 1, pp. 187-206, ISSN 1846-3312, UDK: 004.651
5.	Luković I., Ristić S., Mogin P., Pavićević J.: Database Schema Integration Process – A Methodology and Aspects of Its Applying, Novi Sad Journal of Mathematics, 2006, Vol. 36, No 1, pp. 115-150, ISSN 1450-5444
6.	Luković I., Mogin P., Govedarica M., Ristić S.: The Structure of A Subschema and Its XML Specification, Journal of Information and Organizational Sciences (JIOS), 2002, Vol. 26, No 1-2, pp. 69-85, ISSN 1846-3312
7.	Ristić S., Aleksić S., Luković I., Banović J.: Form-Driven Application Development, Acta Electrotechnica et Informatica, Faculty of Electrical Engineering and Informatics, Technical University Kosice, 2012, Vol. 12, No 1, pp. 9-16
8.	Ristić S.: Lean Thinking Principles in the Context of Model-Driven Software Development, 1. International Scientific Conference on Lean Technologies - LeanTech, Novi Sad: Faculty of Technical Sciences, 13-14 Septembar, 2012, pp. 233-239, ISBN 978-96-7892-445-3
9.	Ristić S., Luković I., Aleksić S., Banović J., Al-Dahoud A.: An Approach to the Specification of User Interface Templates for Business Applications, 5. Balkan Conference in Informatics, Novi Sad: ACM New York, USA, 16-20 Septembar, 2012, pp. 124-129, ISBN 978-1-4503-1240-0
10.	Ristić S., Rakić-Skoković M., Al-Dahoud A.: An Overview of the Approaches for A PLM Application's Customization, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Faculty of Technical Sciences; Department of Industrial Engineering and Management; University of Novi Sad, 14-16 Septembar, 2011, pp. 217-222, ISBN 978-86-7892-341-8

Summary data for teacher's scientific or art and professional activity:

Quotation total :	14			
Total of SCI(SSCI) list papers :	3			
Current projects :	Domestic :	2	International :	2

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Sakulski M. Dušan
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2007
Scientific or art field:	Environment Protection Engineering

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Environment Protection Engineering
PhD thesis	2002	WITS University - Johannesburg	Environment Protection Engineering
Bachelor's thesis	1982	Faculty of Civil Engineering - Beograd	Civil Engineering
Magister thesis	-		Civil Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	URZP23	Applied Information Technologies	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
2.	URZP36	Risks in Manipulating Hazardous Substances	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
3.	URZP41	Disasters and Vulnerability	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
4.	URZP44	Application of geoinformation technology in risk management	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
5.	URZP46	Cycle Elements of Catastrophic Events	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
6.	URZP56	Fundamentals of Risk and Fire Protection Management	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
7.	Z415	Accidental Risks Management	(Z20) Environmental Engineering, Undergraduate Academic Studies
8.	Z511P	Institutional Framework in Risk Management	(ZP0) Disaster Risk Management and Fire Safety, Undergraduate Academic Studies
9.	Z307	Modelovanje i simulacija u IZŽS(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
10.	Z409A	Upravljanje opasnim otpadom(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
11.	Z415	Upravljanje akcidentalnim rizicima(uneti naziv na engleskom)	(Z20) Environmental Engineering, Undergraduate Academic Studies
12.	ZC047	Waste to energy tehnologies	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
13.	ZP515	Qualitative and quantitative methods of risk management	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
14.	Z510	Upravljanje akcidentalnim rizicima i životna sredina(uneti naziv na engleskom)	(Z20) Environmental Engineering, Master Academic Studies
15.	Z511	Institucionalni okviri upravljanja akcidentnim rizicima(uneti naziv na engleskom)	(Z20) Environmental Engineering, Master Academic Studies
16.	ZP501	Integrated Natural Disaster Risk Management	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
17.	IM2707	Methods for the analysis of insurance risk	(I20) Engineering Management, Master Academic Studies
18.	IM2714	Disaster risk management cycle	(I20) Engineering Management, Master Academic Studies
19.	IM2715	Modeling and simulation in risk management	(OM1) Mathematics in Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
20.	IMDS72	Advanced risk assessment methods	(I22) Engineering Management, Specialised Academic Studies
21.	MPK009	Enviromental hazards	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies
22.	MPK012	Solid waste management	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies
23.	MPK014	Monitoring and system control	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
24.	MPK019	Disaster risk management	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies
25.	ZCM06	Security of strategic energy facilities	(ZC0) Clean Energy Technologies, Master Academic Studies
26.	IMDR72	Advanced risk assessment methods	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
27.	ZRD233	Selected topics in the field of insurance from the standpoint of safety and health at work	(Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Marjanovic P., Miloradov M., Cukic Z., Sakulski D., Bogdanovic S.: "Integrated cadastre (Inventory System) for pollution sources in the Danube Basin in Yugoslavia", Water Science and Technology, Vol. 32 No 5-6 pp 265-275, IWA Publishing 1995
2.	Sakulski D.: "Web-enabled GIS in Disaster Management", The Global Magazine for Geomatics, May 2005, Volume 19, Number 5
3.	Sakulski D.: "Implementation of the multi-software solution for the on-the-fly calculation of the Standardized Precipitation Index (SPI) as a drought indicator for South African environment" ENVIROSOFT 2000, 2000, Bilbao, Spain
4.	Sakulski D., "Development and implementation of a database driven web-enabled integrated system for air quality observation and analysis", International Conference on Air Pollution, 2001, Ancona, Italy
5.	Sakulski D. Stephenson D, Marjanovic P.: "WebMathematica as a Core Service for the Calculation of the Drought Indicator for South Africa", The 5th International Mathematica Symposium, 2003, London, UK
6.	Sakulski D.: "South African National Disaster Hazard and Vulnerability ATLAS", International Conference on Disasters and Society – From Hazard Assessment to Risk Reduction, 2004, Karlsruhe, Germany
7.	Sakulski D.: "Geo-Information as an Integral Component of the National Disaster Hazard and Vulnerability ATLAS", First International Symposium on Geo-Information for Disaster Management, 2005, Delft, Netherlands
8.	Sakulski D.: "Analiza zaustavnog puta u funkciji merodavnog vozila", Put i saobraćaj, 1984
9.	Sakulski D.: "Ojačanje kolovoza upotrebom FW deflektometra", Put i saobraćaj, 1986
10.	Sakulski D., Katic Z.: "Klasifikacija oštećenja kolovoza", Put i saobraćaj, 1986

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	1			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Satarić V. Miljko
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	03.01.1973
Scientific or art field:	Physics

Academic carieer	Year	Institution	Field
Academic title election:	1995	Faculty of Technical Sciences - Novi Sad	Physics
PhD thesis	1984	School of Electrical Engineering - Beograd	Physics
Magister thesis	1979	School of Electrical Engineering - Beograd	Physics
Bachelor's thesis	1972	Faculty of Sciences - Novi Sad	Physics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E103	Physics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
2.	E215	Physics	(E20) Computing and Control Engineering, Undergraduate Academic Studies
3.	Z103	Selected Chapters in Physics 1	(Z01) Safety at Work, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
4.	Z110	Selected Chapters in Physics 2	(Z01) Safety at Work, Undergraduate Academic Studies (Z20) Environmental Engineering, Undergraduate Academic Studies
5.	EI410	Biophysics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
6.	DE203S	Odabrana poglavlja iz kvantne elektronike	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
7.	DE301S	Molekularna elektronika(uneti naziv na engleskom)	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies
8.	DZ01FS	Selected Chapters in Physics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
9.	EM511	Quantum and Organic Electronics	(E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
10.	SI028	Biophysics	(E00) Power, Electronic and Telecommunication Engineering, Specialised Professional Studies
11.	DE203	Selected Chapters in Quantum Electronics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies
12.	DE301	Molecular Electronics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
13.	DZ01F	Selected Chapters in Physics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	S. Zdravković, M.V. Satarić, “Single-Molecule Unzipping Experiments on DNA Peyrard-Bishop-Dauxois Model”,Phys.Rev.E73,021905-11,2006.
2.	J. A. Tuszynski, J. A. Brown, E. Crawford, E. J. Carpenter, M. L. A. Nip, J. M. Dixon, M. Satarić, “Molecular dynamics simulations of tubulin structure and calculations of electrostatic properties of microtubules”, Mathematical and Computer Modelling, vol. 41, no.10, pp. 1055-1070, 2005.
3.	M. Satarić, B. Satarić, J. A. Tuszynski, “Nonlinear model of microtubule dynamics”, Electromagnetic Biology and Medicine, vol.24, no. 3, pp. 255-264, 2005.
4.	S. Zdravković J. A. Tuszynski, M. Satarić “Peyrard-Bishop-Dauxois model of DNA dynamics and impact of viscosity”, Journal of Computational and Theoretical Nanoscience, vol. 2, no. 2, pp. 263-271, 2005.
5.	S. Zdravković, M. Satarić, “Optical and Acoustical Frequencies in a Nonlinear Helicoidal Model of DNA Molecule”, Chinese Physics Letters 22, pp. 850-853, 2005.
6.	S. Portet, J. A. Tuszynski, J. M. Dixon, M. Satarić, “Models of spatial and orientational self-organization of microtubules under the influence of gravitational fields”, Physical Review E, vol. 68, no. 2, 2003.
7.	M. Satarić, J. A. Tuszynski, “Relationship between the nonlinear ferroelectric and liquid crystal models for microtubules”, Physical Review E, vol. 67, no. 1, 2003.
8.	S. Zdravković, M. Satarić, “DNA dynamics and big viscosity”, International Journal of Modern Physics B, vol.17, no. 31-32, pp. 5911-5923, 2003.
9.	M. Satarić, J. A. Tuszynski, “Impact of regulatory proteins on the nonlinear dynamics of DNA”, Physical Review E, vol. 65, no. 5, 2002.
10.	G. Keković, D. Raković, M. Satarić, D. Koruga, “A kink-soliton model of charge transport through microtabular cytoskeleton”, Current Research in Advanced Materials and Processes, vol. 494, pp. 507-512, 2005.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	295			
Total of SCI(SSCI) list papers :	67			
Current projects :	Domestic :	1	International :	2

Science, arts and professional qualifications

Name and last name:	Sladoje Matić I. Nataša
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	14.03.1994
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2011		Mathematics
PhD thesis	2005	University of Novi Sad - Novi Sad	Mathematical Sciences
Magister thesis	1998	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1992	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	A101	Mathematics	(A00) Architecture, Undergraduate Academic Studies
2.	E135B	Mathematical Analysis 2	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
3.	GI107	Mathematical Analysis 1	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
4.	IAM001	Mathematical Shape Modeling for Computer Animation	(F10) Engineering Animation, Undergraduate Academic Studies
5.	IAM004	Geometry of Discrete Space	(F10) Engineering Animation, Undergraduate Academic Studies
6.	IGA008	Mathematics for Engineering Graphics	(F10) Engineering Animation, Undergraduate Academic Studies
7.	BMI91	Mathematics 1	(BM0) Biomedical Engineering, Undergraduate Academic Studies
8.	BMI92	Mathematics 2	(BM0) Biomedical Engineering, Undergraduate Academic Studies
9.	E101A	Discrete Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
10.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
11.	Z506	20BAdvanced Course in Mathematics 1	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies (Z20) Environmental Engineering, Master Academic Studies
12.	IA018	Computer Geometry	(F20) Engineering Animation, Master Academic Studies
13.	D0M28	Digital Geometry	(OM1) Mathematics in Engineering, Doctoral Academic Studies
14.	D0M29	Image Processing 1	(OM1) Mathematics in Engineering, Doctoral Academic Studies
15.	D0M30	Image Processing 2	(OM1) Mathematics in Engineering, Doctoral Academic Studies
16.	D0M31	Applied Algorithms	(OM1) Mathematics in Engineering, Doctoral Academic Studies
17.	D0M32	Combinatorial and Geometric Algorithms	(OM1) Mathematics in Engineering, Doctoral Academic Studies
18.	D0M33	Positional Games	(OM1) Mathematics in Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
19.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
20.	AID07	Digital geometry	(F20) Engineering Animation, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Sladoje N., Lindblad J., Nystrom I.: Defuzzification of spatial fuzzy sets by feature distance minimization. , Image and Vision Computing, 2011, Vol. 29, No 2-3, pp. 127-141, ISSN 0262-8856
2.	Lukić T., Lindblad J., Sladoje N.: Regularized Image Denoising Based on Spectral Gradient Optimization, Inverse Problems, 2011, Vol. 27, No 8, pp. 8501-1, ISSN 0266-5611
3.	Sladoje N., Lindblad J.: High precision boundary length estimation by utilizing grey-level information , IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009, Vol. 31, No 2, pp. 357-363, ISSN 0162-8828
4.	N. Sladoje and J. Lindblad, "Representation and Reconstruction of Fuzzy Disks by Moments", Fuzzy Sets and Systems, Vol. 158, No. 5, pp. 517-534, 2007.<\eng>
5.	N. Sladoje, I. Nyström, and P.K. Saha, "Measurements of digitized objects with fuzzy borders in 2D and 3D", Image and Vision Computing, vol. 23, pp 123-132, 2005.<\eng>
6.	J. Zunic and N. Sladoje, "Efficiency of Characterizing Ellipses and Ellipsoids by Discrete Moments", IEEE Trans. Pattern Analysis and Machine Intelligence, vol.22, No.4, pp 407-414, 2000.<\eng>
7.	J. Chanussot, I. Nyström and N. Sladoje, "Shape signatures of fuzzy star-shaped sets based on distance from the centroid", Pattern Recognition Letters, vol. 26(6), pp. 735-746, 2005.<\eng>
8.	Ćurić,V., Lindblad, J., Sladoje, N., Sarve, H., Borgefors, B. A new set distance and its application to shape registration. Accepted for Pattern Analysis and Applications, 2012.
9.	Lindblad L., Sladoje N. Coverage Segmentation based on Linear Unmixing and Minimization of Perimeter and Boundary Thickness. Pattern Recognition Letters, Vol. 33, No.6, pp. 728-738, 2012.
10.	Malmberg F., Lindblad J., Sladoje N., Nystrom I.: A graph-based framework for sub-pixel image segmentation, Theoretical Computer Science, 2011, Vol. 412, No 15, pp. 1338-1349

Summary data for teacher's scientific or art and professional activity:

Quotation total :	71			
Total of SCI(SSCI) list papers :	21			
Current projects :	Domestic :	2	International :	3

Science, arts and professional qualifications

Name and last name:	Stankovski V. Stevan
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	23.03.1987
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	2005	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	1994	School of Electrical Engineering - Beograd	Electrical and Computer Engineering
Magister thesis	1991	School of Electrical Engineering - Beograd	Electrical and Computer Engineering
Bachelor's thesis	1987	Faculty of Technical Sciences - Novi Sad	Electrical and Computer Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H105	Fundamentals in Computer science	(H00) Mechatronics, Undergraduate Academic Studies
2.	H109	Fundamentals in Programming	(H00) Mechatronics, Undergraduate Academic Studies
3.	H1403	Automation of work processes	(H00) Mechatronics, Undergraduate Academic Studies
4.	H1409	Intelligent Systems	(H00) Mechatronics, Undergraduate Academic Studies
5.	H1410	Programming and application of programmable logic controllers	(H00) Mechatronics, Undergraduate Academic Studies
6.	H1501A	Systems for Survailance and Visualisation of Process	(H00) Mechatronics, Undergraduate Academic Studies
7.	H310	Components of technological systems	(H00) Mechatronics, Undergraduate Academic Studies
8.	H311	Application of Sensors and Actuators	(H00) Mechatronics, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
9.	BM116C	Motion control	(BM0) Biomedical Engineering, Undergraduate Academic Studies
10.	BMI106	Rehabilitation devices and systems	(BM0) Biomedical Engineering, Undergraduate Academic Studies
11.	BMI110	Sensors and actuators in medicine	(BM0) Biomedical Engineering, Undergraduate Academic Studies
12.	II1009	Automatic identification systems	(I10) Industrial Engineering, Undergraduate Academic Studies
13.	II1010	Control of technical systems	(I10) Industrial Engineering, Undergraduate Academic Studies
14.	II1011	Automation of work processes 1	(I10) Industrial Engineering, Undergraduate Academic Studies
15.	II1015	Programmable Logic Controllers (PLC)	(I10) Industrial Engineering, Undergraduate Academic Studies
16.	II1038	Automation of work processes 2	(I10) Industrial Engineering, Undergraduate Academic Studies
17.	II1042	Automation of Continual Processes	(I10) Industrial Engineering, Undergraduate Academic Studies
18.	II1045	Systems for measurement, surveillance and control	(I10) Industrial Engineering, Undergraduate Academic Studies
19.	II1048	Artificial intelligence in engineering	(I10) Industrial Engineering, Undergraduate Academic Studies
20.	IM1022	Fundamentals of technical systems control	(I20) Engineering Management, Undergraduate Academic Studies (M20) Mechanization and Construction Engineering, Undergraduate Academic Studies
21.	IM1035	Identification technologies in enterprises	(I20) Engineering Management, Undergraduate Academic Studies
22.	IM1719	Implementation of information systems in insurance	(I20) Engineering Management, Undergraduate Academic Studies
23.	H505	Implementation of automated systems	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
24.	HDOS12	Research in the area of automatic identification technology	(I12) Industrial Engineering, Specialised Academic Studies
25.	HDOS13	Motion control and application of MEMS	(I12) Industrial Engineering, Specialised Academic Studies
26.	HDOS14	Nonindustrial automation	(I12) Industrial Engineering, Specialised Academic Studies
27.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
28.	MBA414	Integrated Business Processes	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
29.	PLM09	Systems and Devices for Tracking Products Through Life Cycle	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
30.	NIT02	Factory Automation	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
31.	NIT06	Advanced Technologies for Manufacturing Support	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
32.	NIT08	Fundamentals of Computer Science and Informatics	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
33.	GS006	Intelligent Buildings	(G10) Energy Efficiency in Buildings, Specialised Academic Studies
34.	H799	Fieldbuses and protocols	(H00) Mechatronics, Master Academic Studies
35.	H828	Advanced robotics	(H00) Mechatronics, Master Academic Studies
36.	H845	Motion control	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
37.	I903	Application of microelectromechanical systems	(I10) Industrial Engineering, Master Academic Studies
38.	IIDS6	Selected chapters in automation	(I12) Industrial Engineering, Specialised Academic Studies
39.	IM2516	Artificial Intelligence in Engineering	(I20) Engineering Management, Master Academic Studies
40.	IM2716	Automation systems in insurance	(I20) Engineering Management, Master Academic Studies
41.	IM2721	Systems for detection, alarming and warning	(I20) Engineering Management, Master Academic Studies
42.	GD018	Automation and Robotics in Construction	(G00) Civil Engineering, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies
43.	HDOK12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
44.	HDOK13	Motion control and the application of MEMS	(H00) Mechatronics, Doctoral Academic Studies
45.	HDOK14	Non-industrial Automation	(H00) Mechatronics, Doctoral Academic Studies
46.	HDOK-3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
47.	HDOKL3	Selected Chapters in Automation Systems Integration	(H00) Mechatronics, Doctoral Academic Studies
48.	HDOL12	Research in the area of automatic identification technologies	(H00) Mechatronics, Doctoral Academic Studies
49.	HDOL13	Motion controla and application of MEMS	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
50.	HDOL14	Nonindustrial automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
51.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
52.	IMDR80	Selected chapters in automation	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Stankovski S., Tarjan L., Škrinjar D., Ostojić G., Šenk I.: Using a Didactic Manipulator in Mechatronics and Industrial Engineering Courses, IEEE Transactions on Education, 2010, Vol. 53, No 4, pp. 572-579, ISSN 0018-9359

Representative refferences (minimum 5, not more than 10)

2.	Gajić G., Stankovski S., Ostojić G., Tešić Z., Miladinović Lj.: Method of evaluating the impact of ERP implementation critical success factors – a case study in oil and gas industries (DOI:10.1080/17517575.2012.690105), Enterprise Information Systems, 2012, ISSN 1751-7575
3.	Stankovski S., Ostojić G., Šenk I., Rakić-Skoković M., Trivunović S., Kučević D.: Dairy cow monitoring by RFID, Scientia Agricola, 2012, Vol. 69, No 1, pp. 75-80, ISSN 0103-9016
4.	Stankovski, S., Ostojić, G., Raković, M., Trajan, L., Šenk, I., Nikolić, M.: Zbirka rešenih zadataka iz: Programiranje i primena programabilno logičkih kontrolera, Fakulte tehničkih nauka, 2009
5.	Stankovski, S., Rakić-Skoković, M., Šešlija, D., Ostojić, G.: Primena RFID tehnologije u automatizaciji
6.	Stankovski S., Lazarević M., Ostojić G., Ćosić I., Purić R.: RFID Technology in Product/Part Tracking During the Whole Life Cycle , Assembly Automation, 2009, Vol. 29, No 4, pp. 364-370, ISSN 0144-5154
7.	Ostojić G., Lazarević M., Stankovski S., Ćosić I.: RFID Technology Application in Disassembly Systems , Strojniski vestnik = Journal of Mechanical Engineering, 2008, Vol. 54, No 11, pp. 759-767, ISSN 0039-2480, UDK: 658.5
8.	Popović B., Popović N., Mijić D., Stankovski S., Ostojić G.: Remote Control of Laboratory Equipment for Basic Electronics Courses: A LabVIEW-based Implementation DOI: 10.1002/cae.20531, Computer Applications in Engineering Education, 2011, ISSN 1061-3773
9.	Stankovski S., Ostojić G., Tarjan L., Škrinjar D., Lazarević M.: IML Robot Grasping Process Improvement,Iranian Journal of Science & Technology, 2011, Vol.35, No M1, pp. 197-207, Transactions B ISSN: 1028-6284
10.	Janković J., Petrović N., Miladinović Lj., Popkonstantinović B., Stoimenov M., Petrović D., Ostojić G., Stankovski S.: Computer Simulation of Fast Hydraulic Actuators,Iranian Journal of Science & Technology, Transactions B, 2012, Vol. 36, No M1, pp. 95-106, ISSN: 1028-6284

Summary data for teacher's scientific or art and professional activity:

Quotation total :	25			
Total of SCI(SSCI) list papers :	20			
Current projects :	Domestic :	3	International :	4

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Stefanović M. Darko
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.02.2001
Scientific or art field:	Information-Communication Systems

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
PhD thesis	2012	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
Magister thesis	2005	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems
Bachelor's thesis	1999	Faculty of Technical Sciences - Novi Sad	Information-Communication Systems

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II1018	Design of Information Systems	(I10) Industrial Engineering, Undergraduate Academic Studies
2.	II1039	Resource planning systems in manufacturing	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	II1049	Manufacturing documentation management (DMS)	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	IM1029	Information and communication systems	(I20) Engineering Management, Undergraduate Academic Studies
5.	IM1048	Enterprise resource planning systems	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1514	Web-oriented Technologies and Systems	(I20) Engineering Management, Undergraduate Academic Studies
7.	IMDS33	Structures of Modern Information and Communication Systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
8.	IMDS37	CAE/CAD/CAM and CIM Concepts and Systems	(I12) Industrial Engineering, Specialised Academic Studies
9.	I913	Expert systems and tools for knowledge management	(I10) Industrial Engineering, Master Academic Studies
10.	IIDS8	Selected chapters from Information, management and communication systems	(GI0) Geodesy and Geomatics, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies
11.	IM2507	Automation of production systems management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
12.	IM2515	Principles and methods of protecting data and software	(I20) Engineering Management, Master Academic Studies
13.	IM2517	e Government systems	(I20) Engineering Management, Master Academic Studies
14.	IM2522	Software testing principles and methods	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
15.	IMDS73	Selected chapters from Information management	(I22) Engineering Management, Specialised Academic Studies
16.	IMDR33	Structures of Modern Information and Communication Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
17.	IMDR73	Selected chapters from Information management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
18.	IMDR81	Selected chapters from Information, management and communication systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Prilog istraživanju uslova za integraciju savremenih ICT u poslovanju industrijskih proizvodno – poslovnih sistema
2.	Elementi savremenog pristupa planiranju efektivne proizvodnje i pripremi procesa rada – upravljanje konfiguracijama sistema.
3.	Darko Stefanović, Milan Mirkovic, Andras Anderla, Miodrag Drapsin, Patrik Drid, Izet Radjo (2011). Investigating ERP systems success from the end user perspective, TTEM - Technics Technologies Education Management, Bosnia and Herzegovina, ISSN 1840-1503, Volume 6/Number 4/2011, p. 1089-1099, IF 0,351.

Representative refferences (minimum 5, not more than 10)

4.	Darko Stefanović, Miodrag Drapšin, Jelena Nikolić, Danijela Šćepanović, Izet Radjo, Patrik Drid (2011). Empirical study of student satisfaction in e-learning system environment, TTEM - Technics Technologies Education Management, Bosnia and Herzegovina, ISSN 1840-1503, Volume 6/Number 4/2011, p. 1152-1164, IF 0,351.
5.	Andraš ANDERLA, Branko BRKLJAČ, Darko STEFANOVIĆ, Cvijan KRSMANOVIĆ, Srđan SLADOJEVIĆ, Dubravko ĆULIBRK (2013). 3D RECONSTRUCTION FROM MRI IMAGES. Metalurgia International, ISSN 1582-2214, no. 4-2013.
6.	Luković Ivan, Ristić Sonja, Stefanović Darko, Rakić Marija:Osnove racunarskih tehnologija i programiranje, FTN Izdavaštvo, Novi Sad, 2007., Univerzitet u Novom Sadu – Fakultet tehničkih nauka, Edicija Tehničke nauke – udžbenici, ISBN 978-86-7892-087-5, COBISS.SR-ID 228166407
7.	Suzić N., Anderla A., Stefanović D., Veža I., Sremčev N. (2012). Successsfully Solving the Configuration of Mass Customized Products, Proceedings – the Seventh International Symposium “KOD 2012”, 24. – 26. May 2012, Balaton Fured, Hungary, Faculty of Technical Sciences, Novi Sad, Serbia, p. 75-78, 978-86-7892-399-9
8.	Stefanović D., Rakić Skoković M., Mirković M., Anderla A., Rašić D. (2011). Contemporary Software Business Suites as a Company’s Competitive Advantage, Proceedings / XV International Scientific Conference on Industrial Systems (IS'11), Novi Sad, Serbia, p. 240-246, 978-86-7892-341-8
9.	Rakić-Skoković M., Stefanović D., Krsmanović C. (2011). Paradigms and Approaches in Development and Implementation of Enterprise Information Systems in the Future, Proceedings / XV International Scientific Conference on Industrial Systems (IS'11), Novi Sad, Serbia, p. 247-253, 978-86-7892-341-8
10.	Milan Mirković, Dubravko Ćulibrk, Andraš Anderla, Darko Stefanović, Stevan Milisavljević (2011). A framework for obtaining publicly available geo-referenced video meta-data, Proceedings / XV International Scientific Conference on Industrial Systems (IS'11), Novi Sad, Serbia, p. 223-228, 978-86-7892-341-8

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	3			
Current projects :	Domestic :	1	International :	0

Science, arts and professional qualifications

Name and last name:	Stojaković M. Mila
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.12.1975
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	1993	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	1980	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	1978	Faculty of Mathematics - Beograd	Mathematical Sciences
Bachelor's thesis	1975	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	E121	Mathematical Analysis 2	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
2.	E135	Probability, Statistics and Stochastic Processes	(MR0) Measurement and Control Engineering, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
3.	E221A	Mathematical Analysis 2	(E20) Computing and Control Engineering, Undergraduate Academic Studies (MR0) Measurement and Control Engineering, Undergraduate Academic Studies
4.	E224A	Probability and Stochastic Processes	(E20) Computing and Control Engineering, Undergraduate Academic Studies (ES0) Power Software Engineering, Undergraduate Academic Studies (SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
5.	ZC006	Probability, Statistics and Random Processes	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
6.	0M504	Operational Research	(OM1) Mathematics in Engineering, Master Academic Studies
7.	0M505	Stochastic Processes	(OM1) Mathematics in Engineering, Master Academic Studies
8.	0ML504	Operational Research	(OM1) Mathematics in Engineering, Master Academic Studies
9.	0ML505	Stochastic Processes	(OM1) Mathematics in Engineering, Master Academic Studies
10.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
11.	IAM005	Mathematical Game Theory	(F20) Engineering Animation, Master Academic Studies (OM1) Mathematics in Engineering, Master Academic Studies
12.	SD0M03	Operational Research	(GI0) Geodesy and Geomatics, Specialised Academic Studies
13.	SD0M15	Statistics	(GI0) Geodesy and Geomatics, Specialised Academic Studies
14.	ZR503	Statistical Advanced Models	(Z01) Safety at Work, Master Academic Studies
15.	D0M03	Operational Research	(OM1) Mathematics in Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
16.	D0M04	Random Processes	(OM1) Mathematics in Engineering, Doctoral Academic Studies
17.	D0M15	Statistics	(OM1) Mathematics in Engineering, Doctoral Academic Studies
18.	D0M27	StatisticsApplied in Engineering	(OM1) Mathematics in Engineering, Doctoral Academic Studies
19.	DAU004	Selected Chapters in Mathematics 2	(E20) Computing and Control Engineering, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies
20.	DOM59	Fixed point theory	(OM1) Mathematics in Engineering, Doctoral Academic Studies
21.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Mila Stojaković, Decomposition and representation of fuzzy valued measure, Fuzzy Sets and Systems, 112(2000) 251-256
2.	Mila Stojaković, Fuzzy conditional expectation, Fuzzy Sets and Systems, 52(1992) 49-54
3.	Mila Stojaković, Fuzzy random variable, expectation, martingales, J.Math.Anal.Appl., 184(1994) 594-606.
4.	Mila Stojaković, Fuzzy martingales, Stochastic Analysis and Applications, 14(1996), 355-368.
5.	Mila Stojaković, Zoran Stojaković, Support function for fuzzy set, Proceedings of Royal Society, London A, 452(1996), 421-438.
6.	Mila Stojaković, Zoran Stojaković, Addition and series of fuzzy sets, Fuzzy Sets and Systems, 83(1996) 341-346.
7.	Mila Stojaković, Representation of fuzzy valued mappings, Fuzzy Sets and Systems, 98(1998) 375-381.
8.	Mila Stojaković, Fuzzy valued measure, Fuzzy Sets and Systems,65(1994) 95-104 .
9.	Mila Stojaković, Common fixed point theorems in complete metric and probabilistic spaces,Bull. Australian Math. Soc.,36(1987)73-88.
10.	Mila Stojaković, Zoran Ovcin,Fixed point theorems and variational principle..., Fuzzy Sets and Systems, 66(1994)353-356.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	71			
Total of SCI(SSCI) list papers :	16			
Current projects :	Domestic :	1	International :	1

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Šešlija D. Dragan
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.06.1985
Scientific or art field:	Mechatronics, Robotics and Automation and Integral Systems

Academic carieer	Year	Institution	Field
Academic title election:	2007	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Integral Systems
PhD thesis	1997	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Intelligent Systems
Magister thesis	1989	Faculty of Technical Sciences - Novi Sad	Mechatronics, Robotics and Automation and Intelligent Systems
Bachelor's thesis	1981	Faculty of Technical Sciences - Novi Sad	Internal Combustion Engines

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H1401	Material Handling Technologies	(H00) Mechatronics, Undergraduate Academic Studies
2.	H1403	Automation of work processes	(H00) Mechatronics, Undergraduate Academic Studies
3.	H1504	Computer Integration of Production Systems	(H00) Mechatronics, Undergraduate Academic Studies
4.	H310	Components of technological systems	(H00) Mechatronics, Undergraduate Academic Studies
5.	II102	The basic theory of industrial systems	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
6.	II1000	Fundamentals of industrial engineering and management	(I10) Industrial Engineering, Undergraduate Academic Studies
7.	II1011	Automation of work processes 1	(I10) Industrial Engineering, Undergraduate Academic Studies
8.	II1013	Material Handling Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
9.	II1029	Computer integrated manufacturing	(I10) Industrial Engineering, Undergraduate Academic Studies
10.	II1038	Automation of work processes 2	(I10) Industrial Engineering, Undergraduate Academic Studies
11.	II1042	Automation of Continual Processes	(I10) Industrial Engineering, Undergraduate Academic Studies
12.	IM1001	Fundamentals of industrial engineering	(I20) Engineering Management, Undergraduate Academic Studies
13.	IM1117	Computer integrated manufacturing (CIM)	(I20) Engineering Management, Undergraduate Academic Studies
14.	H505	Implementation of automated systems	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
15.	HDOK4S	Selected chapters from automation of work processes	(I12) Industrial Engineering, Specialised Academic Studies
16.	I829	Automation of packaging processes	(I10) Industrial Engineering, Master Academic Studies
17.	I830	Energy efficiency of compressed air systems	(I10) Industrial Engineering, Master Academic Studies
18.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
19.	PLM04	Sustainable Production and LCA	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
20.	LIM34	Material Handling	(LIM) Logistic Engineering and Management, Master Academic Studies
21.	NIT02	Factory Automation	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
22.	NIT05	Advanced Technology for Material Handling	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
23.	BMIM4C	Fluid filtration and separation	(BM0) Biomedical Engineering, Master Academic Studies
24.	I911	Sustainable production	(I10) Industrial Engineering, Master Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
25.	IIDS27	Selected chapters of the energy efficiency of automated systems	(I12) Industrial Engineering, Specialised Academic Studies
26.	IIDS6	Selected chapters in automation	(I12) Industrial Engineering, Specialised Academic Studies
27.	IM2103	New technologies in engineering and management	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
28.	HDOK-4	Selected Chapters in Production Process Automation	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
29.	HDOKL4	Selected chapters from automation of work processes	(H00) Mechatronics, Doctoral Academic Studies
30.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
31.	IMDR86	Selected chapters from energy efficiency of compressed air systems	(H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
32.	IMDR80	Selected chapters in automation	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Ignjatović I., Komenda T., Šešlija D., Malisa V.: Optimisation of compressed air and electricity consumption in a complex robotic cell, Robotics and Computer-integrated Manufacturing, 2012, ISSN 0736-5845
2. Dudić S., Ignjatović I., Šešlija D., Blagojević V., Miodrag S.: Leakage quantification of compressed air using ultrasound and infrared thermography, MEASUREMENT, 2012, Vol. 45, No 7, pp. 1689-1694, ISSN 0263-2241
3. Ignjatović I., Šešlija D., Tarjan L., Dudić S.: Wireless sensor system for monitoring of compressed air filters, Journal of Scientific and Industrial Research (JSIR), 2012, Vol. 71, No 5, pp. 334-340, ISSN 0022-4456
4. Dudić S., Ignjatović I., Šešlija D., Blagojević V., Stojiljković M.: Leakage quantification of compressed air on pipes using thermovision, Thermal Science, 2012, Vol. 16, No 2, pp. 621-631, ISSN 0354-9836
5. Čajetinac S., Šešlija D., Aleksandrov S., Todorović M.: PLC Controller used for PWM Control and for Identification of Frequency Characteristics of a Pneumatic Actuator, Electronics and electrical engineering, 2012, Vol. 123, No 7, pp. 21-26, ISSN 1392-1215
6. Blagojević V., Šešlija D., Stojiljković M., Dudić S.: Efficient control of servo pneumatic actuator system utilizing by-pass valve and digital sliding mode, Sadhana - Academy Proceedings in Engineering Science, 2012, ISSN 0256-2499
7. Blagojević V., Šešlija D., Miodrag S.: Cost effectiveness of restoring energy in execution part of pneumatic system, Journal of Scientific and Industrial Research, 2011, Vol. 70, pp. 170-176, ISSN 0022-4456
8. Šešlija D., Ignjatović I., Dudić S., Lagod B.: Potential energy savings in compressed air systems in Serbia, African Journal of Business Management, 2011, Vol. 5, No 14, pp. 5637-5645, ISSN 1993-8233
9. Šešlija D., Ignjatović I., Dudić S.: Increasing the Energy Efficiency in Compressed Air Systems, Rijeka, InTech, 2012, str. 151-174, ISBN 978-953-51-0800-9
10. Stankovski S., Šešlija D., Rakić-Skoković M., Ostojić G.: Primena RFID tehnologije u automatizaciji, Novi Sad, Centar za automatizaciju i mehatroniku, 2009, ISBN 978-86-907827-3-4

Summary data for teacher's scientific or art and professional activity:

Quotation total :	10			
Total of SCI(SSCI) list papers :	10			
Current projects :	Domestic :	0	International :	3

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Šević D. Dragoljub
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.03.2001
Scientific or art field:	Quality, Effectiveness and Logistics

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
PhD thesis	2010	Faculty of Technical Sciences - Novi Sad	Quality, Effectiveness and Logistics
Magister thesis	2004	Faculty of Technical Sciences - Novi Sad	Mechanical Engineering
Bachelor's thesis	1999	Faculty of Technical Sciences - Novi Sad	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	II323	Environmental management system	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
2.	II1016	Reliability of technical systems and Maintenance	(I10) Industrial Engineering, Undergraduate Academic Studies
3.	II1025	Design, Verification and Analysis of the Environmental Management System	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	II1040	Organization and mamanagement of maintenance	(I10) Industrial Engineering, Undergraduate Academic Studies
5.	II1043	Maintenance techniques and technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
6.	IM1036	Reliability Theory	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1037	Environmental Management System	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1615	Maintenance of Technical Equipment	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1620	Reverse and Green Logistic	(I20) Engineering Management, Undergraduate Academic Studies
10.	I501	Risk Management	(I10) Industrial Engineering, Master Academic Studies
11.	I841	Spare parts management	(I10) Industrial Engineering, Master Academic Studies
12.	IMDS95	Trends in Customer Relationship Management	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
13.	PLM10	Product Servicing and Maintenance	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
14.	LIM31	Reverse and Green Logistics	(LIM) Logistic Engineering and Management, Master Academic Studies
15.	IIDS12	Quality and organizational performance	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
16.	IIDS30	Trends in the environmental management systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
17.	IIDS7	Selected topics in quality engineering and logistics	(I12) Industrial Engineering, Specialised Academic Studies
18.	IM2607	Risk management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
19.	IM2620	Lean Maintenance	(I10) Industrial Engineering, Master Academic Studies (I20) Engineering Management, Master Academic Studies
20.	IMDS74	Selected Topics in Quality Management and Logistics	(I22) Engineering Management, Specialised Academic Studies
21.	ZP516	Technical Systems Reliability	(ZP1) Disaster Risk Management and Fire Safety, Master Academic Studies
22.	IMDR94	Trends in the environmental management systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
23.	IMDR95	Trends in Customer Relationship Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
24.	IMDR74	Selected Topics in Quality Management and Logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	IMDR79	Selected topics in quality engineering and logistics	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	IMDR83	Quality abd organisational performance	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Brkljač N., Šević D., Beker I., Kesić I., Milisavljević S.: Procedure for treatment of hazardous waste by MID-MIX procedure in Serbia, International Journal of the Physical Sciences, 2012, Vol. 7, No 18, pp. 2639-2646, ISSN 1992-1950
2.	Jocanović M., Šević D., Karanović V., Beker I., Dudić S.: Increased Efficiency of Hydraulic Systems Through Reliability Theory and Monitoring of System Operating Parameters, Strojniški vestnik - Journal of Mechanical Engineering, 2012, Vol. 58, No 4, pp. 281-288, ISSN 0039-2480
3.	D. Šević, I. Beker „Projektovanje greda na bazi pouzdanosti“, Naučno – stručni skup ISTRAŽIVANJE I RAZVOJ MAŠINSKIH ELEMENATA I SISTEMA – Jahorina – IRMES 2002., Srpsko Sarajevo – Jahorina, Septembar 2002
4.	Šević D., Uščebrka G., Milisavljević S., Brkljač N.: MODEL VREDNOVANJA ZNAČAJNOSTI UTICAJA NA ŽIVOTNU SREDINU SA STANOVNIŠTVA ZAHTEVA STANDARDA ISO 14001:2004, UDK: 658.5
5.	Šević D., Stefanović N., Prokopić L.: Upotreba podataka i informacija koji se odnose na vrednovanje učinka na zaštiti životne sredine, International Journal Total Quality Management
6.	Beker I., Stanivuković D., Šević D.: Postupak za ocenu uspešnosti održavanja , 26. Majski skup održavalaca Jugoslavije, Novi Sad: Fakulte tehničkih nauka, 1 Maj, 2002, str. 87-93, UDK: 621-772
7.	D. Šević, I. Beker, S. Milisavljević „Uporedna analiza zahteva standarda ISO 14001:2004 i standarda ISO 14001:1996“, Menadžment totalnim kvlaitetom & izvrsnost – European Quality Week, Novi Sad, October 31st - November 2nd 2006
8.	RAZVOJ MODELA INTEGRALNOG SISTEMA, Novi Sad, 2004
9.	RAZVOJ MODELA UPRAVLJANJA LOGISTIČKIM PROCESIMA NA BAZI PROCESNOG PRILAZA, ODRŽIVOG RAZVOJA I SISTEMA UPRAVLJANJA ZAŠTITOM ŽIVOTNE SREDINE
10.	Stanivuković D., Kamberović B., Beker I., Šević D.: TENDENCIJE RAZVOJA KVALITETA, POUZDANOSTI, ODRŽAVANjA I LOGISTIKE Naziv skupa: XII međunarodna konferencija IS 2002, Vrnjačka Banja, 2002. , 12. International Scientific Conference on Industrial Systems - IS, Vrnjačka Banja: Institut za industrijske sisteme, FTN, Novi Sad, 22-23 Novembar, 2002, pp. 75-89

Summary data for teacher's scientific or art and professional activity:

Quotation total :	0			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	1	International :	1

Science, arts and professional qualifications

Name and last name:	Šormaz N. Dušan
Academic title:	Guest Professor
Name of the institution where the teacher works full time and starting date:	-
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2009		Production Systems, Organization and Management
Magister thesis	1995	University of Southern California - Nepoznato	Computer Science
PhD thesis	1994	University of Southern California - Nepoznato	Engineering Management
Magister thesis	1985	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1979	Faculty of Technical Sciences - Novi Sad	Plastic Deformation Technology

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	H1403	Automation of work processes	(H00) Mechatronics, Undergraduate Academic Studies
2.	H1504	Computer Integration of Production Systems	(H00) Mechatronics, Undergraduate Academic Studies
3.	H310	Components of technological systems	(H00) Mechatronics, Undergraduate Academic Studies
4.	II102	The basic theory of industrial systems	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
5.	II1000	Fundamentals of industrial engineering and management	(I10) Industrial Engineering, Undergraduate Academic Studies
6.	II1013	Material Handling Technologies	(I10) Industrial Engineering, Undergraduate Academic Studies
7.	IM1719	Implementation of information systems in insurance	(I20) Engineering Management, Undergraduate Academic Studies
8.	EE546	Entrepreneurship in Electrical Engineering	(E10) Power, Electronic and Telecommunication Engineering, Master Academic Studies
9.	H505	Implementation of automated systems	(H00) Mechatronics, Master Academic Studies (I10) Industrial Engineering, Master Academic Studies
10.	I829	Automation of packaging processes	(I10) Industrial Engineering, Master Academic Studies
11.	I830	Energy efficiency of compressed air systems	(I10) Industrial Engineering, Master Academic Studies
12.	IMDS56	Product traceability during the lifetime	(I12) Industrial Engineering, Specialised Academic Studies
13.	IMDS57	Strategic Planning and Designing Procedures and Systems at the End of Product Lifecycle	(I12) Industrial Engineering, Specialised Academic Studies
14.	IMDS62	Integration of business processes of companies	(I22) Engineering Management, Specialised Academic Studies
15.	IMDS93	Virtual Enterprises and Collaborative Systems	(I22) Engineering Management, Specialised Academic Studies
16.	LIM34	Material Handling	(LIM) Logistic Engineering and Management, Master Academic Studies
17.	NIT02	Factory Automation	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
18.	NIT05	Advanced Technology for Material Handling	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
19.	NIT08	Fundamentals of Computer Science and Informatics	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
20.	I911	Sustainable production	(I10) Industrial Engineering, Master Academic Studies
21.	IIDS10	Effective technological and production structures	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
22.	IIDS9	Effective Production and Service Systems	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
23.	IM2315	Product and Process Improvement Projects	(I20) Engineering Management, Master Academic Studies
24.	IMDR31	Effective Production and Service Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
25.	IMDR56	Traceability of Product Lifecycle	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	IMDR62	Enterprise Business Process Integration	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
27.	IMDR93	Virtual Enterprises and Collaborative Systems	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
28.	IMDR85	Effective technological and production structures	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Sormaz DN, Arumugam J, Ganduri C, 2007, Integration of rule-based process selection with virtual machining for distributed manufacturing planning, Process Planning and Scheduling for Distributed Manufacturing, 61-90
2.	Šormaz DN, Arumugam J, Harihara RS, Patel C, Neerukonda N, 2010, Integration of product design, process planning, scheduling, and FMS control using XML data representation, Robotics and Computer-Integrated Manufacturing 26 (6), 583-595
3.	Šormaz DN, Rajaraman SN, 2008, Problem space search algorithm for manufacturing cell formation with alternative process plans, International Journal of Production Research 46 (2), 345-369
4.	Sormaz DN, Arumugam J, Rajaraman S, 2004, Integrative process plan model and representation for intelligent distributed manufacturing planning, International Journal of Production Research, Vol. 42, No. 17, p. 3397 - 3417.
5.	Koonce D, Judd R, Sormaz D, Masel DT, 2003, A hierarchical cost estimation tool, Computers in Industry 50 (3), 293-302
6.	Sormaz DN, Khoshnevis B, 2003, Generation of alternative process plans in integrated manufacturing systems, Journal of Intelligent Manufacturing 14 (6), 509-526
7.	Šormaz DN, Tennety C, 2010, Recognition of interacting volumetric features using 2D hints, Assembly Automation 30 (2), 131-141
8.	Sormaz DN, Pisipati DV, Borse PA, 2006, Virtual manufacturing of milling operations with multiple tool paths, International journal of manufacturing technology and management 9 (3), 237-264
9.	Sormaz DN, Khoshnevis B, 2000, Modeling of manufacturing feature interactions for automated process planning, Journal of manufacturing systems, 19 (1), 28-45
10.	Nešić S, Li H, Huang J, Sormaz D, 2009, An open source mechanistic model for CO2/H2S Corrosion of carbon steel, CORROSION 2009, March 22 - 26, 2009 , Atlanta, GA

Summary data for teacher's scientific or art and professional activity:

Quotation total :	126			
Total of SCI(SSCI) list papers :	10			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Teofanov Đ. Ljiljana
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	18.12.1995
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2009	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	2008	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	2000	Faculty of Sciences - Novi Sad	Mathematical Sciences
Bachelor's thesis	1994	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	A101	Mathematics	(A00) Architecture, Undergraduate Academic Studies
2.	EE204	Selected Chapters in Mathematics	(MR0) Measurement and Control Engineering, Undergraduate Academic Studies (E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
3.	GG00	Mathematical Methods 1	(G00) Civil Engineering, Undergraduate Academic Studies
4.	GI101	Algebra	(GI0) Geodesy and Geomatics, Undergraduate Academic Studies
5.	IAM001	Mathematical Shape Modeling for Computer Animation	(F10) Engineering Animation, Undergraduate Academic Studies
6.	M102	Mathematics 1	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
7.	M106	Mathematics 2	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies (P00) Production Engineering, Undergraduate Academic Studies
8.	E101A	Discrete Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Undergraduate Academic Studies
9.	IM1523	Discrete Mathematics	(M30) Energy and Process Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
10.	P216	Numerical Analysis	(P00) Production Engineering, Undergraduate Academic Studies
11.	SE0009	Discrete Mathematics	(SE0) Software Engineering and Information Technologies, Undergraduate Academic Studies (SEL) Software Engineering and Information Technologies - Loznica, Undergraduate Academic Studies
12.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
13.	IA022	Numerical Optimization	(F20) Engineering Animation, Master Academic Studies
14.	D0M48	Numerical Methods for Solving Differential Equations	(OM1) Mathematics in Engineering, Doctoral Academic Studies
15.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Surla, K., Teofanov, Lj., Uzelac, A Robust Layer-Resolving Spline Collocation Method for a Convection-Diffusion Problem, Applied Mathematics and Computation,(2009), 208(1): 76-89
2. Teofanov, Lj., Roos, H. -G, An elliptic singularly perturbed problem with two parameters II: robust finite element solution, J. Comput. Appl. Math. Vol. 212, 2008, 374-389
3. Teofanov, Lj., Roos, H. -G, An elliptic singularly perturbed problem with two parameters I: solution decomposition, J. Comput. Appl. Math. Vol. 206, 2007, 1082-1097
4. Surla, K., Uzelac, Z., Teofanov, Lj., The discrete minimum principle for quadratic spline discretization of a singularly perturbed problem, Math. Comput. Simul. 2009, Vol. 79, No 8, pp.2490-2505
5. Teofanov, Lj., Zarin, H., Superconvergence for two-parameter singularly perturbed problem, BIT Numerical Mathematics, Vol. 49, No. 4, 2009, 743-765
6. Vulanović, R., Teofanov, Lj., A uniform numerical method for semilinear reaction-difusion problems with a boundary turning point, Numer. Algor. 54, 2010, 431-444
7. Teofanov, Lj., Uzelac, Z., Family of Quadratic Spline Difference Schemes for a Convection-Diffusion Problem, Int. J. Comput. Math., Vol. 84, No. 1, 2007, 33-50
8. Surla, K., Uzelac, Z., Teofanov, Lj., On collocation methods for singular perturbation problems of convection-diffusion type, Novi Sad J. Math, Vol. 31, No. 1, 2001, 125-132
9. Surla, K., Uzelac, Z., Pavlović, Lj., On collocation methods for singular perturbation problems, Novi Sad J. Math., Vol. 30, No. 3, 2000, 173-183
10. Čomić, I., Pavlović, Lj., Funkcije više promenljivih, Fakultet tehničkih nauka, Novi Sad, 2000, 95 str.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	12			
Total of SCI(SSCI) list papers :	7			
Current projects :	Domestic :	1	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Tešić M. Zdravko
Academic title:	Associate Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	02.10.1981
Scientific or art field:	Production Systems, Organization and Management

Academic carieer	Year	Institution	Field
Academic title election:	2011	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
PhD thesis	2006	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	1989	Faculty of Technical Sciences - Novi Sad	Engineering Management
Bachelor's thesis	1982	Faculty of Technical Sciences - Novi Sad	Mechanical Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	IM1044	Business process integration	(I20) Engineering Management, Undergraduate Academic Studies
2.	IM1101	Production planning and control	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
3.	IM1115	Business process modelling	(I20) Engineering Management, Undergraduate Academic Studies
4.	IMDR0S	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
5.	IMDS14	Production planning and control	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
6.	IMDS62	Integration of business processes of companies	(I22) Engineering Management, Specialised Academic Studies
7.	IMDS63	Intelligent Organisation	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
8.	IS001	Effective management	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
9.	MBA414	Integrated Business Processes	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
10.	MBA604	E-Commerce and Electronic Payment System	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	PLM03	Information System for PLM	(I1U) Industrial Engineering - Product Lifecycle Management and Development, Master Academic Studies
12.	LIM32	ERP Systems	(LIM) Logistic Engineering and Management, Master Academic Studies
13.	I901	Manufacturing performace measurement	(I10) Industrial Engineering, Master Academic Studies
14.	I905	Enterprise integration	(I10) Industrial Engineering, Master Academic Studies
15.	IIDS10	Effective technological and production structures	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies
16.	IIDS31	Production management structures	(I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
17.	IIDS5	Selected chapters in enterprise's design, organization and control	(I12) Industrial Engineering, Specialised Academic Studies
18.	IM2101	Intelligent Enterprising and Effective Management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
19.	IM2107	SAP Enterprise systems	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
20.	IM2120	Virtual Enterprises	(I20) Engineering Management, Master Academic Studies
21.	IM2318	ERP systems	(I20) Engineering Management, Master Academic Studies
22.	IMDS69	Selected chapters in enterprise's design, organization and control	(I22) Engineering Management, Specialised Academic Studies
23.	PLM03	Information System for Product Lifecycle Management - PLM	(I20) Engineering Management, Specialised Professional Studies
24.	IMDR0	Science of Industrial Engineering and Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
25.	IMDR14	Selected Approach in Production Process Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
26.	IMDR38	Production control structure	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
27.	IMDR62	Enterprise Business Process Integration	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
28.	IMDR63	Intelligent Organisation	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
29.	IMDR5	Selected chapters in enterprise's design, organization and control	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
30.	IMDR69	Selected chapters of enterprise's management and control	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
31.	IMDR85	Effective technological and production structures	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Zelenović D., Tešić Z.: PERIOD BATCH CONTROL AND GROUP TECHNOLOGY, Interntional Journal of Production Research, 1988, Vol. 26, No. 3, str. 539- 552, UDK: xxx, ISSN 0020-7543.
2. Tešić Z., Maksimović R., Radaković N. Razvoj modela integrisanih poslovnih procesa u industrijskom preduzeću, SYMORG 2006, Beograd, Fakultet organizacionih nauka, 7-10.jun 2006, pp 158-161, UDK:005, ISBN 86-7680-086-3
3. Tešić Z., Šešlija D. Prilog razvoju komunikacije između upravljačkih sistema tehnoloških sistema i sistema za upravljanje proizvodnjom, HIPNEF2004, Niš, Mašinski fakultet Niš, 19-21. maj 2004, pp 499-504, UDK:681.5, ISBN 86-80587-31-1
4. Šešlija D., Odri S., Tešić Z., Stankovski S. Bridging the gap between machine and production control system, Facta Universitates, 2005, Vol.3, No.1, pp 81-92. ISSN 0354-2025
5. Šešlija D., Tešić Z. RFID MIDDLEWARE AS A CONNECTION BETWEEN MANUFACTURING PROSESSES AND ENTERPRISE LEVEL INFORMATION SYSTEM, FACTA UNIVERSITATIS, SERIES MECHANICAL ENGINEERING, UDC 681.518:65.011.56 , Vol.4, No 1, pp. 63 – 74, 2006.
6. Šešlija D., Odri S., Tešić Z., Stankovski S. oN THE COMMUNICATION BETWEEN MACHINE AND PRODUCTION CONTROL SYSTEM, International Scientific Conference UNITECH, GABROVO,2004, pp 229-232, ISBN 954-683-304-5
7. Tešić, Z., Ćosić, I., Mitrović, V., Lalić, D.:Integration of information for manufacturing shop control, Journal of Mechanical Engineering - Strojniski Vestnik, 2010, Vol.56, No,3, pp 217-223, ISBN 0039-2480.
8. Golišin, M., Tešić, Z., Ostojić, A.: The analysis of the renewable energy production sector in Serbia, Renewable and Sustainable Energy Rewiews, 2010, Vol.14, No.5, pp 1477-1483, ISSN 1364-0321
9. Lalić d., Popovski k., Gecevska V., Tešić Z. Analysis of the opportunities and challenges for renewable energy market in the Western Balkan countries, Renewable and Sustainable Energy Reviews, 2011, Vol. 15, pp 3187-3195.ISSN: 1364-0321
10. Gajić G., Stankovski S., Ostojić G., Tešić Z., Miladinović Lj. Method of evaluating the impact of ERP implementation critical success factor - a case study in oil and gas industries, Enterprise information systems, 2012, Vol 0, 1-23. ISSN 1751-7575.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	30			
Total of SCI(SSCI) list papers :	5			
Current projects :	Domestic :	2	International :	2

Science, arts and professional qualifications

Name and last name:	Uzelac S. Zorica
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.1975
Scientific or art field:	Mathematics

Academic carieer	Year	Institution	Field
Academic title election:	2000	Faculty of Technical Sciences - Novi Sad	Mathematics
PhD thesis	1989	Faculty of Sciences - Novi Sad	Mathematical Sciences
Magister thesis	1980	Faculty of Mathematics - Beograd	Mathematical Sciences
Bachelor's thesis	1974	Faculty of Sciences - Novi Sad	Mathematical Sciences

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	GG00	Mathematical Methods 1	(G00) Civil Engineering, Undergraduate Academic Studies
2.	GG05	Mathematical Methods 2	(G00) Civil Engineering, Undergraduate Academic Studies
3.	II1052	Mathematics 2	(I10) Industrial Engineering, Undergraduate Academic Studies
4.	IM1002	Mathematics 1	(I10) Industrial Engineering, Undergraduate Academic Studies (I20) Engineering Management, Undergraduate Academic Studies
5.	IM1006	Mathematics 2	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1120	Knowledge management	(I20) Engineering Management, Undergraduate Academic Studies
7.	0M518	Numerical Solutions of Differential Equations	(OM1) Mathematics in Engineering, Master Academic Studies
8.	0ML518	Numerical Solution of Differential Equations	(OM1) Mathematics in Engineering, Master Academic Studies
9.	DZ01MS	Selected Chapters in Mathematics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
10.	HR013	Knowledge Economy	(I20) Engineering Management, Specialised Professional Studies (IB0) Engineering Management - MBA, Specialised Professional Studies
11.	MBA309	Human Resource Management in Knowledge Economy	(IB0) Engineering Management - MBA, Specialised Professional Studies
12.	OIR010	Mathematics for Business and Finance	(I20) Engineering Management, Specialised Professional Studies
13.	IA022	Numerical Optimization	(F20) Engineering Animation, Master Academic Studies
14.	D0M16	Differential Equations	(OM1) Mathematics in Engineering, Doctoral Academic Studies
15.	D0M18	Numerical Analysis	(OM1) Mathematics in Engineering, Doctoral Academic Studies
16.	DM322	Numeric Methods in Power Machines and Plants	(M00) Mechanical Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
17.	DZ01M	Selected Chapters in Mathematics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Surla K., Teofanov Lj., Uzelac Z.: A robust layer-resolving spline collocation method for a convection-diffusion problem, Applied Mathematics and Computation, 2009, Vol. 208, No 1, pp. 76-89, ISSN 0096-3003
2.	Surla K., Uzelac Z., Teofanov Lj.: The discrete minimum principle for quadratic spline discretization of a singularly perturbed problem, Math. Comput. Simul, 2009, Vol. 79, No 8, pp. 2490-2505, ISSN 0378-4754
3.	Surla, K., Uzelac, Z., Some uniformly convergent spline difference schemes for singularly perturbed boundary value problems, IMA J. Numer. Anal.10(1990) 209-222
4.	Sekulić, D., Edeskuty, F.J.,Uzelac, Z., Heat Transfer Through a High Temperature Superconducting Current Lead at Criogenic temperatures, Int.J. Heat Mass Transfer, Vol. 40,No 16, 1997, 3917-3926,
5.	Uzelac, Z., Surla, K., Discretization of the Semilinear Singularly Perturbed Problem, Nonlinear Analysis: Theory, Methods and Applications, Vol.30, No.8, (1997), 4741-4747
6.	Sekulic, D., Uzelac, Z., Edeskuty, F., J., Entropy generation in a high temperaturesuperconducting current lead, Cryogenics, Vol 32(1992) 1154-1161
7.	Cvetićanin, L., Uzelac, Z., Longitudinal Vibration of Rod with Non-Linear Constitutive Equation, Journal of Vibration and Control,5, (1999), 827-849
8.	Teofanov, Lj., Uzelac, Z., Family of Quadratic Spline Difference Schemes for a Convection-Diffusion Problem, International Journal of Computer Mathematics, Vol. 84, No. 1, 2007, 33-50
9.	Z. Uzelac, L. Nešić, D. Hristić, A Contribution to Research the Caracteristics of Women Managers and a New Style of Leadedrship, Proceedings of IC-Congress, Haarlem, The Netherlands, 3-4. May 2007
10.	Dj. Ćelić, Z. Uzelac, Vrednosne mreže, Zborniki radova XIII Medjunarodna konferncija industrijski sistemi-IS05, Herceg Novi, 07-09. septembar, 2005, 921-931

Summary data for teacher's scientific or art and professional activity:

Quotation total :	52			
Total of SCI(SSCI) list papers :	26			
Current projects :	Domestic :	1	International :	0

Science, arts and professional qualifications

Name and last name:	Vilotić Ž. Dragiša
Academic title:	Full Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.01.1975
Scientific or art field:	Plastic Deformation Technology, Rapid Prototyping, Virtual

Academic carieer	Year	Institution	Field
Academic title election:	1998	Faculty of Technical Sciences - Novi Sad	Plastic Deformation Technology, Rapid Prototyping, Virtual
PhD thesis	1986	Faculty of Technical Sciences - Novi Sad	Plastic Deformation Technology, Rapid Prototyping, Virtual
Magister thesis	1981	Faculty of Technical Sciences - Novi Sad	Plastic Deformation Technology, Rapid Prototyping, Virtual
Bachelor's thesis	1974	Faculty of Technical Sciences - Novi Sad	Plastic Deformation Technology, Rapid Prototyping, Virtual

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	P207	Metal forming	(P00) Production Engineering, Undergraduate Academic Studies
2.	P2401	Advanced Methods in Metal Forming	(P00) Production Engineering, Undergraduate Academic Studies
3.	P2413	Computer Aided Design of Tools and Dies for Metal Forming	(P00) Production Engineering, Undergraduate Academic Studies
4.	P303	Machines for Processing by Deforming	(P00) Production Engineering, Undergraduate Academic Studies
5.	P3403	Technology of Plastic Forming - Shaping of plastic material	(P00) Production Engineering, Undergraduate Academic Studies
6.	P3503	Machines and Devices for Plastic Processing	(P00) Production Engineering, Undergraduate Academic Studies
7.	M2062	Mechanical engineering technologies 2	(M20) Mechanization and Construction Engineering, Undergraduate Academic Studies (M40) Technical Mechanics and Technical Design, Undergraduate Academic Studies
8.	M3203	Technology of machinery	(M30) Energy and Process Engineering, Undergraduate Academic Studies
9.	P3402	Physical and Phase States of Polymers	(P00) Production Engineering, Undergraduate Academic Studies
10.	ZR408A	Safety at work on the machines for processing	(Z01) Safety at Work, Undergraduate Academic Studies
11.	P2407	Rapid Prototyping and Rapid Tooling	(PM0) Production Engineering, Master Academic Studies
12.	P3501	Tool Designing for Plastic	(PM0) Production Engineering, Master Academic Studies
13.	P3503A	Contemporary Process Systems for Plastic Treatment	(PM0) Production Engineering, Master Academic Studies
14.	BMIM4B	Technologies of shaping biomedical materials	(BM0) Biomedical Engineering, Master Academic Studies (PM0) Production Engineering, Master Academic Studies
15.	PMISP1	Modelling and Simulation of Metal Forming Processes	(PM0) Production Engineering, Master Academic Studies
16.	PTS01	Technology of sintering	(PM0) Production Engineering, Master Academic Studies
17.	DP001	Design and Research Methods in Production Engineering	(M00) Mechanical Engineering, Doctoral Academic Studies
18.	DP005	State and Tendencies in Development of Metrology, Quality and Equipment	(M00) Mechanical Engineering, Doctoral Academic Studies
19.	DP008	Contemporary Methods and TPD Systems	(M00) Mechanical Engineering, Doctoral Academic Studies
20.	DP012	Physical Modelling and TPD Simulation by Computers	(M00) Mechanical Engineering, Doctoral Academic Studies
21.	DP015	Nonconventional Procedures of Forming in TPD	(M00) Mechanical Engineering, Doctoral Academic Studies

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES Industrial Engineering / Engineering Management

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
22.	SID04	Current State in the Field	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (F20) Engineering Animation, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies
23.	DP026	Modern methods for polymers investigation	(M00) Mechanical Engineering, Doctoral Academic Studies
24.	DP028	Theoretical basis for forming polymer technology	(M00) Mechanical Engineering, Doctoral Academic Studies
25.	SID04	Present State in the Field	(A00) Architecture, Doctoral Academic Studies (AS0) Scenic Design, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Essa K., Kačmarčik I., Hartley P., Plančak M., Vilotić D.: Upsetting of bi-metallic ring billets, Journal of Materials Processing Technology, 2012, Vol. 212, No 4, pp. 817-824, ISSN 0924-0136
2.	Alexandrov S., Vilotić D., Konjovoć Z., Vilotić M.: An Improved Experimental Method for Detrmining the Workability Diagram, Experimental Mechanics, 2012, Vol. 52, No 11340, ISSN 0014-4851
3.	Alexandrov S., Vilotić D.: A study on an effect of geometric singularities on ductile fracture , Engineering Fracture Mechanics, 2009, Vol. 76, No 14, pp. 2309-2315, ISSN 0013-7944
4.	Vilotić D., Plančak M., Čupković Đ., Aleksandrov S., Aleksandrov N.: Free Surface Fracture in Three Upsetting Tests , Experimental Mechanics, 2006, Vol. 46, pp. 115-120, ISSN 0014-4851
5.	Plančak M., Hartley P., Esssa K., Vilotić D., Movrin D., Lužanin O.: Deformation analysis during bi-metallic coining operations, Steel Research International, 2012, pp. 1247-1250, ISSN 1611-3683
6.	Vilotić D., Alexandrov S., Plančak M., Vilotić M., Ivanišević A., Kačmarčik I.: Material Formability at Upsetting by Cylindrical and Flat Dies, Steel Research International, 2012, pp. 1175-1178, ISSN 1611-3683
7.	Vilotić D., Alexandrov S., Plančak M., Movrin D., Ivanišević A., Vilotić M.: Material Formability of Upsetting by V-Shape Dies , Steel Research International, 2011, pp. 923-928, ISSN 1611-3683
8.	Lyamina E., Alexandrov S., Vilotić D., Movrin D.: Effect of Shape of Samples on Ductile Fracture Initiation in Upsetting, Steel Research International, 2010, Vol. 9, No 81, pp. 306-3090, ISSN 1611-3683
9.	D. Vilotić, D. Milikić, M. Plančak, M. Milutinović: Obrazovanje inženjera proizvodnog mašinstva iz oblasti oblikovanja plastike na Fakultetu tehničkih nauka u Novom Sadu, 4. kongres inženjera plastičara i gumara K – IPG 2006., zbornik na CDu, ppt 100 slajdova, Vršac, 13-16. juni 2006.
10.	Obradović R., Vilotić D.: Prikaz tehnologije i opreme za za ultrazvučno zavarivanje termoplastičnih komponenata, Zbornik radova MMA 2006, strana 27-28, FTN, Novi Sad, juni 2006.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	17			
Total of SCI(SSCI) list papers :	15			
Current projects :	Domestic :	1	International :	1

Science, arts and professional qualifications

Name and last name:	Vojinović-Miloradov B. Mirjana
Academic title:	Emeritus Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.01.2000
Scientific or art field:	Environment Protection Engineering

Academic carieer	Year	Institution	Field
Academic title election:	2008	Faculty of Technical Sciences - Novi Sad	Environment Protection Engineering
PhD thesis	1976	Faculty of Technology - Novi Sad	Technological Engineering
Magister thesis	1971	Faculty of Technology - Novi Sad	Technological Engineering
Bachelor's thesis	1963	Faculty of Technology - Novi Sad	Technological Engineering

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	Z503	Practical Course in Environment Protection	(Z20) Environmental Engineering, Master Academic Studies
2.	Z507	Physical and Chemical Principles	(Z20) Environmental Engineering, Master Academic Studies
3.	Z510	Accidental Risk Management and the Environment	(OM1) Mathematics in Engineering, Master Academic Studies (Z01) Safety at Work, Master Academic Studies (Z20) Environmental Engineering, Master Academic Studies
4.	ZR504	Protection against Chemical Harms, Fire and Explosion	(OM1) Mathematics in Engineering, Master Academic Studies
5.	Z507	Fizičko hemijski principi(uneti naziv na engleskom)	(Z20) Environmental Engineering, Master Academic Studies
6.	IM2819	Industrial eco-marketing	(I20) Engineering Management, Master Academic Studies
7.	IMDS82	Industrial eco-marketing management	(I22) Engineering Management, Specialised Academic Studies
8.	MPK005	Analysis of environmental protection systems	(MPK) Inženjerstvo tretmana i zaštite voda - TEMPUS(uneti naziv na engledskom), Master Academic Studies
9.	SZD050	Transport and distribution of pollutants in heterogeneous multicomponent systems	(Z00) Environmental Engineering, Specialised Academic Studies
10.	SZDO03	Applied Analysis of Physical and Chemical Parameters	(Z00) Environmental Engineering, Specialised Academic Studies
11.	SZSP09	Remediation of contaminated locations	(Z00) Environmental Engineering, Specialised Academic Studies
12.	ZR504A	Chemical risk assessment of fire and explosion	(Z01) Safety at Work, Master Academic Studies
13.	ZD050	Transport and distribution of pollutants in heterogeneous multicomponent systems	(Z00) Environmental Engineering, Doctoral Academic Studies
14.	ZDO03	Applied Analysis of Physical and Chemical Parameters	(OM1) Mathematics in Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies
15.	ZSP09	Remediation of Contaminated Sites	(Z00) Environmental Engineering, Doctoral Academic Studies
16.	IMDR82	Industrial eco-marketing management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1.	Sonja Kaišarević, Nebojša Andrić, Stanka Bobić, Jelena Tričković, Ivana Teodorović, Mirjana Vojinović-Miloradov, Radmila Z. Kovačević, Detection of Dioxin-like Contaminants in Soil from the Area of Oil Refineries in Vojvodina Region of Serbia, Bulletin of Environmental Contamination and Toxicology (2007), online, 10.1007/s00128-007-9241-4
2.	S. Pavkov, M. Vojinović, D. Buzarov, RESIDUES OF PERSISTENT ORGANOCHLORINE COMPOUNDS IN SELECTED AQUATIC ECOSYSTEMS OF VOJVODINA, Wat. Sci. Tech., 22(5), 107-111 (1990)
3.	M. Vojinović-Miloradov, P. Marjanović, D. Buzarov, S. Pavkov, L. Dimitrijević, M. Miloradov, BIOACCUMULATION OF POLYCHLORINATED BIPHENYLS AND ORGANOCHLORINE PESTICIDES IN SELECTED FISH SPECIES AS AN INDICATOR OF THE POLLUTION OF AQUATIC RESOURCES IN VOJVODINA, YUGOSLAVIA, Wat. Sci. Tech., 26(9-11), 2361-2364 (1992)
4.	Turk M, Jakšić J, Vojinović Miloradov M, Klanova J, Post-war levels of persistent organic pollutants (POPs) in air from Serbia determined by active and passive sampling methods, Environ Chem Lett (2007), 5:109-113

Representative refferences (minimum 5, not more than 10)

5.	B.Škrbić, M.Vojinović-Miloradov, A CONTIBUTION TO THE QUALITATIVE GC ANALYSIS OF SOME NON-CHLORINATED XENOBIOTIC CHEMICALS IN WASTE WATERS, Wat.Sci.Tech., 30 (3) 91-93, 1994
6.	Kovačević R., Vojnović-Miloradov M., Teodorović I. and Andrić S. EFFECT OF PCBs ON ANDROGEN PRODUCTION BY SUSPENSION OF ADULT RAT LEYDIG CELLS in vitro. J Steroid Bioch Mol Biol .52(6): 595-597 (1995)
7.	Miloradov M., Jakšić J., Turk M., Popov S., Vojinović-Miloradov M.: Integralni katastar - harmonizacija zakonske regulative sa EU zakonodavstvom, rad po pozivu, 33. nacionalna konferencija o kvalitetu, zbornik radova, ISBN 86-80581-86-0, maj 2006., str. B-45 - B-48
8.	Vojinović Miloradov M., Chriastel R.,Miloradov M., Jakšić J., Turk M.,: Joint project Serbia and Slovakia on the institutional support of integrated water pollution control, 1. međunarodni kongres ,,Ekologija, zdravlje, rad, sport"", Zbornik apstrakata, Banja Luka, jun 2006., str. 66-67.
9.	Mlić N., Milanović M., Grujić Letić N.,Turk Sekulić M., Radonić (Jakšić) J., Mhajlović I., Vojinović-Miloradov M.: Occurrence of antibiotics as emerging contaminant substances in aquatic environment DOI: 10.1080/09603123.2012.733934, INT J ENVIRON. HEAL. R., 2012, pp. 1-15, ISSN 0960-3123
10.	Grujić Letić N., Mlić N., Turk Sekulić M., Radonić (Jakšić) J., Milanović M., Mhajlović I., Vojinović-Miloradov M.: Quantification of emerging organic contaminants in the Danube River samples by HPLC, Chemicke Listy, 2012, Vol. 106, pp. 264-266, ISSN 1213-7103

Summary data for teacher's scientific or art and professional activity:

Quotation total :	120			
Total of SCI(SSCI) list papers :	25			
Current projects :	Domestic :	3	International :	3

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Vrgović D. Petar
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	01.10.2006
Scientific or art field:	Industrial Engineering and Engineering Management

Academic carieer	Year	Institution	Field
Academic title election:	2012	Faculty of Technical Sciences - Novi Sad	Industrial Engineering and Engineering Management
PhD thesis	2012	Faculty of Technical Sciences - Novi Sad	Engineering Management
Magister thesis	2009	Faculty of Technical Sciences - Novi Sad	Production Systems, Organization and Management
Bachelor's thesis	2005	Faculty of Philosophy - Novi Sad	Psychological Science

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	I409	Psychology in Management	(ZC0) Clean Energy Technologies, Undergraduate Academic Studies
2.	II934	Psichology of Work	(SII) Software and Information Technologies (Inđija), Undergraduate Professional Studies
3.	IM1017	Communicology	(I20) Engineering Management, Undergraduate Academic Studies
4.	IM1052	Engineering Ethics	(I20) Engineering Management, Undergraduate Academic Studies (M30) Energy and Process Engineering, Undergraduate Academic Studies
5.	IM1621	Quality in individual work	(I20) Engineering Management, Undergraduate Academic Studies
6.	IM1913	Research Methodology for Human Resources 1	(I20) Engineering Management, Undergraduate Academic Studies
7.	IM1915	Employee protection	(I20) Engineering Management, Undergraduate Academic Studies
8.	IM1918	Conflict Management	(I20) Engineering Management, Undergraduate Academic Studies
9.	IM1922	Value management	(I20) Engineering Management, Undergraduate Academic Studies
10.	IMDS11	Employees' creativity management	(I22) Engineering Management, Specialised Academic Studies
11.	MBA308	Business communication	(IB0) Engineering Management - MBA, Specialised Professional Studies
12.	NIT04	Communication Skills	(NIT) Industrial Engineering - Advanced Engineering Technologies, Master Academic Studies
13.	IM2214	Creative Problem Solving	(I20) Engineering Management, Master Academic Studies
14.	IM2917	Creative potentials management	(I20) Engineering Management, Master Academic Studies
15.	IM2918	Human Resources Research Methodology 2	(I20) Engineering Management, Master Academic Studies
16.	IM2920	Personnel Management	(M50) Energy Management, Master Academic Studies (I20) Engineering Management, Master Academic Studies
17.	IMDS77	Selected Chapters from Human Resource Management	(I22) Engineering Management, Specialised Academic Studies
18.	IMDR10	COGNITIVE MANAGEMENT	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
19.	IMDR11	Employees' creativity management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
20.	IMDR77	Selected Chapters from Human Resource Management	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies
21.	IMDR84	Data ACQUISITION, ANALYSIS AND INTERPRETATION 1	(I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

Representative refferences (minimum 5, not more than 10)

1.	Vrgović P., Glassman B., Walton A., Vidicki P.: Open innovation for SMEs in developing countries – an intermediated communication network model for collaboration beyond obstacles, Innovation-Management Policy and Practice, 2012, Vol. 14, No 3, pp. 290-302, ISSN 1447-9338
2.	Jošanov-Vrgović I., Savić N., Jošanov B., Vrgović P.: Development plans and the state of e-tourism: Case study in Novi Sad, African Journal of Business Management, 2011, Vol. 5, No 7, pp. 2545-2550, ISSN 1993-8233
3.	Kapor-Stanulović, N., Vrgović, P. (2009) Komunikologija za menadžere. Fakultet tehničkih nauka. Novi Sad
4.	Kapor-Stanulović Nila, Vrgović Petar, Hinić Darko. (2009) Komunikologija i komuniciranje u organizaciji. Državni univerzitet u Novom Pazaru.
5.	Vrgović Petar, Hinić Darko, Matijević Nikolina, Barać Milena. (2010) Poslovno i organizaciono komuniciranje. Fakultet za poslovni menadžment. Bar, Crna Gora.
6.	Vrgović P., Kovačević J., Mihailović D.: Effective communication and idea generation, 5. International Conference on Mass Customization and Personalization in Central Europe MCP-CE, Novi Sad: Fakultet tehničkih nauka, 19-21 Septembar, 2012, pp. 261-265, ISBN 978-86-7892-432-3.
7.	Vrgović P., Mihailović D.: Idea management in a developing country with transition economy: good intention, bad communication, 13. International symposium SymOrg, Zlatibor: Fakultet organizacionih nauka, 5-9 Jun, 2012, pp. 320-328, ISBN 978-86-7680-255-5.
8.	Vrgović P., Antonova A., Vidicki P.: Limiting innovation gaps - Building communication bridges between inventors and SMEs in developing countries, 15. International Scientific Conference on Industrial Systems - IS, Novi Sad: Fakultet tehničkih nauka, 14-16 Septembar, 2011, pp. 437-441, ISBN 978-86-7892-341-8.
9.	Vrgović Petar, Glassman Brian, Walton Abram, Vidicki Predrag, Suzić Nikola. (2010) Market Driven Inventions in SMEs - A Model for Growing Economies by Connecting Entrepreneurial Inventors with Local Companies. International Conference on Entrepreneurship, Innovation and Regional Development, p 810-817. ICEIRD (3; Novi Sad; 2010). ISBN 978-86-7892-250-3
10.	Vidicki, P. Vrgović, P.: Measuring innovation in service sector, International Scientific Conference on Industrial Systems IS"08 (14th), Novi Sad: Faculty of technical sciences, 2-3 oktobar, 2008, str. 565- 570, ISBN 978-86-7892-135-3.

Summary data for teacher's scientific or art and professional activity:

Quotation total :	1			
Total of SCI(SSCI) list papers :	2			
Current projects :	Domestic :	0	International :	0

Study Programme Accreditation - PhD Studies

DOCTORAL ACADEMIC STUDIES — Industrial Engineering / Engineering Management

Science, arts and professional qualifications

Name and last name:	Vučinić-Vasić T. Milica
Academic title:	Assistant Professor
Name of the institution where the teacher works full time and starting date:	Faculty of Technical Sciences - Novi Sad
	15.04.2000
Scientific or art field:	Physics

Academic carieer	Year	Institution	Field
Academic title election:	2007	Faculty of Technical Sciences - Novi Sad	Physics
PhD thesis	2007	Faculty of Sciences - Novi Sad	Physics
Magister thesis	2000	Faculty of Sciences - Novi Sad	Physics
Bachelor's thesis	1996	Faculty of Sciences - Novi Sad	Physics

List of courses being held by the teacher in the accredited study programmes

	ID	Course name	Study programme name, study type
1.	F102	Physics	(F00) Graphic Engineering and Design, Undergraduate Academic Studies
2.	GG06	Civil Engineering Physics	(G00) Civil Engineering, Undergraduate Academic Studies
3.	S014	Physics	(S00) Traffic and Transport Engineering, Undergraduate Academic Studies (S01) Postal Traffic and Telecommunications, Undergraduate Academic Studies
4.	DZ01FS	Selected Chapters in Physics	(E11) Power, Electronic and Telecommunication Engineering, Specialised Academic Studies (I12) Industrial Engineering, Specialised Academic Studies (I22) Engineering Management, Specialised Academic Studies (Z00) Environmental Engineering, Specialised Academic Studies
5.	DZ01F	Selected Chapters in Physics	(E10) Power, Electronic and Telecommunication Engineering, Doctoral Academic Studies (E20) Computing and Control Engineering, Doctoral Academic Studies (F00) Graphic Engineering and Design, Doctoral Academic Studies (G00) Civil Engineering, Doctoral Academic Studies (GI0) Geodesy and Geomatics, Doctoral Academic Studies (H00) Mechatronics, Doctoral Academic Studies (I20) Industrial Engineering / Engineering Management, Doctoral Academic Studies (M00) Mechanical Engineering, Doctoral Academic Studies (M40) Technical Mechanics, Doctoral Academic Studies (OM1) Mathematics in Engineering, Doctoral Academic Studies (S00) Traffic Engineering, Doctoral Academic Studies (Z00) Environmental Engineering, Doctoral Academic Studies (Z01) Safety at Work, Doctoral Academic Studies

Representative refferences (minimum 5, not more than 10)

1. Milica Vučinić-Vasić, Divko Ćirić, Tatjana Škrbić, Miroljub Đurić, Zbirka zadataka iz fizike, FTN Izdavaštvo, Novi Sad 2005.
2. Ljuba Budinski-Petković, Milica Vučinić, Dušan Ilić, Praktikum eksperimentalnih vežbi iz fizike – odsek za računarstvo i automatiku, S PRINT, Novi Sad, 2003
3. Ljuba Budinski-Petković, Milica Vučinić-Vasić, Dušan Ilić, Praktikum eksperimentalnih vežbi iz fizike – odsek za mašinstvo – odsek za grafičko inženjerstvo – odsek za mehatroniku, Delta press, Novi Sad, 2003.
4. Vučinić-Vasić M.: Exchange-Bias and Grain-Surface Relaxations in Nanostructured NiO/Ni Induced by a Particle Size Reduction, Journal of Physical Chemistry C, 2012, Vol. 116, pp. 4356-4364, ISSN 1932-7447

Representative refferences (minimum 5, not more than 10)

5.	Vučinić-Vasić M., Mihailović A., Kozmidis-Luburić U., Nemeš T., Ninkov J., Zeremski T., Antić B.: Metal contamination of short-term snow cover near urban crossroads: Correlation analysis of metal content and fine particles didtribution, Chemosphere, 2012, Vol. 6, No 86, pp. 585-592
6.	Kremenović A., Jančar B., Ristić M., Vučinić-Vasić M., Rogan J., Pacevski A., Antić B.: Exchange-Bias and Grain-Surface Relaxations in Nanostructured NiO/Ni Induced by a Particle Size Reduction, Journal of Physical Chemistry C, 2012, Vol. 116, pp. 4356-4364, ISSN 1932-7447
7.	Antić B., Kremenović A., Vučinić-Vasić M., Dohcević-Mitrović Z., Nikoloć A., Gruden-Pavlović M., Jančar B., Meden A.: Composition related properties of (Yb,Y)(2)O-3 nanoparticles synthesized by controlled thermal degradation of AA complexes, Materials chemistry and physics, 2010, Vol. 122, No 2-3, pp. 386-391, ISSN 0254-0584
8.	Antić B., Rogan J., Kremenović A., Nikoloć A., Vučinić-Vasić M., Božanić D., Goya G., Colomban P.: Optimization of photoluminescence of Y2O3:Eu and Gd2O3:Eu phosphors synthesized by thermolysis of 2,4-pentanedione complexes, NANOTECHNOLOGY, 2010, Vol. 21, No 24, pp. 2457-2457, ISSN 0957-4484
9.	Jović N., Vučinić-Vasić M., Kremenović A., Antić B., Jovalekić Č., Vulić P., Kahlenberg V., Kaindl R.: HEBM synthesis of nanocrystalline LiZn0.5Ti1.5O4 spinel and thermally induced order-disorder phase transition (P4332-Fd3m), Materials chemistry and physics, 2009, No 2-3, pp. 542-549, ISSN 0254-0584
10.	Vučinić-Vasić M., Antić B., Blanuša J., Rakić S., Kremenović A., Nikolić A., Kapor A.: Formation of nanosize Li-ferrites from acetylacetonato complexes and their crystal structure, microstructure and order-disorder phase transition , Applied Physics A, 2006, Vol. 82, No 1, pp. 49-54, ISSN 0947-8396

Summary data for teacher's scientific or art and professional activity:

Quotation total :	53			
Total of SCI(SSCI) list papers :	17			
Current projects :	Domestic :	2	International :	1

Standard 10. Organizational and Material Resources

To perform the study programme, the adequate human, spatial, technical and technological, library and other resources suitable to the study programme features and predicted students` number are provided.
To perform the study programme, the adequate space for lecturing is provided, as well as the adequate laboratory space necessary for the experimental work and the equipment based on contemporary information and communication technologies. Lectures are held in amphitheatres, classrooms and specialized laboratories.
Faculty provides the usage of the library fund from its own or other sources (books, monographs, scientific magazines, other periodicals) in the amount necessary for the Doctoral study programme. Doctoral study students have the access to databases necessary for Doctoral dissertation elaboration and scientific and research work.
The library possesses more than 100 library units relevant for the performance of the study programme. All courses from the study programme have adequate textbooks, devices and supplementary equipment available on time and in a satisfactory number for the normal teaching process. There is also adequate information support.
Faculty has the library and the study room and provides a seat for each student in amphitheatres, classrooms and laboratories.
Faculty has a short-term and a long-term plan and the budget for the realization of scientific and research work.
Means for the realization of Doctoral studies, besides the ones provided by the resource ministries, are also provided in cooperation with other higher education institutions, accredited scientific institutions and international organizations.
Faculty provides students to utilize equipment or have access to necessary and adequate equipment in the possession of the Faculty, for scientific and research work.
Faculty provides students to utilize equipment or have access to the equipment necessary for scientific and research work on the basis of contracts on cooperation with other appropriate institutions.

Standard 11. Quality Control

Estimation of the study programme quality is elaborated regularly and systematically via self-evaluation and external quality control. One should place an emphasis on the multi-decade practice of students` surveys.
Study programme quality control is elaborated in the following manners:
- Surveying students at final lecture from the given course.
- Surveying students on the quality of the study programme and logistic support to the studies in the event of awarding the Diploma. Also, the studying comfort (classroom cleanness and tidiness) is evaluated there.
- Surveying students during the confirmation on completing a year of studies. Then students evaluate the logistic support to the studies.
- Surveying students on enrolling each year of studies. Then students evaluate the study programme at the year they completed in the prior academic year.
- Surveying the teaching and non-teaching staff on the quality of the study programme and the logistic support to the studies. This survey evaluates the work of the Dean`s office, Registrar`s office, library, and other services at the Faculty.

To monitor the quality of the study programme, there is a committee whose members are Doctoral Studies Council (composed of Faculty of Technical Science professors)two members of non-faculty stuff (administrative officer), together with a one student.
Additional quality is obtained by the obligatory scientific production of candidates. Prior to beginning the defence of the Doctoral dissertation, each candidate is obliged to publish at least one paper in the magazine from the SCI list.