11

NUMERICAL ANALYSIS

11.0. INTRODUCTION

Numerical analysis was evolved to develop methods for obtaining solutions of mathematical problems for which there exists no known methods or when the problem is not clearly defined. The development of numerical analysis is strongly influenced by the advent and use of digital computers, which is now an indispensable tool for scientists and engineers. This development includes the creation of new methods as well as modification of existing methods to make them more effective. Since the solution obtained by numerical methods are only approximate to the true solutions, the study of errors in the numerical solutions is important. In this chapter, numerical methods for approximate solutions to the problems commonly encountered in scientific and engineering applications are discussed. These methods include numerical integration, numerical solution of ordinary differential equations, matrix inversion, solution of linear simultaneous equations, and eigenvalues and eigenvectors.

11.1. APPROXIMATION

An approximate number may be defined as a number which is used as an approximation to an exact number and differs slightly from its exact value. Approximations are made in the process of calculating numerical results. Some of the reasons are:

(*a*) A computer has a finite word length and a fixed number of digits are stored and used during computation. Most stored values and the results of computation are only approximate to their true values.

(*b*) Numbers such as $\sqrt{2}$, $\sqrt{3}$, π, e, etc., have to be approximated as they cannot be expressed exactly by a finite number of digits. In digital forms, these numbers are:

$$\sqrt{2} = 1.4142, \qquad \sqrt{3} = 1.7320, \qquad \pi = 3.1416, \qquad e = 2.7183$$

These numbers, being only approximation to the true numbers, are approximate numbers.

(*c*) When an infinite process (like sum of an infinite series) is replaced by a finite process, an approximation occurs.

(*d*) The very process of physical measurement is approximate.

11.2. SIGNIFICANT FIGURES

The accuracy of an approximate number A is frequently measured by the number of significant digits in A. In decimal system, the digits 1, 2, 3, 4, 5, 6, 7, 8, 9 are significant figures and 0 is a significant figure except when it is used to fix up the position of the decimal point or to fill up the position of unknown digits. The formal rules for significant digits are :

1. A non-zero digit is always significant.
2. The digit 0 is significant if it lies between other significant digits.
3. The final 0s are all significant if the approximate number has an embedded decimal point.
4. The digit 0 is never significant when it precedes all the non-zero digits.

The most significant digit is the leftmost significant digit and the least significant digit is the rightmost significant digit.

Number	*Significant digits*	
2.78	2, 7, 8	by rule 1
0.00017	1, 7	by rules 1 and 4
0.1100	1, 1, 0, 0	by rules 1 and 3
0.0001110	1, 1, 1, 0	by rules 1, 3, and 4
770.020	7, 7, 0, 0, 2, 0	by rules 1, 2, and 3

Note : For the number $N = 123000$ has final three 0s but no embedded decimal point, there is no indication to show that zeros are significant or not. Such an ambiguity can be removed by writing numbers in exponential form (in powers of 10). The decimal point appears directly after the first non-zero digit. The number 123000 can be written as 1.23×10^5, 1.230×10^5, 1.2300×10^5, 1.23000×10^5. Every digit in the mantissa (left of 10^5) is significant.

Example 11.1. *Determine the number of significant digits, most significant and least significant digits for (a) 0.0234, (b) 6.005, (c) 22.33, (d) 0.004500.*

Solution. (*a*) 3 significant digits are 2, 3, 4 leftmost, 2 is the most significant rightmost, 4 is the least significant.

(*b*) 4 significant digits are 6, 0, 0, 5 leftmost, 6 is the most significant rightmost, 5 is the least significant.

(*c*) 4 significant digits are 2, 2, 3, 3 leftmost, 2 is the most significant rightmost, 3 is the least significant.

(*d*) 4 significant digits are 4, 5, 0, 0 most significant leftmost, first significant digit 4 and least significant rightmost, 0.

Example 11.2. *What are the number of significant figures in (a) 26.302, (b) 7.520, (c) the product 2.675×36.1?* **(AMIE, W '93)**

Solution. (*a*) There are 5 significant figures in 26.302 and they are 2, 6, 3, 0 and 2.

(*b*) The number of significant figures are 3 or 4 and the significant figures are 7, 5, 2 or 7, 5, 2, 0.

(*c*) $2.675 \times 36.1 = 96.5675$. There are 6 significant figures in the product and they are 9, 6, 5, 6, 7 and 5.

Note : When the term significant digit is used to describe results obtained from calculations, it is used to signify the number of digits in the answer (excluding leading zeros) which are known to be correct. For instance, if $x = 2.254$ is correct to 4 significant digits, we mean that

$$2.254 - .0005 \leq x \leq 2.254 + .0005$$

The subtraction of two nearly equal numbers causes a considerable loss of significant digits and this may greatly effect the result of calculation. Let $x = .127683 \times 10^2$ and $y = .127683 \times 10^2$, each correct to six significant digits. Then $x - y = .000245 \times 10^2$ which is correct to only 3 significant digits.

11.3. TYPES OF ERRORS

In numerical computation, we usually come across the following errors:

Absolute Error

The true value *minus* approximate value is called an error. True value = Approximate value + Error. If N be the exact quantity and $\overline{N}$ be the approximation, then

$$E \text{ (error)} = N - \overline{N}$$

The numerical difference between the true value and its approximate value is calle the absolute error.

$$\text{Absolute Error} = |N - \bar{N}|$$

Relative Error

This is the error measured relative to the true value. The ratio of the absolute error and true value of the quantity is called relative error. It often makes good sense when very large or very small numbers are used in calculations. In the common event that the true value is unknown, the approximations is sometimes substituted for it and the result still called somewhat loosely, the relative error.

$$\text{Relative error} = \frac{\text{Error}}{\text{True value}}$$

Percentage Error

The 100 times the relative error is called percentage error. Hence,

$$\text{Percentage error} = 100 \times \text{Relative Error}$$

$$= 100 \times \frac{|\text{ Error }|}{\text{True value}}$$

The relative error and percentage error are independent of the unit of measurements.

Truncation Error

Truncation errors are errors corresponding to the fact that a (finite or infinite) sequence of computational steps necessary to produce an exact result is "truncated" prematurely after a certain number of steps. Such error occurs when an infinite (like sum of an infinite series) or an infinitesimal process (like differentiation) is replaced by a finite process.

If we take the first n terms of the Taylor's series expansion of $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$ – ..... to approximate its value, then truncation error occurs.

If the value of $\frac{dv}{dx}$ is approximated by $\frac{f(x+h) - f(x)}{h}$, then truncation error occurs. This error is not machine caused but is due to the method.

Round-off Error

Such error occurs when we do not keep all the decimal placed in a number but retain only a finite number of them. To represent a number in finite form, we round them. The error we introduce in rounding a number is called round-off error.

Let x be a number having finite or an infinite decimal representation and let $x^{(d)}$ be an approximation to x. Then $x^{(d)}$ is correctly rounded to a decimal digit number when

$$|x - x^{(d)}| \leq \frac{1}{2} \times 10^{-d}$$

Rules for rounding-off numbers to K decimals are:

1. Discard the $(K + 1)$th and all subsequent decimals.
2. If the number thus discarded is less than half a unit in the Kth place, leave the Kth decimal unchanged.
3. If it is greater than half a unit in the Kth place, add one to the Kth decimal.
4. If it is exactly half a unit, round-off to the nearest even decimal (i.e., add 1 if it is an odd number).

Example 11.3. Round-off and truncate the following numbers correct to 4 significant digits: (i) 44.665 (ii) 61.553 (iii) 9.9950 (iv) 4.29473 (v) 0.0056789 (vi) 21234.56.

Solution.

Number	*Rounded number*	*Truncated number*
44.665	44.66	44.66
61.558	61.56	61.55
9.9950	9.995	9.995
4.29473	4.295	4.294
0.0056789	0.005679	0.005678
21234.56	2123	2123

Example 11.4. *If $P = 2.426$, find the absolute error, relative error. When (a) P is rounded to 2.43, (b) P is truncated to 2.42.*

Solution. (*a*) Absolute error $= |P - \overline{P}_r|$, where P_r is the rounded value of P.

$$= |2.426 - 2.43| = |-0.004| = 0.004$$

$$\text{Relative error} = \frac{|-0.004|}{2.426} = 1.6488 \times 10^{-3}$$

(*b*) Absolute error $= |P - P_T|$, where P_T is the truncated value of P.

$$= |2.426 - 2.421| = |0.006| = 0.006$$

$$\text{Relative error} = \frac{0.006}{2.426} = 2.473 \times 10^{-3}$$

Applications of Error Formulae to Fundamental Operations

(*a*) Addition

Let $N = N_1 + N_2 + ... + N_n$

Then $|\Delta N| \leq |\Delta N_1| + |\Delta N_2| + ...+ |\Delta N_n|$, where ΔN, ΔN_1, ΔN_2, ... Δ_n are errors in N, N_1, N_2,....., N_n, respectively. The absolute error of a sum of approximate number is almost equal to the algebraic sum of their absolute errors.

(*b*) Subtraction

Let $N = N_1 - N_2$.

Then $|\Delta N| \leq |\Delta N_1| + |\Delta N_2|$

The absolute error of the difference of two approximate numbers is almost equal to the sum of their absolute errors.

(*c*) Multiplication

Let $N = N_1 N_2 .....N_n$

Then $\left|\frac{\Delta N}{N}\right| \leq \left|\frac{\Delta N_1}{N_1}\right| + \left|\frac{\Delta N_2}{N_2}\right| + ... + \left|\frac{\Delta N_n}{N_n}\right|$

The relative error of a product of n approximate numbers is almost equal to the algebraic sum of the relative errors of the separate numbers.

(*d*) Division

Let $N = N_1/N_2$.

Then $\left|\frac{\Delta N}{N}\right| \leq \left|\frac{\Delta N_1}{N_1}\right| + \left|\frac{\Delta N_2}{N_2}\right|$

The relative error of a quotient of two numbers is almost equal to algebraic sum of the relative errors of divisor and dividend.

A simple formula for the absolute error of a quotient can be found as shown under:

Let ΔN be the absolute error of $\frac{n_1}{n_2}$. Then

$$\Delta N = \frac{n_1 + \Delta n_1}{n_2 + \Delta n_2} - \frac{n_1}{n_2}$$

$$= \frac{n_1 n_2 + n_2 \Delta n_1 - n_1 n_2 - n_1 \Delta n_2}{n_2 (n_2 + \Delta n_2)}$$

$$= \frac{n_2 \Delta n_1 - n_1 \Delta n_2}{n_2 (n_2 + \Delta n_2)} = \frac{n_1 \left(\frac{n_2 \Delta n_1 - n_1 \Delta n_2}{n_1 n_2} \right)}{n_2 + \Delta n_2}$$

$$= \frac{n_1 \left(\frac{\Delta n_1}{n_1} - \frac{\Delta n_2}{n_2} \right)}{n_2 + \Delta n_2}$$

Let ω be the greatest absolute value of $\frac{\Delta n_1}{n_1}$ or $\frac{\Delta n_2}{n_2}$. If n_1 and n_2 are both subject to errors of the *same order of magnitude*, then

$$\Delta N \leq \frac{n_1 (\omega + \omega)}{n_2 - \omega n_2}, \text{ i.e., } \Delta N \leq \frac{2 n_1 \omega}{n_2 (1 - \omega)}$$

If ω is negligible in comparison with 1,

$\Delta N \leq 2 \left(\frac{n_1}{n_2} \right) \omega$; if only n_1 and n_2 are both subject to error.

However, if only n_1 or n_2 is subject to error, then

$$\Delta N \leq \left(\frac{n_1}{n_2} \right) \omega$$

Some Important Rules to Remember

Let x be any number and the exponent of x is n, i.e., $x = 0.d_1 d_2 d_3 \ldots \times 10^n$, then

1. **Absolute error because of truncation**
   If x_a is the approximate value of x after truncating to k digits, then
   $$|x - x_a| \leq 10^{n-k}$$
2. **Absolute error because of rounding-off**
   If x_a is the approximate value of x after rounding-off to k digits, then
   $$|x - x_a| < 0.5 \times 10^{n-k}$$
3. **Relative error because of truncation**
   If x_a is the approximate value of x after truncating to k digits, then
   $$\left| \frac{x - x_a}{x} \right| < 10^{-k+1}$$
4. **Relative error because of rounding-off**
   If x_a is the approximate value of x after rounding-off to k digits, then
   $$\left| \frac{x - x_a}{x} \right| < 0.5 \times 10^{-k+1}$$

This type of analysis is useful if we want to keep track of the errors that occur when the numbers in a calculation are truncated or rounded-off to a certain number of decimal digits.

Relation between Relative Error and Significant Digits

The accuracy of a result is determined by the number of significant digits required to express it. The true index of accuracy of a computation is the relative error. In general, if we demand a solution accurate to n significant digits, we must ensure that relative

error is $\leq 10^{-n}$. Note that the absolute error in a numerical computation correct to n significant figures cannot be greater than half a unit in the nth place.

Theorem: If the first significant figure of a number is k and the number is correct to n significant figures, then the relative error is less than $\frac{1}{K \times 10^{n-1}}$.

Error in Functional Evaluation

Let $N = f(x_1, x_2, ..., x_n)$, where $x_1, x_2, ..., x_n$ are true values. Let y_i be the approximate values of x_i $(i = 1, 2, ..., n)$ and $N + \Delta N = f(y_1, y_2, ...y_n)$, where ΔN is an error in N. Moreover, we assume $\varepsilon_i = x_i - y_i$, where ε_i is the error in x_i. Now,

$$N + \Delta N = f(x_1 - \varepsilon_1, x_2 - \varepsilon_2, ..., x_n - \varepsilon_n) \quad ...(1)$$

Expanding right side of (1),

$$f(x_1, x_2, ..., x_n) - \sum_{i=1}^{n} \varepsilon_i \frac{\partial f_i}{\partial x_i} + \frac{1}{2!}\left[\sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} (\varepsilon_i)^2 + 2\varepsilon_1\varepsilon_2 \frac{\partial^2 f}{\partial x_1 \partial x_2} + ...\right] + ...$$

Since the errors $\varepsilon_1, \varepsilon_2, ..., \varepsilon_n$ are always relatively small, neglecting their squares, products and higher powers,

$$N + \Delta N = f(x_1, x_2, ..., x_n) - \sum_{i=1}^{n} \varepsilon_i \frac{\partial f_i}{\partial x_i}$$

$$= N - \sum_{i=1}^{n} \varepsilon_i \frac{\partial f_i}{\partial x_i}$$

Hence, $$\Delta N = -\sum \varepsilon_i \frac{\partial f_i}{\partial x_i} = -\left[\varepsilon_1 \frac{\partial f_1}{\partial x_1} + \varepsilon_2 \frac{\partial f_2}{\partial x_2} + ... + \varepsilon_n \frac{\partial f_n}{\partial x_n}\right]$$

Replacing $\varepsilon_i = \Delta x_i$,

$$\Delta N = -\left[\frac{\partial f_1}{\partial x_1} \Delta x_1 + \frac{\partial f_2}{\partial x_2} \Delta x_2 + ... + \frac{\partial f_n}{\partial x_n} \Delta x_n\right]$$

$$\frac{\Delta N}{N} = -\left[\frac{\partial N}{\partial x_1} \cdot \frac{\Delta x_1}{N} + \frac{\partial N}{\partial x_2} \cdot \frac{\Delta x_2}{N} + ... + \frac{\partial N}{\partial x_n} \cdot \frac{\Delta x_n}{N}\right]$$

$$\left|\frac{\Delta N}{N}\right| \leq \left|\frac{\partial N}{\partial x_1} \cdot \frac{\Delta x_1}{N}\right| + \left|\frac{\partial N}{\partial x_2} \cdot \frac{\Delta x_2}{N}\right| + ... + \left|\frac{\partial N}{\partial x_2} \cdot \frac{\Delta x_n}{N}\right|$$

In particular, if $N = K \frac{a^p b^q c^r}{d^l e^m}$, where $p, q, r, l, m > 0$.

Here $$N = f(p, q, r, l, m)$$

$$\frac{\partial N}{\partial P} = kp \frac{a^{p-1} b^q c^r}{d^l e^m}$$

or $$\frac{1}{N} \frac{\partial N}{\partial P} = \frac{p}{a}$$

Similarly, $$\frac{1}{N} \frac{\partial N}{\partial q} = \frac{q}{b}, \quad \frac{1}{N} \frac{\partial N}{\partial r} = \frac{r}{c}, \quad \frac{1}{N} \frac{\partial N}{\partial l} = -\frac{l}{d}, \quad \frac{1}{N} \frac{\partial N}{\partial m} = -\frac{m}{e}$$

Hence, $$\frac{\Delta N}{N} = p \cdot \frac{\Delta a}{a} + q \frac{\Delta b}{b} + r \frac{\Delta c}{c} - l \frac{\Delta d}{d} - m \frac{\Delta e}{e}$$

or $$\left|\frac{\Delta N}{N}\right| \leq p \left|\frac{\Delta a}{a}\right| + q \left|\frac{\Delta b}{b}\right| + r \left|\frac{\Delta c}{c}\right| + l \left|\frac{\Delta d}{d}\right| + m \left|\frac{\Delta e}{e}\right|$$

Example 11.5. *Evaluate* $D = \sqrt{5.01} - \sqrt{5}$ *correct to 3 significant digits.*

Solution. Given, $$D = \sqrt{5.01} - \sqrt{5} = \frac{(\sqrt{5.01} - \sqrt{5})(\sqrt{5.01} + \sqrt{5})}{\sqrt{5.01} + \sqrt{5}}$$

$$= \frac{0.01}{\sqrt{5.01} + \sqrt{5}} = \frac{0.01}{2.238 + 2.236} = \frac{0.01}{4.474} = 2.235 \times 10^{-3}$$

Example 11.6. *If* $u = \frac{5xy^2}{z^3}$ *and error in x, y, z be 0.001, 0.002 and 0.003, compute the relative error in n where* $x = y = z = 1$.

Solution. Here $u = f(x, y, z)$, where $f(x, y, z) = \frac{5xy^2}{z^3}$

$$\Delta u = \frac{\partial u}{\partial x}\Delta x + \frac{\partial u}{\partial y}\Delta y + \frac{\partial u}{\partial z}\Delta z$$

$$= \frac{5y^2}{z^3}\Delta x + \frac{10xy}{z^3}\Delta y - \frac{15xy^2}{z^4}\Delta z$$

or $$\frac{\Delta u}{u} = \frac{\Delta x}{x} + \frac{2\Delta y}{y} - \frac{3\Delta z}{z}$$

or $$\left|\frac{\Delta u}{u}\right| \le \left|\frac{\Delta x}{x}\right| + 2\left|\frac{\Delta y}{y}\right| + 3\left|\frac{\Delta z}{z}\right|$$

$$\le \frac{0.001}{1} + \frac{2 \times 0.002}{1} + \frac{3 \times 0.003}{1}$$

$$\le .014$$

Hence, the relative error in u is .014.

Example 11.7. *The numbers 48.392 and 6852.4 are both approximate and true only to their last digits. Find their difference and state how many figures in the result are trustworthy.*

[AMIE, S '93]

Solution. Let $u_1 = 48.392$ or $|\Delta u_1| \le .0005$

$u_2 = 6852.4$ or $|\Delta u_2| \le .05$

and $u = |u_1 - u_2|$

or $|\Delta u| \le |\Delta u_1| + |\Delta u_2|$

$\le .0005 + .05$

$\le .0010$

Again, $u = |48.392 - 6852.4| = 6804.008$

The true values of u lie between 6804.008 + .0010 and 6804.008 – .0010, i.e., 6804.0090 and 6804.007.

Hence, the first six figures are trustworthy.

Example 11.8. *Find the product 349.1 × 863.4 and state how many figures of the result are trustworthy (assuming that each number is correct to four figures).*

Solution. Let $u_1 = 349.1$ or $|\Delta u_1| \le .05$

$u_2 = 863.4$ or $|\Delta u_2| \le .05$

and $u = u_1 \times u_2$

or $$\left|\frac{\Delta u}{u}\right| \le \left|\frac{\Delta u_1}{u_1}\right| + \left|\frac{\Delta u_2}{u_2}\right|$$

$$\le \left|\frac{.05}{u_1}\right| + \left|\frac{.05}{u_2}\right|$$

$$\leq .05\left[\frac{1}{|u_1|} + \frac{1}{|u_2|}\right]$$

or $$|\Delta u| \leq .05\left[\left|\frac{u}{u_1}\right| + \left|\frac{u}{u_2}\right|\right]$$

$$\leq .05\left[\frac{|u|(|u_1| + |u_2|)}{|u_1|\,|u_2|}\right]$$

$$\leq .05\,[|u_1| + |u_2|]$$

$$\leq .05\,[349.1 + 863.4]$$

$$\leq 60.62$$

The true values of u lie between 301412.94 + 60.62 and 301412.94 − 60.62,i.e., 301473.56 and 301352.32, or 3015×10^2 and 3014×10^2.

Hence, only the first three figures are reliable.

Example 11.9. *Given the solution of a problem as $x = 45.15$ with relative error in the solution being almost 2%. Find to 4 decimal digits the range of values within which the exact value of the solution must lie.*

Solution. Given maximum relative error = 0.02. Hence,

$$-0.02 < \frac{x - x_a}{x} < 0.02, \text{ where } x_a \text{ denotes the approximate in } x.$$

If $$\frac{x - x_a}{x} < 0.02$$

then $$x - x_a < 0.02\,x \qquad (\text{since } x > 0)$$

or $$x(1 - 0.02) < x_a$$

or $$x < \frac{x_a}{1 - 0.02} = \frac{45.15}{0.98} = 46.071429$$

If $$-0.02 < \frac{x - x_a}{x}$$

then $$x - x_a > -0.02x \qquad (\text{since } x > 0)$$

or $$x(1 + 0.02) < x_a$$

or $$x > \frac{x_a}{1 + 0.02} = \frac{45.15}{1.02} = 44.264706$$

Hence, $44.2647 < x < 46.0714$

Example 11.10. *Find $(x^2 - y^2)/(x + y)$ for $x = 0.4845$, $y = 0.4800$ correct up to 6 decimal places and compare the result with $(x - y)$.*

Solution. Let $$z_1 = \frac{x^2 - y^2}{x + y}$$

$$= \frac{(0.4845)^2 - (0.4800)^2}{0.4845 + 0.4800}$$

$$= \frac{0.234740 - 0.230400}{0.964500} = 0.004499 \qquad \text{(taking up to 6 digits)}$$

Again, let $z_2 = x - y = 0.4845 - 0.4800 = 0.0045$

Hence, the results by two methods are 0.004499 and 0.0045.

Example 11.11. *Find the number of trustworthy figures in the quotient of $\frac{568.5}{289.3}$, assuming that both numbers are approximate and true only to the number of digits given.*

Solution. Let $n_1 = 568.5$ and $n_2 = 289.3$, then $n = \frac{n_1}{n_2}$. Then

$$\frac{n_1}{n_2} = \frac{568.5}{289.3} = 1.9651$$

Let ω = greatest absolute of $\frac{\Delta n_1}{n_1}$ or $\frac{\Delta n_2}{n_2}$

$$= \frac{.05}{289.3} = .0001728$$

and $\Delta N \leq 2\left(\frac{n_1}{n_2}\right)\omega$

$$\leq 2 \times 1.9651 \times .0001728$$

$$\leq .0006791.$$

Since the error affects the fourth decimal place only, the quotient is correct to three decimal places, and that $N = 1.965$.

REVIEW QUESTIONS SET

1. If $u = 4x^2y^3/z^3$ and error in x, y, z be 0.001, 0.001 and 0.002, compute the relative error in u when $x = y = z = 1$.
2. If $y = 4x^6 - 5x$, find the percentage error in y at $x = 1$, if the error in y is 0.04.
3. Round-off the following numbers to four significant figures: (*i*) 0.77029, (*ii*) 68.243, (*iii*) 0.0022218, (*iv*) 101101.
4. Find the product 45.2 and 4.57 (figures obtained by rounding off) and find the likely error.
5. Given $A = 66.888$, find the absolute and the relative error if (*a*) A is rounded to 66.89, (*b*) A is truncated to 66.88.
6. Find the number of significant figures in the following: (*i*) 1.0236 (*ii*) .002406 (*iii*) 371000.
7. Compute relative error in computing $u = x^3 + 2x^2 - x$ for $x = \sqrt{3}$, taking $\sqrt{3} \simeq 1.73$.
8. If $u = \frac{4x^2y}{z^2}$ and error in x, y, z be .001, .002 and .0015, find the maximum possible error in computation if $x = 2$, $y = 4$ and $z = 3$. Find also the relative error.
9. Find the relative error in computation of $x - y$ for $x = 9.05$ and $y = 6.56$ and have absolute error $\Delta x = 0.001$ and $\Delta y = 0.003$, respectively.
10. Define the following: (*a*) Absolute error, (*b*) Relative error, (*c*) Round-off error, (*d*) Inherent error.

(AMIE, W '97)

ANSWERS TO REVIEW QUESTIONS SET

1. .004 2. 76% 3. (*a*) 0.7703 (*b*) 68.24 (*c*) 0.002222 (*d*) 1011 4. $\pm$ 0.474
5. (*a*) $|-0.002|$, $|-0.00299|$%, (*b*) 0.008, 0.01196% 6. (5, 4, 3 or 4 or 5 or 6)
7. .0016 8. 0.0185, 0.00173 9. 0.00034

11.4. SOLUTIONS OF ALGEBRAIC AND TRANSCENDENTAL EQUATIONS

In this section, some mathematical techniques are discussed to find approximate solutions. Since a computer can perform numerical computations very fast, the study of numerical methods become more important. Sometimes a rough approximation of a root can be found by graph.

The equations of the form $f_n(x) = 0$ are called algebraic or transcendental if $f_n(x)$ is a polynomial in x or contains some other functions such as trigonometric, logarithmic, and exponential functions. For example, the equations $x^7 + 9x^6 + 7x^3 - 5 = 0$ and $6x^2 + \log(x + 4) + e^{-x}\cos x = 0$ are called algebraic and transcendental, respectively.

Synthetic Division by Linear Factors

Consider the division of $f_n(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n$ by $x - \alpha$. Suppose

$Q(x)$ be the quotient and R the remainder, then

$$\frac{a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}\,x + a_n}{x - \alpha} = Q(x) + \frac{R}{x - \alpha}$$

or $$a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n = (x - \alpha)\,(b_0\,x^{n-1} + b_1\,x^{n-2} + \ldots + b_{n-1})\,R$$

where $Q(x) = b_0x^{n-1} + b_1x^{n-2} + \ldots + b_{n-1}$.

Comparing coefficients of like powers of x,

$a_0 = b_0$	$b_0 = a_0$
$a_1 = b_1 - b_0\alpha$	$b_1 = a_1 + b_0\alpha$
$a_2 = b_2 - b_1\alpha$	$b_2 = a_2 + b_1\,\alpha$
... ...	... ...
... ...	... ...
$a_{n-1} = b_{n-1} - b_{n-2}\,a\,\alpha$	$b_{n-1} = a_{n-1} + b_{n-2}\,\alpha$
$a_n = R - b_{n-1}\,\alpha$	$R = a_n + b_{n-1}\,\alpha$

In tabular form,

α	a_0	a_1	a_2	...	...	...	a_{n-2}	a_{n-1}	a_n
	b_0	$b_0\alpha$	$b_1\alpha$	...	...	...	$b_{n-3}\alpha$	$b_{n-2}\alpha$	$b_{n-1}\alpha$
	$b_0 \equiv c_0$	b_1	b_2	...	...	...	b_{n-2}	b_{n-1}	$R = f_n(\alpha)$

This is known as *Homer's scheme*. Here every term in the second row is obtained by multiplying the preceding term in the third row by a, whereas each trerm of third row is obtained by the addition of the corresponding term of first and second row. Also, remainder

$$\begin{aligned} R &= a_n + b_{n-1}\,\alpha = a_n + (a_{n-1} + b_{n-2}\,\alpha)\,\alpha \\ &= a_n + a_{n-1}\,\alpha + (a_{n-2} + b_{n-3}\,\alpha)\alpha_2 \\ &= a_n + a_{n-1}\,\alpha + a_{n-2}\,\alpha^2 + \ldots + a_1\,\alpha^{n-1} + a_0\,\alpha^n = f_n\,(\alpha) \end{aligned}$$

If the divisor is $x + \alpha$, then $-\alpha$ can be written for α and the procedure is carried out as earlier.

Suppose the algebraic equation of nth degree be

$$a_0\,x^n + a_1\,x^{n-1} + \ldots + a_{n-1}\,x + a_n = 0, \quad a_0 \neq 0 \qquad \ldots(1)$$

This equation has n roots, some of which may be real and different or real and repeated and the rest imaginary (in pairs). One is concerned with only the real roots which are obtained by the method of synthetic division. Some general theorems concerning with the real roots of algebraic equations are given below.

Theorem 1. Descarte's rule of signs: In any algebraic equation $f_n\,(x) = 0$, *(a)* the number of positive roots cannot exceed the number of change of signs (or variations) from positive to negative and from negative to positive in $f_n\,(x)$, and *(b)* the number of negative roots cannot exceed the number of variations in $f_n\,(-x)$.

Theorem 2. If two real quantities a and b be substituted for x in any polynomial $f_x\,(x)$, $f_n\,(a)$ *and* $f_n\,(b)$ are of opposite sign, then at least one or an odd number of real roots of the equation $f_n\,(x) = 0$ lies between a and b. If $f_n\,(a)$ and $f_n\,(b)$ are of same sign, then either on real root or an even number of roots of $f_n\,(x) = 0$ lies between a and b.

11.4.1 Graphical Method

To solve the equation $f\,(x) = 0$, a graph is drawn for the function $y = f(x)$ with respect to

x and y-axis, and obtain x-coordinates of those points for which y-co-ordinates are zero. These x-coordinates will then determine the roots of the equation $f(x) = 0$. In other words, the roots may be interpreted as x-coordinate of the points of intersection of the curve $y = f(x)$ with x-axis. In case $f(x)$ involves differences of two functions, then $f(x) = 0$ can be written as $f_1(x) = f_2(x)$, where $f_1(x)$ and $f_2(x)$ are functions of x. To solve this equation, draw graphs for two functions $y = f_1(x)$ and $y = f_2(x)$. Then x-coordinates of the points of intersection of two curves will determine the roots of the equation $f(x) = 0$.

Stepwise procedure. Following steps may be followed to solve the equations graphically:

Step 1: Find the smallest interval (a, b) between which the root of the equation lies $f(x) = 0$.

Step 2: Write the equation $f(x) = 0$ as $f_1(x) = f_2(x)$

Step 3: Prepare two tables for $y = f_1(x)$ and $y = f_2(x)$ taking values of x between a and b.

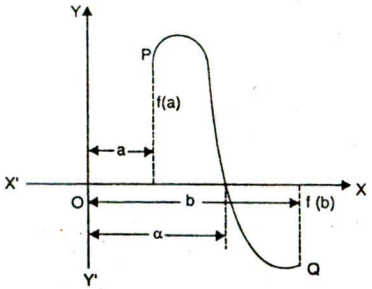
Step 4: Plot these values and join them for a smooth curve.

Step 5: Find the value of point of intersection of the curves $y = f_1(x)$ and $y = f_2(x)$. This is the required root of the equation $f(x) = 0$.

11.4.2 Bisection Method

The method of bisection is that which is always convergent but the convergence is very slow. This method is also known as half-interval method or Bolzano Bisection method. This method is based on the following theorem:

Theorem: If a function $f(x)$ is continuous in the closed interval $[a.b]$ and $f(a).f(b)$ are of opposite signs, then there is at least one number α in the open interval $]a.b[$ such that $f(\alpha) = 0$



Proof: Let $f(x) = 0$ at the points where graph of the function crosses the *x-axis*. Given $f(a)$ and $f(b)$ are of opposite signs at the end points P and Q.

Therefore, $f(x)$ changes its sign from positive to negative or vice-versa in changing from P to Q. Thus, the graph crosses x-axis at least at one between P and Q.

$\therefore$ $x = \alpha$ in the interval $[a.\ b]$

Hence, the theorem states that if $f(x) = 0$ is a polynomial equation and one has two numbers a and b, where $f(a)$ and $f(b)$ are of opposite signs, then there is at least one root of the equation between a and b.

Working rule : After having observed that the root lies in the interval $[a.\ b]$, the interval is bisected containing the root, by following these steps:

Step 1. Find $f(a)$ and $f(b)$ and check that $f(a)$, $f(b)$ are of opposite signs. This step is

known as initialisation.

Step 2. First iteration

Let $x_1 = \frac{a + x_1}{2}$. Now, we have to find $f(x_1)$

(*i*) If $f(x) = 0$, then a root lies in (a, x_1) if $f(a)$ and $f(x_1)$ are of opposite sign. Suppose $x_2 = \frac{a + x_1}{2}$ be the second approximation of the root and repeat the process in further steps till a root approximated to desired level of accuracy is obtained.

(*ii*) If $f(x) \neq 0$, then a root lies in (x_1, b) if $f(x_1)$ and $f(b)$ are of opposite sign.

Here, $x_2 = \frac{x_1 + b}{2}$ be the second approximation of the root and repeat the process in further steps till a root approximated to desired level of accuracy is obtained.

The various steps in the bisection method are called first iteration, second iteration and so on.

11.4.3 Regula Falsi Method (Method of False Position)

This method is applied to locate the range in which a root lies.

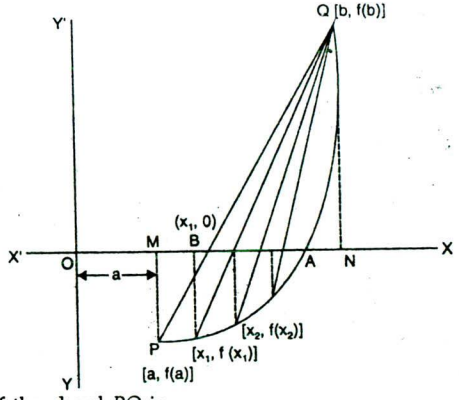
Let $f(x) = 0$ and $y = f(x)$ be presented by the curve PQ.

The curve PQ cuts the x-axis at A.

The real root of $f(x) = 0$ is OA. The false position of the curve PQ is taken as the chord PQ.

The chord PQ cuts the x-axis at B. The approximate root of $f(x) = 0$ is OB, *i.e.*, $OB = x_1$, then B $\equiv (x_1, 0)$. By the method of false position, we find OB.

Let P and Q be the extremities of the chord PQ having point $[a.\ f(a)]$, $[b.\ f(b)]$, respectively.



The equation of the chord PQ is

$$y - f(a) = \frac{f(b) - f(a)}{b - a}(x - a)$$

$$\left[\because \text{two point form, i.e., } (x_1, y_1) \text{ and } (x_2, y_2 = y - y_1) = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)\right]$$

Since PQ cuts x-axis at $B\ (x_1, 0)$, therefore

$$0 - f(a) = \frac{f(b) - f(a)}{b - a}(x_1 - a)$$

or $$(x_1 - a) = -\frac{b - a}{f(b) - f(a)} f(a)$$

or $$x_1 = a - \frac{b - a}{f(b) - f(a)} f(a)$$

Here, $$x_1 = \frac{af(b) - bf(a)}{f(b) - f(b)}$$

This value of x_1 gives the first approximation of the root. One can go on repeating this procedure and get a sequence of approximation till the desired level of accuracy is obtained.

Remember : Let $f(x_1) = 0$, x_1 is a root of $f(x) = 0$. Then

(*i*) if $f(a)$ and $f(x_1)$ are of opposite sign, the root of $f(x) = 0$ will lie in (a, x_1).

(*ii*) if $f(x_1)$ and $f(b)$ are of opposite sign, the root of $f(x) = 0$ will lie in (x_1, b).

11.4.4 Iteration Method

This method is applicable to the equations of the form $x = f(x)$ and is sometimes known as 'method of successive approximations'. Thus, an equation

$$F(x) = 0 \qquad ...(1)$$

can be solved by this method if it can be solved for the unknown x in terms of a function of x, i.e., it can be written in the form $x = f(x)$.

Suppose x_0 be an initial approximation to the solution of (1), then the next approximation x_1 is given by

$$x_1 = f(x_0) \qquad ...(2)$$

Similarly, treating x_1 as the initial value, the next approximation x_2 is given by

$$x_2 = f(x_1)$$

Proceeding in this manner, the nth approximation is given by

$$x_n = f(x_{n-1})$$

In this way, a sequence of successive approximations $x_2, x_3, \ldots$ are obtained, which may converge to the desired root.

In order to obtain the root of $F(x) = 0$, i.e., $x = f(x)$, it is required to find the abscissa of the point of intersection of the line $y = x$ with the curve $y = f(x)$, which may or may not intersect. In case two curves do not intersect, then no real root exists of the equation $F(x) = 0$. However, assume that the two curves intersect, and $f(x)$ and its derivative $f'(x)$ are continuous in the neighbourhood of the real root.

The method is conditionally convergent. The condition is that $|f'(x)| < 1$ in the neighbourhood of the real root $x = a$.

Suppose $x_0, x_1, x_2, \ldots$ be the successive iterations and $0 < f'(x) < 1$. Then it is obvious that y-coordinate of A is $f(x_0)$.

The horizontal line AB intersects the line $y = x$ at $B \equiv (x_1, x_1)$. Further, x-coordinate of B is the same as the x-coordinate of C, implying y-coordinate of $C = f(x_1) = x_2$, i.e., coordinates of C are (x_1, x_2). Proceeding in this manner, it can be proved that $x_1, x_2, \ldots, x_n$ converges to $x = a$.

Again, sequence $x_1, x_2, \ldots, x_n$ converges to $x = a$, whereas the sequence $x_1, x_2, \ldots, x_n$ diverges at other points.

Theorem. Assume $x = a$ be a root of $f(x) = 0$. Let I be an interval containing the point $x = a$, and $f(x)$ and $f'(x)$ be continuous in I, where equation $F(x) = 0$ is equivalently written as $x = x(x)$. Then if $|f'(x)| < 1$, $\forall\ x \in I$, the sequence of approximations $x_0, x_1, x_2, \ldots, x_n$ defined by $x_{n+1} = f(x_n)$ converges to $x = a$ provided x_0 is chosen in I.

11.4.5. Newton-Raphson Method (Successive Substitution Method)

Of all the methods available for the numerical solution of algebraic and transcendental equations, the Newton-Raphson method seems to be most satisfactory and easy for computer programming.

Suppose x_0 be an approximate value of a root of the equation $f(x) = 0$, and $x_0 + h$ be the exact value of the corresponding root, where h is a small quantity. Then

$$f(x_0 + h) = 0 \qquad \ldots(1)$$

Expanding (1) by Taylor's series,

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \ldots = 0$$

Since h is small, neglecting second and higher order terms of h, an approximate value of h is obtained as

$$f(x_0) + hf'(x_0) = 0$$

or
$$h = -\frac{f(x_0)}{f'(x_0)}, \text{ provided } f'(x_0) \neq 0 \qquad \ldots(2)$$

From (2), a value of h is obtained, which gives, when added to x_0, a better value of the root. If this improved value of the root is x_1, then

$$x_1 = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)} \qquad \ldots(3)$$

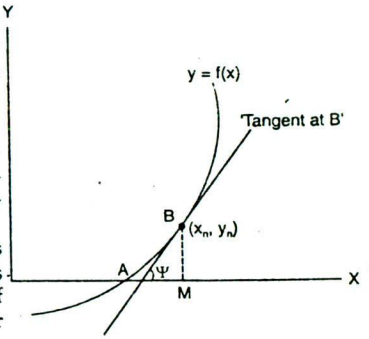
Now, substituting x_1 for x_0 and x_2 for x_1 in (3),

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_0)} \qquad \ldots(4)$$

In general, if $h_3, h_4, \ldots, h_n$ are the corrections to be applied to $x_2, x_3, \ldots, x_{n-1}$, respectively to be the better approximations of the root, then an iteration formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \ldots(5)$$

is obtained for calculating successive better values of the root. The formula (5) is known as Newton-Rephson method.



From above, it is inferred that this method is applicable only when h is small or $f'(x)$ is large. This method is thus useful in case of large values of $f'(x)$, that is, when the graph of the function near the root is nearly verticle.

This method can be used for both algebraic and transcendental equations and is also valie for complex coefcients or roots of an equation.

11.4.6. Newton's Iterative Formula for Finding square Root, Inverse and Inverse Square Root

(*a*) *Square root*: The square root of b can be considered as a root of the equation

$x^2 - b = 0$, solvable by Newton's method. Since $f9x) = x^2 - b, f'(x) = 2x$. Hence, the formula reduces to the form

$$x_{n+1} = x_n - \frac{x_n^2 - b}{2x_n}$$

or
$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{b}{x_n}\right) \quad ...(1)$$

(*b*) *Inverse*: The reciprocal or inverse of a number b can be considered as a root of the equation $\frac{1}{x} - b = 0$, which can be solved by Newton's method? Since $f(x) = x^{-1} - b$, $f'(x) = -x^{-2}$. Therefore, Newton's formula gives

$$x_{n+1} = x_n - \frac{x_n^{-1} - b}{-x_n^{-2}} = x_n(2 - bx_n) \quad ...(2)$$

(*c*) *Inverse square root*: The inverse square root of b is the root of the equation $\frac{1}{x^2} - b = 0$. Since $f(x) = x^{-2} - b, f'(x) = -2x^{-3}$. Then Newton's formula reduces to the form

$$x_{n+1} = x_n - \frac{x_n^{-2} - b}{-2x_n^{-3}} = \frac{1}{2}x_n(3 - bx_n^2) \quad ...(3)$$

(*d*) *General formula for pth root*: Suppose $f(x) = x^p - b$. Then

$$x_{n+1} = x_n - \frac{x_n p - b}{px_n^{p-1}} = \frac{(p-1)x_n^b + b}{px_n^{p-1}} \quad ...(4)$$

In order to obtain the reciprocal pth root of b, assume $f(x) = b - x^{-p}$. Then $f(x) = px^{-p-1}$. Therefore,

$$x_{n+1} = x_n \frac{(p + 1 - bx_n^b)}{p}.$$

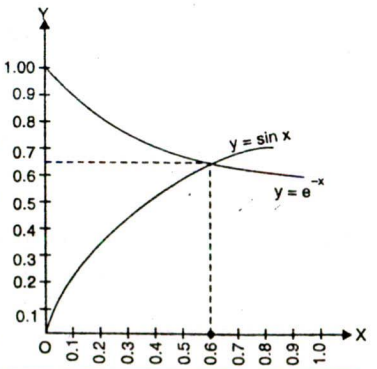
SOLVED EXAMPLES

Example 12. *Find the approximate value of the smallest real root of* $e^{-x} - \sin x = 0$.

Solution. From the graphs drawn for $y = e^{-x}$ and $y = \sin x$ on the axes as shown in the figure, it can be seen that the intersection of two curves is at the neighbourhood of $x = 6$.

This gives the approximate value of the smallest real root of the equation

$e^{-x} = \sin x$



Example 13. *Solve graphically*

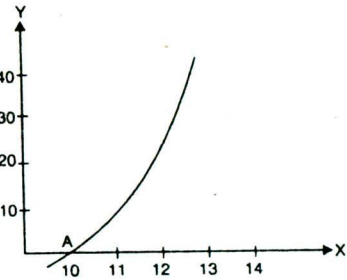
$e^{3x} - 5x^2 - 17 = 0$

Solution. Suppose $f(x) = e^{3x} - 5x^2 - 17 = 0$. Then

x	e^{3x}	x^2	$5x^2 + 17$	$y = f(x)$
1	20.085	1	22	–1.915
1.2	27.113	1.21	23.05	4.063
1.2	36.598	1.44	24.20	12.398
1.3	49.402	1.69	25.45	23.952

1.4	66.686	1.96	26.80	39.886
1.5	90.017	2.25	28.25	61.767

A graph on the basis of these values is shown in the figure. Here $f(1)$ is negative and $f(0.5)$ is positive. Here, a root of $f(x) = 0$ lies between 1 and 1.5.



The abscissa of the point of intersection of the curve with x-axis gives the solution of the equation $x = 1.04$.

Example 14. *Draw the graph of* $y = e^{x-1}$ *and find graphically the value of the root of equation* $3 - x = e^{x-1}$.

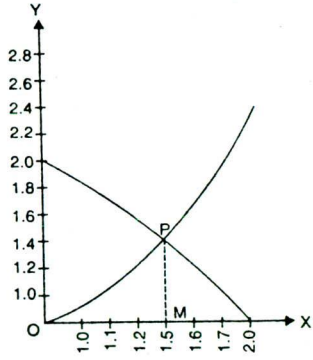
Solution. Suppose $f(x) = e^{x-1} + x - 3 = 0$. Then

$f(1) = 1 + 1 - 3 = -1 < 0$

$f(2) = e + 2 - 3 = 2.718 - 1 = 1.718 > 0$

Therefore, a root of $f(x) = 0$ lies between 1 and 2. The values of e^{x-1} may be read from the table as below:

x	e^{x-1}
1.0	1
1.1	1.1052
1.2	1.2214
1.3	1.3499
1.4	1.4918
1.5	1.6487
1.6	1.8221
1.7	2.0138
1.8	2.2255
1.9	2.4596
2.0	2.7183



The abscissa of the point of intersection of two graphs is 1.44. This is the required solution.

Example 15. *Find the real root of the equation* $x \log_{10} x - 1.2 = 0$. *Correct to five decimal places, using the method of false position.*

Solution. If $f(x) = x \log_{10} x - 1.2$, Then

$$y_1 = f(2) = 2 \log_{10} 2 - 1.2$$
$$= 2 \times 0.30103 - 1.2 = -0.59794$$
$$y_2 = f(3) = 3 \log_{10} 3 - 1.2 = 3 \times 0.47712 - 1.2 = 0.23136$$

which shows that there lies at least one real root of the equation $f(x) = 0$ in the interval $x = 2$ to $x = 3$ and it is nearer to 3.

Hence, using the formula, the first approximation of the root is given by

$$x^{(1)} = x_1 + \frac{(x_2 - x_1)\, y_1}{y_1 - y_2} = 2 + \frac{(3-2)(-0.59794)}{(-0.59794) - (0.23136)} = 2 + 0.72100 = 2.721$$

Now, $f(2.721) = 2.721 \times \log_{10}(2.721) - 1.2$

$$= 2.721 \times (0.43743) - 1.2 = 1.1829 - 1.2 = -0.0171$$

Since $f(2.721)$ and $f(3)$ are of opposite sign, the root lies between $x = 2.721$ and $x = 3.00$. The next approximate value of the root is given by

$$x^{(2)} = 2.721 + \frac{(3 - 2.721)(-0.0171)}{-0.0171 - 0.23136} = 2.721 + 0.019201 = 2.74021$$

Now, $f(2.74021) = 2.74021 \times \log_{10} 2.74021 - 1.2$

$$= 2.74021 \times (0.43778) - 1.2 = 1.1996 - 1.2 = -0.0004$$

Again, since $f(2.74021)$ and $f(3)$ are of opposite sign, there lies at least one root in the interval $x + 2.7402$ and $x = 3$. Also, the root is nearer to $x = 2.7402$. Therefore, the third approximation of the root is

$$x^{(3)} = 2.7402 + \frac{(3 - 2.7402)(-0.0004)}{-0.0004 + 0.23136}$$

$$= 2.7402 + \frac{(0.2598)(0.0004)}{0.23176} = 2.7402 + 0.00044844 = 2.74064844$$

Thus, 2.74064 is the required value of the root.

Example 16. *Find an approximate value of the root of equation $x^3 + x - 1 = 0$ near $x - 1$, using the method of false position.*

Solution. Given

Then

$$f(x) = x^3 + x - 1 = 0.$$

$$f(0.5) = (0.5)^3 + 0.5 - 1 = -0.375$$

$$f(1) = 1 + 1 - 1 = 1.0$$

Therefore, the root lies between 0.5 and 1. Suppose $x_1 = 0.5$ and $x_2 = 1$. Then

$$x^{(1)} = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)} = \frac{0.5(1) - 1(-0.375)}{1 + 0.375}$$

$$= \frac{7}{11} = 0.64$$

Also, $f(0.64) = -0.0979$ and $f(1) = 1$. Then the root lies between 0.64 and 1. Now, $x_1 = 0.64$ and $x_2 = 1$. Therefore,

$$x^{(2)} = \frac{0.64(1) - (-0.0979)}{1 - (-0.0979)} = 0.672$$

Also, $f(0.672) = -0.0245$ and $f(1) = 1$.

Again, $x_1 = 0.672$ and $x_2 = 1$. Then

$$x^{(3)} = \frac{0.672 + 0.0245}{1 + 0.0245} = 0.68$$

Now, $f(0.68) = -0.0069$, $x_1 = 0.68$ and $x_2 = 1$. Then

$$x^{(4)} = \frac{0.68 + 0.0069}{1 + 0.0069} = 0.6822$$

Hence, the required solution of the equation is $x = 0.6822$.

Example 17. *Find a real root of the following equation by false position method (maximum 3 intervals).* *(AMIE W'99)*

$$f(x) = x^3 - 2x - 5 = 0$$

Solution. Given $f(x) = x^3 - 2x - 5 = 0$. Then

$$f(2) = 2^3 - 2.2 - 5 = -1 \text{ and } f(3) = 3^3 - 2.3 - 5 = 16$$

Hence a real root lies between 2 and 3.

Suppose $x_0 = 2, x_1 = 3$. Hence, by false position method,

Iteration 1

$$x_2 = x_0 - \frac{f(x_0)}{f(x_1) - f(x_0)}(x_1 - x_0)$$

$$= 2 - \frac{f(2)}{f(3) - f(2)}(3 - 2) = 2 + \frac{1}{17} = 2.059$$

Now, $f(x_2) = f(2.059) = -0.386$. Hence, the root lies between 2.059 and 3.0.

Iteration 2

$$x_0 = 2.059 \quad f(x_0) = -0.386$$

$$x_1 = 3 \quad f(x_1) = 16$$

$$x_3 = 2.059 - \frac{(0.386)}{16 - (0.386)}(3 - 2.059)$$

$$= 2.059 + \frac{0.386}{16 + 0.386} \times 0.941 = 2.059 + 0.221668 = 2.0812$$

Iteration 3

Now, $f(x_3) = f(2.0812) = (2.0812)^3 - 2 \times 2.0812 - 5 = -0.1479$

Hence, the root lies between 2.0812 and 3.

$$x_0 = 2.0812, \quad x_1 = 3$$

$$f(x_0) = -0.1479, \quad f(x_1) = 16$$

$$x_4 = 2.0812 - \frac{(-0.1479)}{16 + 0.1479}(3 - 2.0812) = 2.0812 + \frac{0.1479 \times 0.9188}{16.1479}$$

$$= 2.0812 + \frac{0.13589}{16.1479} = 2.0896$$

Example 18. *Find a real root of the equation* $f(x) = x^3 + x^2 - 1 = 0$, *using iteration method.*

Solution. Here $f(0) = -1$ and $f(1) = 1$. Therefore, the root lies between 0 and 1. The given function is $f(x) = x^2(x + 1) - 1$ or $x^2 = \frac{1}{x+1}$ or $x = \frac{1}{\sqrt{x+1}}$. Therefore, $f(x) = \frac{1}{\sqrt{x+1}}$.

Now, $$f'(x) = -\frac{1}{2(x+1)^{3/2}}$$

and $$|f'(x)| = \left|\frac{1}{2(x+1)^{3/2}}\right| < 1, \quad \text{for } x < 1$$

$$x_1 = f(x_0) = \frac{1}{\sqrt{1.5}} = 0.81649$$

$$x_2 = f(x_1) = \frac{1}{\sqrt{1.81649}} = 0.74196$$

$$x_3 = f(x_2) = \frac{1}{\sqrt{1.74196}} = 0.75767$$

$$x_4 = f(x_3) = \frac{1}{1.75767} = 0.75427$$

$$x_5 = f(x_4) = \frac{1}{\sqrt{1.75427}} = 0.75500$$

$$x_6 = f(x_5) = \frac{1}{\sqrt{1.75500}} = 0.75485$$

$$x_7 = f(x_6) = \frac{1}{1.75485} = 0.75488$$

$$x_8 = f(x_7) = \frac{1}{1.75488} = 0.75487, \text{ and so on.}$$

One can take $x = 0.7548$ correct up to four places of decimal.

Example 19. *Find a real root of the equation* $\cos x = 3x - 1$ *correct up to four decimal places by iteration method.*

Solution. Let $f(x) = \cos x - 3x + 1 = 0$

$$f(0) = 2 = +ve;\ f\left(\frac{\pi}{2}\right) = 1 - 3\left(\frac{\pi}{2}\right) = -ve$$

Therefore, a root lies between 0 and $\pi/2$. The given equation may be written as

$$x = \frac{1}{3}(1 + \cos x) = \phi(x)$$

$$\phi''(x) = -\frac{1}{3}\sin x$$

$|\phi'(x)| = \left|\frac{1}{3}\sin x\right| < 1$ for all x and, in particular, in $(0, \pi/2)$, *i.e.* (0, 1.5708).

Hence, the iteration method may be applied.

Suppose $x_0 = 0.6$. Then

$$x_1 = \frac{1}{3}[1 + \cos(0.6)] = 0.60845$$

$$x_2 = \frac{1}{3}[1 + \cos(0.60845)] = 0.60684$$

$$x_3 = \frac{1}{3}[1 + \cos(0.60684] = 0.60715$$

$$x_4 = \frac{1}{3}[1 + \cos(0.60715)] = 0.60709$$

$$x_5 = \frac{1}{3}[1 + \cos(0.60709)] = 0.60710$$

$$x_6 = \frac{1}{3}[1 + \cos(0.60710)] = 0.60710$$

Due to repetition of the value of x_5 and x_6 no further calculation is required. Hence, the root is 0.6071 and correct up to four decimal places.

Example 20. *Find the root of* $x^2 - 5x + 2 = 0$ *correct to five decimal places by Newton-Raphson method.*

Solution. The The given equation has two real roots and approximate values of the roots lie between 0 and 4. Since

$$f(x) = x^2 - 5x + 2$$

$$f'(x) = 2x - 5$$

By Newton-Raphson formula,

$$x_{n+1} = x_n - \frac{x_n^2 - 5x_n + 2}{2x_n - 5} = \frac{x_n^2 - 2}{2x_n - 5}$$

Substitute $n = 0$, $x_1 = 0.4$. Again,

$$x_2 = \frac{(0.4)^2 - 2}{2(0.4) - 5} = \frac{-1.84}{-4.20} = 0.438095$$

$$x_3 = \frac{(0.438095)^2 - 2}{2(0.438095) - 5} = 0.438447$$

$$x_4 = \frac{(0.438447)^2 - 2}{2(0.438447)} = 0.438447$$

Since $x_3 = x_4$, real root is 0.438447. For the second real root, starting with $x_0 = 4$, one obtains,

$$x_1 = \frac{(4)^2 - 2}{2(4) - 5} = \frac{14}{3}$$

$$x_2 = \frac{(14/3)^2 - 2}{2(14/3) - 5} = 4.564104$$

$$x_3 = \frac{(4.564104)^2 - 2}{2(4.564104) - 5} = 4.561554$$

$$x_4 = \frac{(4.561554)^2 - 2)}{2(4.561554) - 5} = 4.561553$$

$$x_5 = \frac{(4.561553)^2 - 2}{2(4.561553) - 5} = 4.561553$$

Since $x_4 = x_5$, the real root of the given equation is 4.561553.

Example 21. *Use Newton Raphson's method to find a root of the equation* $x^3 - 3x - 5 = 0$ *correct up to three places of decimal. Assume initial guess to be* $x_0 = 3$. *(AMIE S'99)*

Solution. Initial guess $x_0 = 3$. Then

$$f(x) = x^3 - 3x - 5 \text{ and } f'(x) = 3x^2 - 3$$

From Newton Raphson's method, one obtains.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^3 - 3x_n - 5}{3x_n^2 - 3}$$

$$x_1 = x_0 - \frac{x_0^3 - 3x_0 - 5}{3x_0^2 - 3} = 3 - \frac{3^3 - 3 \times 3 - 5}{3 \times 3^2 - 3} = 3 - \frac{13}{24} = 2.458$$

$$x_2 = 2.458 - \frac{(2.458)^3 - 3 \times 2.458 - 5}{3 \times (2.458)^2 - 3} = 2.2943$$

$$x_3 = 2.2943 - \frac{(2.2943)^3 - 3 \times 2.2943 - 5}{3 \times (2943)^2 - 3} = 2.2791$$

$$x_4 = 2.2791 - \frac{(2.2791)^3 - 3 \times 2.2791 - 5}{3 \times (2.2791)^2 - 3} = 2.2790$$

Hence, the real root correct up to three decimal places is 2.279.

Example 22. Solve the equation $x - 1 - \sin x = 0$ *graphically.*

Solution. Let $f(x) = x - 1 - \sin x$ ...(1)

The equation (1) can be written in the form

$f_1(x) = \sin x$ ...(2)

$f_2(x) = x - 1$ ...(3)

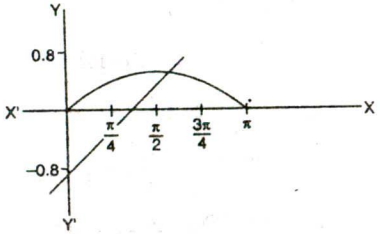
Now, draw two curves for $y_1 = \sin x$ and $y_2 = x - 1$.

$y_1 = \sin x$

x	0	$\pi/4$	$\pi/2$	$3\pi/4$	π
$y_1 = \sin x$	0	0.71	1.00	0.71	0

$y_2 = x - 1$

x	0	1	$3\pi/4$
$y_2 = x - 1$	-1	0	1.4



From the graph, one gets $wx = 1.95$ radians approximately.

***Example* 23.** *Use Bolzana bisection method for the equation* $x^3 - 5x - 4 = 0$ *(use bisection three times).*

Solution. Let $f(x) = x^3 - 5x - 4 = 0$ has a root between

$x = 2$, *i.e.*, $f(2) = 2^3 - 10 - 4 = 8 - 10 = -6$

$x = 3$, *i.e.*, $f(3) = 3^3 - 15 - 4 = 27 - 19 = 8$

Let us bisect the range (2, 3) by

$x_1 = (2 + 3)/2 = 2.5$, $f(x_1) = -0.875$

Hence, the root now lies in the range (2.5, 3).

Again, bisect the range,

$$x_2 = \frac{2.5 + 3}{2} = 2.75,\ f(x_2) = 3.0468$$

Hence, the root lies in the interval (2.5, 2.75).

Again, after bisection,

$$x_3 = \frac{2.5 + 2.75}{2} = 2.5625,\ f(x_3) = 0.9625$$

The root lies in the region (2.5, 2.5625) and this procedure is continued for further refinement.

***Example* 24.** *Use Bisection method to solve the equation* $x^3 - 9x + 1 = 0$ *for the root between 2 and 4.*

Solution. Given $f(x) = x^3 - 9x + 1$

$f(2) = 2^3 - 9 \times 2 + 1 = -9$

$$f(4) = 4^3 - 36 + 1 = 29$$

Since $f(2)$ and $f(4)$ are of opposite sign, the roots lie between 2 and 4, bisect the range (2, 4).

$$x_1 = \frac{2+4}{2} = 3,\ f(x_1) = 1\ (+ve)$$

Hence, the root lies between 2 and 3. Again bisecting the range,

$$x_2 = \frac{2+3}{2} = 2.5,\ f(x_2) = -5.875$$

Here the root lies between (2.5, 3). Again bisecting the range

$$x_3 = \frac{2.5+3}{2} = 2.75,\ f(x_3) = -2.9531$$

Now, the root lies between *2.75,3). Again, bisecting the range.

$$x_4 = \frac{2.75+3}{2} = 2.875$$

$$f(x_4) = -1.113$$

Since $f(3)$ and $f(2.9375)$ are of the opposite sign, the root lies between (2.875, 3).

$$x_4 = \frac{2.875+3}{2} = 2.9375$$

$$f(2.9375) = -0.090089$$

Since $f(3)$ and $f(2.875)$ are of the opposite sign, the root lies in (2.9375, 3). Similarly, highter iterations can be calculated till the root to the desired level of accuracy is obtained.

Example 25. *Find a root of equation $x^3 + x - 1 = 0$ near $x = 1$ using the method of false position (regula falsi) two times.*

Solution. Given $f(x) = x^3 + x - 1 = 0$

$$f(1) = 1 + 1 - 1 = +1$$

$$f(0.5) = (0.5)^3 + (0.5 - 1 = -0.375$$

The root lies between 0.5 and 1. Then

$$a = 0.5,\ b = 1$$

$$f(a) = -0.375,\ f(b) = 1$$

$$x_1 = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

$$x_1 = \frac{0.5(1) - 1(-0.375)}{1 + 0.375} = 0.64,\ f(0.64) = -0.0979$$

Since $f(0.64)$ and $f(1)$ are of opposite sign, the root lies in (0.64, 1). Now,

$$a = 0.64,\ b = 1$$

$$f(0.64) = -0.0979,\ f(1) = 1$$

$$x_2 = \frac{0.64(1) - 1(-0.0979)}{1 + 0.0979} = 0.672$$

$$f(0.672) = -0.0245$$

Since $f(0.672)$ and $f(1)$ are of opposite sign, the root lies in (0.672, 1). Then

$$a = 0.672 \text{ and } b = 1$$

$$x_3 = \frac{0.672(1) - 1(-0.0245)}{1 + 0.0245}$$

or

$$x_3 = 0.6822$$

Example 26. *Find by method of regula falsi a root of the equation*

$$x^3 + x^2 - 3x - 3 = 0$$

lying between 1 and 2.

Solution. Given $f(x) = x^3 + x^2 - 3x - 3 = 0$

$$f(1) = 1 + 1 - 3 - 3 = -4$$

$$f(2) = 8 + 4 - 6 - 3 = 3$$

Since $f(1)$ and $f(2)$ are of opposite sign, a root lies between 1 and 2. Then

$$x_1 = \frac{af(b) - bf(a)}{f(b) - f(a)} = \frac{1f(2) - 2f(1)}{f(2) - f(1)} = \frac{1 \times 3 - 2 \times (-4)}{3 - (-4)} = \frac{11}{7} = 1.571$$

$$f(1.571) = (1.571)^3 + (1.571)^2 - 3(1.571) - 3$$

$$= 3.877 + 2.468 - 4.713 - 3 = 1.368$$

Since $f(1.571)$ and $f(1)$ are of opposite sign, a root lies between 1.571 and 2.

$$x_2 = \frac{1.571\,(f(2) - 2f(1.571)}{f(2) - f(1.571)} = \frac{1.571 \times 3 - 2 \times (-1.368)}{3 - (-1.368)}$$

$$= \frac{4.713 + 2.736}{4.368} = 1.705$$

$$f(1.705) = (1.705)^3 + (1.705)^2 + 3\,(1.705) - 3$$

$$= 4.960 + 2.908 - 5.115 - 3 = -0.247$$

Since $f(1.705)$ and $f(2)$ are of opposite sign, a root lies between 1.705 and 2.

$$x_3 = \frac{(1.705 f(2) - 2f(1.705)}{f(2) - f(1.705)} = \frac{1.705 \times 3 - 2\,(-0.247)}{3 - (-0.247)}$$

or $$x_3 = 1.727$$

REVIEW QUESTION SET

1. Find the approximate value of the root of equation $x - 1 = \sin x$
2. Find graphically the positive root of the equation $x^3 - 6x - 13 = 0$
3. Draw a graph for $y = x^3$ and $y + 2x = 0$ and find the approximate solution of the equation $x^3 + 2x - 20 = 0$.
4. Draw a graph for the function $y = x \tan x$ from $x = 0$ to $x = \pi/2$ and hence obtain approximately a solution of the equation $x \tan x = 1$.
5. Find graphically a value of x lying between 15 and 20 and satisfying the equation $\frac{x^2}{100} + 25 \log_{10} x - 6.35 = 0$.
6. Find the real root of the equation $xe^x - 3 = 0$ by the false position method, correct to three decimal places.
7. Find the real root of $x^3 + x^2 + 3x + 4 = 0$ correct to four decimal places by the Regula Falsi method.
8. The equation $x^6 - x^4 - x^3 - 1 = 0$ has one real root between 1.4 and 1.5. Find this root correct to four decimal places by the false position method.
9. Apply the false position method to solve the equation $3x - \cos x - 1 = 0$.
10. Find the root of the equation $x^3 - 5x - 7 = 0$, which lies between 2 and 3 by the method of false position.
11. Evaluate (*a*) $\sqrt{29}$ (*b*) $\sqrt{30}$ (*c*) cube root of 10, and (*d*) $\left(\frac{1}{3}\right)^{1/4}$
12. Using iterative method, compute the real root of the equation $x^3 - x - 0.2 = 0$ correct up to three decimal places.
13. Find the real root of the following equations by the iteration method, correct to two decimal places:

(*a*) $x^3 + x - 1 = 0$
(*b*) $x^3 - 5x - 11 = 0$
(*c*) $x^3 + 10x - 15 = 0$

14. Apply Newton's method twice to find the real root near $x = 0.9$ of the equation $3 \tan x = 3x + 1$, x being defined in radians.
15. Find the position root of each of the following equations:
(*a*) $x^4 + x^3 - 7x^2 - x + 5 = 0$
(*b*) $x^4 + x^3 - 2x^2 - 6x - 8 = 0$
16. Use Newton-Raphson method to find the root of the equation $x^3 - 10x^2 + 40x - 35 = 0$, which lies between 1 and 2 correct to three decimal places.
17. Find the positive of the equation $x^2 - 78.8 = 0$ correct up to four decimal places, using the Newton-Raphson method. . Assume that the initial approximation to the root is 14.0.
18. Using Newton-Raphson method, find the root of $x^4 - x - 10 = 0$, which is nearer to $x = 2$, correct to three decimal places.
19. Solve $x^4 - 5x^3 + 20x^2 - 40x + 60 = 0$ by Newton-Raphson method, given that all the roots of the equation are complex.
20. Determine the real root of $2x - 3 \sin x - 5 = 0$ up to six decimal places, using Newton-Raphson method.
21. Find the smallest positive root of $x^3 - 5x + 3 = 0$, using the Newton-Raphson method.
22. Find the root(s) of the following equations correct up to four decimal places, using Newton-Raphson method: (*a*) $\log x = \cos x$, (*b*) $x^2 - 5x + 2 = 0$.
23. Find the real root of the following equations correct up to four decimal places, using Newton-Raphson method: (*a*) $4 \sin x + x^2 = 0$, (*b*) $3x - \cos x - 1 = 0$.
24. Find a root of $y = \frac{1}{x} + \sin x$ near $x = 1.5$.
25. Apply Newton-Raphosn method to the equation $1 - \frac{a}{x^2} = 0$ to give the following recurrence relation for $\sqrt{a}$:
$x_{n+1} = x_n \left[1 - \frac{x_n^2 - a}{2a}\right]$

ANSWER TO REVIEW QUESTION SET

1. 1.9 **2.** 3.176 **3.** 2.47 **4.** 0.864 **5.** 17.933 **6.** 1.060
7. – 1.3582 **8.** 1.4036 **9.** 2.746 **11.** (*a*) 5.385165, (*b*) 5.47722, (*c*) 2.1566
12. 1.045 **13.** (*a*) 0.61, (*b*) 2.95, (*c*) 1.30 **14.** 0.8835849 **15.** (*a*) 2.0609, (*b*) 2.0669192
16. 1.174 **17.** 8.8769 **18.** 1.856 **19.** $1.915 \pm 1.908i$; $0.585 \pm 2.805i$
20. 2.883238 **21.** 1.834 **22.** (*a*) 1.30296, (*b*) 0.4384, (*c*) 4.5615
23. (*a*) – 1.9338, (*b*) – 0.6071 **24.** 1.4973

11.4. NUMERICAL INTEGRATION

A problem which often arises in applied mathematics is that a definite integral does not have a closed-form solution. Every function cannot be integrated analytically but a numerical approximation can be obtained for any continuous function. Numerical integration is a primary tool used by engineers and scientists to obtain approximate answers for definite integrals.

Numerical integration is the process of finding or evaluating a definite integral $I = \int_a^b f(x)\, dx$ from a set of numerical values of the integrand $f(x)$. If it is applied to the integration of a function of a single variable, the process is known as *quadrature*. The problem of numerical integration is solved by first approximating the integrand by polynomial with the help of an interpolation formula and then integrating the expression

between the desired limits. In order to evaluate the definite integral $\int_a^b f(x)\,dx$, *first express the function* $f(x)$ by an interpolation formula, say $p(x)$, and then integrate between the limits a and b, that is,

$$\int_a^b f(x)\,dx \sim \int_a^b p(x)\,dx$$

The error E in such type of approximation is given by

$$\int_a^b f(x)\,dx - \int_a^b p(x)\,dx = \int_a^b [f(x) - p(x)]\,dx$$

11.5.1. A General Quadrature Formula for Equidistant Ordinates

Suppose $I = \int_a^b y\,dx$, where $y = f(x)$ and $f(x)$ be given for certain equally distant values of arguments, say, x_0, $x_0 + h$, $x_0 + 2h$, ... Let the range $(b - a)$ be divided into n equal parts, each of which is of width h, that is, $b - a = nh$. If $x_0 = a$, $x_1 = x_0 + h = a + h$, $x_2 = a + 2h$, ..., $x_n = a + nh = b$ and $(n + 1)$ ordinates y_0, y_1, ..., y_n are at equal intervals, then

$$I = \int_a^b f(x)\,dx = \int_{x_0}^{x_0 + nh} f(x)\,dx = h\int_0^n f(u)\,du$$

where $u = \dfrac{x - x_0}{h}$ and $dx = h\,du$.

Replacing $f(u)$ by Newton's forward interpolation formula,

$$I = h\int_0^n \left[y_0 + u\,\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 + \ldots + \frac{u(u-1)\ldots(u-n+1)}{n!}\Delta^n y_0\right] du$$

$$= h\left[n y_0 + \frac{n^2}{2}\Delta y_0 + \left(\frac{n^3}{3} - \frac{n^2}{2}\right)\frac{\Delta^2 y_0}{2!} + \left(\frac{n^4}{4} - n^3 + n^2\right)\frac{\Delta^3 y_0}{3!} + \ldots.\right.$$

$$\left.\text{up to } (n+1) \text{ terms}\right] \qquad \ldots(1)$$

This is the general quadrature formula.

Trapezoidal Rule

If $n = 1$ in (1) and neglect second and higher differences, the interval of integration will be from x_0 to $x_0 + h$, and there are only two functional values y_0 and y_1 in this interval. There can be no differences higher than the first. Thus,

$$\int_{x_0}^{x_0+h} y\,dx = h\left[y_0 + \frac{1}{2}\Delta y_0\right] = h\left[y_0 + \frac{y_1 - y_0}{2}\right] = h\left[\frac{y_0 + y_1}{2}\right]$$

Similarly, $\displaystyle\int_{x_0+h}^{x_0+2h} y\,dx = h\left(\frac{y_1 + y_2}{2}\right)$

... ...

$$\int_{x_0+(n-1)h}^{x_0+nh} y\,dx = h\left(\frac{y_{n-1} + y_n}{2}\right)$$

Adding these n integrals,

$$\int_{x_0}^{x_0+nh} y\,dx = h\left[\frac{1}{2}(y_0 + y_n) + (y_1 + y_2 + \ldots + y_{n-1})\right]$$

= distance between two consecutive ordinates × [mean of the first and last ordinates + sum of all intermediate ordinates]

This is called the Trapezoidal rule.

Except for sufficiently small values of h, the formula gives crude result. The formula replaces the curve by n straight lines joining the points (x_0, y_0) and (x_1, y_1); (x_1, y_1) and (x_2, y_2), ... , (x_{n-1}, y_{n-1}) and (x_n, y_n). The area bounded by the curve $y = f(x)$, the ordinates $x = x_0$ and $x = x_n$ and the x-axis is then approximately equivalent to the sum of the areas of the trapeziums obtained (Fig. 11.1).

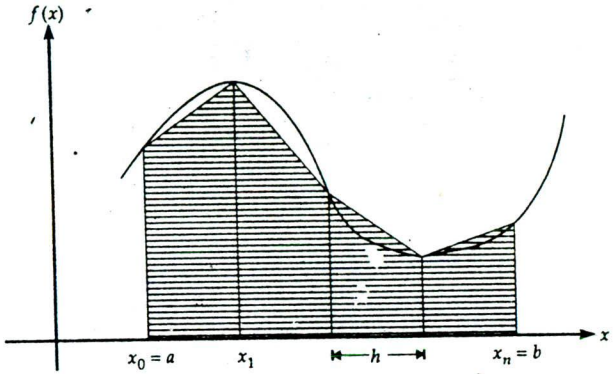


Fig. 11.1 : Trapezoidal rule : Approximation of $\int_a^b f(x)\,dx$

Simpson's One-third Rule

If $n = 2$ in (1) and neglect third and higher differences,

$$\int_{x_0}^{x_0+2h} y\,dx = h[2y_0 + 2(y_1 - y_0) + \frac{(8/3 - 2)}{2}(y_2 - 2y_1 + y_0)]$$

$$= \frac{h}{3}(y_0 + 4y_1 + y_2)$$

Similarly, $\int_{x_0+2h}^{x_0+4h} y\,dx = \frac{h}{3}(y_2 + 4y_3 + y_4)$

... ...

$$\int_{x_0+(n-2)h}^{x_0+nh} y\,dx = \frac{h}{3}(y_{n-2} + 4y_{n-1} + y_n) \qquad (n \text{ is even})$$

Adding all these integrals,

$$\int_{x_0}^{x_0+nh} y\,dx = \frac{h}{3}[(y_0 + y_n) + 4(y_1 + y_2 + \ldots + y_{n-1}) + 2(y_2 + y_4 + \ldots + y_{n-2})]$$

This formula is known as Simpson's one-third rule.

One can get better approximation to the area under curve by a series of parabolas each with a vertical axis and passing through three points of the curve that are equally spaced in the direction of x. To use this formula, the interval of integration must be divided into an even number of sub-intervals (Fig. 11.2).

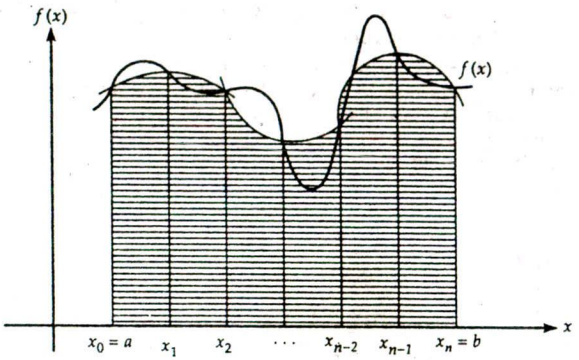


Fig. 11.2 : Simpson's rule : Approximation of $\int_a^b f(x)\,dx$

Simpson's Three-eighth Rule

If $n = 3$ in (1) and neglect fourth and higher differences,

$$\int_{x_0}^{x_0+3h} y\,dx = h\left[3y_0 + \frac{9}{2}(y_1 - y_0) + \frac{9}{4}(y_2 - 2y_1 + y_0) + \frac{3}{8}(y_3 - 3y_2 + 3y_1 - y_0)\right]$$

$$= \frac{3h}{8}[y_0 + 3y_1 + 3y_2 + y_3]$$

Similarly, $\int_{x_0+3h}^{x_0+6h} y\,dx = \frac{3h}{8}[y_2 + 3y_4 + 3y_5 + y_6]$

... ...

$$\int_{x_0+(n-3)h}^{x_0+nh} y\,dx = \frac{3h}{8}[y_{n-3} + 3y_{n-2} + 3y_{n-1} + y_n]$$

where n is multiple of 3.

Adding all these integrals,

$$\int_{x_0}^{x_0+nh} y\,dx = \frac{3h}{8}\left[(y_0 + y_n) + 3(y_1 + y_2 + y_4 + y_5 + \ldots + y_{n-1}) + 2(y_3 + y_6 + \ldots + y_{n-3})\right]$$

This formula is known as Simpson's three-eighth rule.

Weddle's Rule

If $n = 6$ in (1) and neglect seventh and higher differences,

$$\int_{x_0}^{x_0+6h} y\,dx = h\left[6y_0 + 18\Delta y_0 + 27\Delta^2 y_0 + 24\Delta^3 y_0 + \frac{123}{10}\Delta^4 y_0 + \frac{33}{10}\Delta^5 y_0 + \frac{41}{140}\Delta^6 y_0\right]$$

Now, replace $\frac{41}{140}\Delta^6 y_0$ by $\frac{3}{10}\Delta^6 y_0$, since the error involved is negligible, the expression takes the form

$$\int_{x_0}^{x_0+6h} y\,dx = h\left[6y_0 + 18\Delta y_0 + 27\Delta^2 y_0 + 24\Delta^3 y_0 + \frac{123}{10}\Delta^4 y_0 + \frac{33}{10}\Delta^5 y_0 + \frac{3}{10}\Delta^6 y_0\right]$$

$$= \frac{3h}{10}\left[y_0 + 5\,y_1 + y_2 + 6y_3 + y_4 + 5y_5 + y_6\right]$$

Similarly,

$$\int_{x_0+6h}^{x_0+12h} y\,dx = \frac{3h}{10}\left[y_6 + 5y_7 + y_8 + 6y_9 + y_{10} + 5y_{11} + y_{12}\right]$$

....

$$\int_{x_0+(n-6)h}^{x_0+nh} y\,dx = \frac{3h}{10}\left[y_{n-6} + 5\,y_{n-5} + y_{n-4} + 6y_{n-3} + y_{n-2} + 5y_{n-1} + y_n\right]$$

If n is multiple of 6, adding all these integrals,

$$\int_{x_0}^{x_0+nh} y\,dx = \frac{3h}{10}\left[y_0 + 5y_1 + y_2 + 6y_3 + y_4 + 5y_5 + 2y_6 + 5y_7 + y_8 + \ldots.\right]$$

This formula is known as Weddle's rule, and is more accurate than the Simpson's rule. However, it requires at least seven consecutive values of the function. Also, this method is applicable only when n is a multiple of six.

11.5.2. Errors of Quadrature Formulae

Error in Trapezoidal Rule

Let $y = f(x)$ be continuous and possess continuous derivatives in (x_0, x_n). Expanding y in a Taylor's series around $x = x_0$, we obtain

$$\int_{x_0}^{x_1} y\,dx = \int_{x_0}^{x_1}\left[y_0 + (x - x_0)\,y_0' + \frac{(x - x_0)^2}{2!}\,y_0'' + \ldots.\right]dx$$

$$= hy_0 + \frac{h^2}{2}\,y_0' + \frac{h^3}{6}\,y_0'' + \ldots. \qquad \text{(where } x_1 - x_0 = h\text{)} \qquad \ldots(1)$$

Again, $\frac{h}{2}(y_0 + y_1) = \frac{h}{2}\left[y_0 + y\,(x_0 + h)\right]$

$$= \frac{h}{2}\left[y_0 + y_0 + hy_0' + \frac{h^2}{2!}\,y_0'' + \frac{h^3}{3!}\,y_0''' + \ldots\right] \qquad \ldots(2)$$

Expressions (1) and (2)

$$\int_{x_0}^{x_1} y\,dx - \frac{h}{2}(y_0 + y_1) = -\frac{1}{12}\,h^3 y_0'' + \ldots.$$

In case derivatives of higher order than two are neglected, $E_1 = -\frac{1}{12}\,h^3 y_0''$ is the error term in the interval (x_0, x_1) and is known as *local error*. Proceeding in a similar manner, we obtain errors in the remaining sub-intervals, viz., (x_1, x_2), (x_2, x_3), ..., (x_{n-1}, x_n). Considering the whole interval (x_0, x_n), the total or global error is given by

$$E = -\frac{h^3}{12}\left[y_0'' + y_1'' + \ldots.. + y''_{n-1}\right] \qquad \ldots(3)$$

Assuming that $y''(\xi)$ is the largest value of n quantities on the right side of (3),

$$E = -\frac{h^3}{12} n y''(\xi), \; x_0 < \xi < x_n$$

$$= -\frac{(b-a)}{12} h^2 y''(\xi), \; nh = b - a$$

Error in Simpson's One-third Rule **(AMIE, S '94, S '96)**

Expanding y in Taylor's series around $x = x_0$, we obtain

$$\int_{x_0}^{x_2} y\,dx = \int_{x_0}^{x_2} \left[y_0 + (x - x_0) y_0' + \frac{(x - x_0)^2}{2!} y_0'' + \frac{(x - x_0)^3}{3!} y_0''' + \frac{(x - x_0)^4}{4!} y_0^{iv} + \ldots.\right]$$

$$= 2hy_0 + \frac{4h^2}{2!} y_0' + \frac{8h^3}{3!} y_0'' + \frac{16h^4}{4!} y_0''' + \frac{32h^5}{5!} y_0^{iv} + \ldots..$$

(where $x_2 - x_0 = h$) ...(1)

Again,

$$\frac{h}{3}[y_0 + 4y_1 + y_2] = \frac{h}{3}[y_0 + 4y(x_0 + h) + y(x_0 + 2h)]$$

$$= \frac{h}{3}\left[y_0 + \left(4y_0 + 4hy_0' + \frac{4h^2}{2!} y_0'' + \frac{4h^3}{3!} y_0''' + \frac{4h^4}{4!} y_0^{iv} + \ldots..\right)\right.$$

$$\left. + \left(y_0 + 2hy_0 + \frac{4h^2}{2!} y_0'' + \frac{8h^3}{3!} y_0''' + \frac{16h^4}{4!} y_0^{iv} + \ldots.\right)\right] \quad ...(2)$$

Expressions (1) and (2) give

$$\int_{x_0}^{x_2} y\,dx - \frac{h}{3}[y_0 + 4y_1 + y_2] = -\frac{1}{90} h^5 y_0^{iv} + \ldots.$$

Neglecting the derivatives of higher order than four, the local error in (x_0, x_2),

$$E_1 = -\frac{1}{90} h^5 y_0^{iv}$$

The global error $E = -\frac{1}{90 \times 2} nh^5 y''(\xi), \; x_0 < \xi < x_n$

$$= -\frac{(b-a)}{180} h^4 y''(\xi) \qquad \text{(since } nh = b - a\text{)}$$

Error in Simpson's Three-eighth Rule

The global error $E = -\frac{nh^5}{80} y^{iv}(\xi), \; x_0 < \xi < x_n$

Error in Weddle's Rule

The global error $E = -\frac{nh^7}{840} f^{vi}(\xi), \; x_0 < \xi < x_n$

Example 11.27. *Calculate* $\int_2^{10} \frac{dx}{1+x}$ *by dividing the range into eight equal parts.*

(AMIE, S '96, W '97)

Solution. Divide the given range of integration (2, 10) into eight equal parts each of width $h = 1$, and compute the value of $y = \frac{1}{1+x}$ at each point of sub-division. Computed values are as follows:

x :	2	3	4	5	6	7	8	9	$10 = x_0 + 8h$
$y_n = \frac{1}{1+x}$:	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{7}$	$\frac{1}{8}$	$\frac{1}{9}$	$\frac{1}{10}$	$\frac{1}{11}$

or

y_n :	0.333	0.250	0.200	0.167	0.143	0.125	0.111	0.100	0.091

Now, by Simpson's one-third rule,

$$\int_2^{10} \frac{dx}{1+x} = \int_{x_0}^{x_0+8h} y\,dx = \frac{h}{3}[y_0 + y_8 + 4(y_1 + y_3 + y_5 + y_7) + 2(y_2 + y_4 + y_6)]$$

$$= \frac{1}{3}[0.333 + 0.091 + 4(0.250 + 0.167 + 0.125 + 0.100) + 2(0.200 + 0.143 + 0.111)]$$

$$= 1.233$$

Example 11.28. *Calculate the value of* $\int_{-3}^{3} x^4\,dx$ *by Simpson's one-third rule (use seven equidistant ordinates). Compare the result with its exact value.* **(AMIE, S '94)**

Solution. Divide the given range (–3, 3) into 6 equal parts each of width $h = 1$. The values of y computed at each point of sub-division are as follows:

x :	–3	–2	–1	0	1	2	$3 = x_0 + 6h$
$y_n = x^4$:	$(-3)^4$	$(-2)^4$	$(-1)^4$	0	$(1)^4$	$(2)^4$	$(3)^4$

or

y_n :	81	16	1	0	1	16	81

By Simpson's rule,

$$\int_{-3}^{3} x^4\,dx = \frac{h}{3}[y_0 + y_6 + 4(y_1 + y_3 + y_5) + 2(y_2 + y_4)]$$

$$= \frac{1}{3}[81 + 81 + 4(16 + 0 + 16) + 2(1 + 1)]$$

$$= 98$$

Again, exact value is

$$\int_{-3}^{3} x^4\,dx = \left|\frac{1}{5}x^5\right|_{-3}^{3} = \frac{3^5}{3} + \frac{3^5}{3} = 97.2$$

There is an error of – 0.8 in Simpson's result.

Example 11.29. *Evaluate the following integral by Simpson's one-third rule:*

$$I = \int_0^4 \sin x\,dx. \qquad \textit{Take } h = 0.5.$$

Find out the relative error between actual value of the integral and the value obtained by Simpson's rul. **(AMIE, W '95)**

Solution. Dividing the given range (0, 4) into eight equal parts each of width $\mu = 0.5$ and compute the values of $y = \sin x$ at each point of sub-division.

x :	0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
$y_n = \sin x$:	0	0.4794	0.8415	0.9975	0.9093	0.5985	0.1411	–0.3508	–0.7568

By Simpson's one-third rule,

$$\int_0^4 \sin x\,dx = \int_{x_0}^{x_0+8h} y\,dx = \frac{h}{3}[y_0 + y_8 + 4(y_1 + y_3 + y_5 + y_7) + 2(y_2 + y_4 + y_6)]$$

$$= \frac{0.5}{3}[0 - 0.7568 + 4(0.4794 + 0.9975 + 0.5985 - 0.3508) + 2(0.8415 + 0.9093 + 0.1411)]$$

$$= 1.6542$$

By actual integration, we get

$$\int_0^4 \sin x\, dx = [-\cos x]^4 = -\cos 4 + \cos 0 = 1.6536$$

$$\text{Relative error} = \frac{I_{\text{simpson}} - I_{\text{actual}}}{I_{\text{actual}}} = \frac{1.6542 - 1.6536}{1.6536} = \frac{00.0006}{1.6536} = 0.00036$$

Example 11.30. *Compute by any method the value of the integral* $\int_0^{\pi/3} \sqrt{\cos\theta}\, d\theta$.

(AMIE, S '93)

Solution. Dividing the given range $(0, \pi/3)$ into four equal parts each of width $\pi/12$ and the values of $y = \sqrt{\cos\theta}$ computed at each point of sub-intervals are as follows:

x	:	0	$\pi/12$	$\pi/6$	$\pi/4$	$\pi/3$
$y = \sqrt{\cos\theta}$	:	1.000	0.983	0.931	0.841	0.707

By Simpson's one-third rule,

$$\int_0^{\pi/3} \sqrt{\cos\theta}\, d\theta = \int_{x_0}^{x_0+4h} y\, dx = \frac{h}{3}[y_0 + y_4 + 4(y_1 + y_3) + 2y_2]$$

$$= \frac{\pi/12}{3}[1.000 + 0.707 + 4(0.983 + 0.841) + 2 \times 0.931] = 0.948$$

Example 11.31. *Approximate* $\int_0^{\pi} \frac{\sin x}{x}$ *by Simpson's rule with seven ordinates.*

Solution. Here $h = \frac{\pi - 0}{6} = \frac{\pi}{6}$.

The values of $y = \frac{\sin x}{x}$ computed at each point of sub-intervals are as follows :

x	:	0	$\pi/6$	$\pi/3$	$\pi/2$	$2\pi/3$	$5\pi/6$	$\pi = x_0 + 6h$
$\sin x$	:	0.00	0.50	0.86603	1.00	0.86603	0.50	0.00
$y = \frac{\sin x}{x}$	:	1	$\frac{3}{\pi}$	$\frac{2.59809}{\pi}$	$\frac{2}{\pi}$	$\frac{1.29904}{\pi}$	$\frac{0.6}{\pi}$	0

By Simpson's rule,

$$\int_0^{\pi} \frac{\sin x}{x} dx = \int_{x_0}^{x_0+6h} y\, dx = \frac{h}{3}[(y_0 + y_6) + 4(y_1 + y_3 + y_5) + 2(y_2 + y_4)]$$

$$= \frac{\pi}{18}\left[(0 + 1) + 4\left(\frac{3}{\pi} + \frac{2}{\pi} + \frac{0.6}{\pi}\right) + 2\left(\frac{2.59809}{\pi} + \frac{1.29904}{\pi}\right)\right]$$

$$= \frac{33.337}{18} = 1.8520$$

Example 11.32. *The velocity of a train which starts from rest is given by the following table, the time being reckoned in minute from the start and the speed in km/h. Find the distance transversed in 20 min.*

t	2	4	6	8	10	12	14	16	18	20	min
v	16	28.8	40	46.4	51.2	32.0	17.6	8	3.2	0	km/h

Solution. We know that $\frac{ds}{dt} = v$ or $ds = vdt$. Hence, $s = \int v\,dt$

Distance covered in 20 minutes is $s = \int_0^{20} vdt$

Since the train starts from rest, $v = 0$, when $t = 0$.

Here $h = \frac{2}{60}$ hr $= \frac{1}{30}$ hr

By Simpson's one-third rule,

$$\int_0^{20} vdt = \frac{h}{3}[v_0 + v_{10} + 4(v_1 + v_3 + v_5 + v_7 + v_9) + 2(v_2 + v_4 + v_6 + v_8)]$$

$$= \frac{1}{90}[0 + 0 + 4(16 + 40 + 51.2 + 17.6 + 3.2) + 2(28.8 + 46.4 + 32.0 + 8)]$$

$$= 8.25 \text{ km}$$

Example 11.33. *Evaluate* $\int_0^1 \frac{dx}{1 + x^2}$ *by using Simpson's one-third rule. Hence, obtain the approximate value of* π.

Solution. Divide the given range in six equal parts each of width $h = 1/6$ and compute the value of $y = \frac{1}{1 + x^2}$ at each point of sub-division. These values are given as follows :

x :	0	$\frac{1}{6}$	$\frac{2}{6}$	$\frac{3}{6}$	$\frac{4}{6}$	$\frac{5}{6}$	$1 = x_0 + 6h$
$y_n = \frac{1}{1+x^2}$:	1.0000	0.97297	0.90000	0.80000	0.69231	0.59016	0.50000

By Simpson's rule,

$$\int_0^1 ydx = \frac{h}{3}[y_0 + y_6 + 4(y_1 + y_3 + y_5) + 2(y_2 + y_4)]$$

$$= \frac{1}{18}[1.0000 + .50000 + 4(0.97297 + 0.80000 + 0.59016) + 2\,(0.90000 + 0.69231)]$$

$$= 0.785395 \qquad \ldots(1)$$

Again, $\int_0^1 \frac{dx}{1 + x^2} = [\tan^{-1} x]_1^0 = \tan^{-1} 1 - \tan^{-1} 0 = \pi/4$ $\qquad \ldots(2)$

From eqs. (1) and (2), $\pi/4 = 0.785395$

or $\pi = 3.14188$

Example 11.34. *Compute* $\int_1^2 \frac{1}{x}\,dx$ *by using (i) the trapezoidal rule, and (ii) the Simpson's one-third rule. Also, compute the error bounds.*

Solution. Divide the given range (1, 2) into four equal parts each of width $h = .25$. The values of $y = \frac{1}{x}$, computed at each point of sub-intervals are as follows:

x :	1	1.25	1.50	1.75	$2.00 = x_0 + 4h$
$y_n = \frac{1}{x}$:	1	.8	.6666667	.5714285	.5

(*i*) Using Trapezoidal rule,

$$\int_1^2 \frac{1}{x}\,dx = \int_{x_0}^{x_0+4h} y\,dx = \frac{h}{2}[y_0 + y_4 + 2(y_1 + y_2 + y_3)]$$

$$= \frac{.25}{2}[1 + .5 + 2(.8 + .6666667 + .5714285)]$$

$$= .6970238$$

The global error of the trapezoidal rule is $E = -\frac{(x_n - x_0)}{12} h^2 f''(\xi), \quad x_0 \le \xi \le x_n$

Hence, $E = -\frac{(2-1)}{12}(.25)^2 f''(\xi)$

Here $f(x) = \frac{1}{x}, \quad f'(x) = -\frac{1}{x^2}, \quad f''(\xi) = \frac{2}{x^3}$

Hence, $\max\limits_{1 \le x \le 2} f''(x) = 2$ and it occurs at $x = 1$

and $\min\limits_{1 \le x \le 2} f''(x) = \frac{2}{8}$ and it occurs at $x = 2$

Hence, $|E_{max}| = \frac{1}{12}(.25)^2 \times 2 = .0104166$

and $|E_{min}| = \frac{1}{12}(.25)^2 \times \frac{2}{8} = .001302$

(*ii*) Using Simpson's one-third rule,

$$\int_1^2 \frac{1}{x}\,dx = \int_{x_0}^{x_0+4h} y\,dx = \frac{h}{3}[y_0 + y_4 + 4(y_1 + y_3) + 2y_2]$$

$$= \frac{.25}{3}[1 + .5 + 4(.8 + .5714285) + 2 \times .6666667]$$

$$= .6932537$$

The global error in Simpson's one-third rule is

$$E = -\frac{(x_n - x_0)}{180} h^4 f^{iv}(\xi), \; x_0 \le \xi \le x_n$$

$$= -\frac{1}{180}(.25)^4 f^{iv}(\xi)$$

Here $f(x) = \frac{1}{x}$ Thus, $f^{iv}(x) = \frac{24}{x^5}$

Hence, $\max\limits_{1 \le x \le 2} f^{iv}(x) = 24$ and it occurs at $x = 1$

and $\min\limits_{1 \le x \le 2} f^{iv}(x) = \frac{24}{32} = \frac{3}{4}$ and it occurs at $x = 2$.

Hence, $|E_{max}| = \frac{1}{180}(.25)^4 \times 24 = .5208 \times 10^{-3}$

and $|E_{min}| = \frac{1}{180}(.25)^4 \times \frac{3}{4} = .1628 \times 10^{-4}$

Example 11.35. *Evaluate numerically* $\int_0^{\pi/4} \cos x\,dx$ *using Trapezoidal rule with step length so chosen that the error is less than 0.01.*

Solution. Here $y(x) = \cos x$, $y''(x) = -\cos x$ and $\max\limits_{0 \le x \le \pi/4} y''(x) = 1$

The error in Trapezoidal rule $= -\frac{h^2}{12}(x_n - x_0) f''(\xi)$

$$|E_{max}| = \left|-\frac{h^2}{12} \times \frac{\pi}{4} \times 1\right| = \frac{\pi h^2}{48}$$

According to the problem, $\frac{\pi h^2}{48} < 0.01$

or $$h^2 < \frac{.48}{3.42} = .14035$$

or $$h < .375$$

Thus, one could take $h = \pi/12$ (= 0.262) so that the error is less than 0.01.

REVIEW QUESTIONS SET

1. Following table gives the values of a function at equal intervals:

x	0.0	0.5	1.0	1.5	2.0
$f(x)$	.3989	0.3521	0.2420	0.1295	0.0540

Evaluate $\int_0^2 f(x)\,dx$, using Simpson's rule.

2. A river is 80 m wide. The depth (d in metre) at a distance x metre from one bank is given in the following table :

x :	0	10	20	30	40	50	60	70	80
d :	0	4	7	9	12	15	14	8	3

Find approximately the area of cross-section.

3. A body is in the form of a solid of revolution and is 6 cm long. Following table gives the diameter D (in cm) of the section at distance x cm from the end. Find the volume of the solid.

Distance	x	0	1	2	3	4	5	6
Diameter	D	2	2.2	2.4	2.7	2.5	2.2	2

[**Hint:** The cross-sectional area of diameter D is given by $\pi D^2/4$]

4. A rod is rotating in a plane about one of its ends. The following table gives the angle θ (radians) through which the rod has turned for various values of time (sec). Find the (*a*) angular velocity of the rod, and (*b*) angular acceleration when $t = 0.7$ sec.

t :	0	0.2	0.4	0.6	0.8	1.0	1.2
θ :	0	1.22	0.493	1.123	2.022	3.200	4.666

5. A curve is drawn to pass through the point given by the following table:

x :	1	1.5	2	2.5	3	3.5	4
y :	2	2.4	2.7	2.8	3	2.6	2.1

6. Calculate the approximate value of $\int_{-3}^{3} x^4\,dx$ by using (*a*) Trapezoidal rule, and (*b*) Simpson's rule by dividing the range in six equal parts.

7. Evaluate $\int_0^{\pi/2} \sqrt{\sin x}\,dx$, taking $h = \pi/12$.

8. Applying Simpson's one-third rule, evaluate (*a*) $\int_{1/2}^{1} \frac{dx}{x}$ with five ordinates and from the result, find log 2 correct to three decimal places; and (*b*) $\int_2^3 \frac{dx}{x}$, taking 10 equal intervals and from the result, find $\log_e (3/2)$.

9. Evaluate $\int_0^{\pi} \sin^3 x\,dx$ by Simpson's one-third rule using 4 strips.

10. Find the value of log 3 from $\int_0^1 \frac{x^2}{1+x^3}\,dx$, using Simpson's one-third rule, by dividing the range into four equal parts. Also find the error.
11. Determine the value of h so that the Trapezoidal rule will yield the value of $I = \int_0^1 e^{-x^2}\,dx$ to 6 significant figures and hence find the number of divisions required.
12. Find the approximate value of $I = \int_0^1 \frac{dx}{1+x}$, using Simpson's rule. **(AMIE, S '94)**
13. Calculate $\int_2^{10} \frac{dx}{1+x}$ by Simpson's one-third rule (use eight equal parts). **(AMIE, S '96)**
14. Find the value of $\log_e 2$ by evaluating the integral correct up to four decimal places

$$\int_0^1 \frac{dx}{1+x}$$

numerically usiing Simpson's one-third rule. Take 10 equal intervals. Compare the result with analytical result. **(AMIE, W '97)**

ANSWERS TO REVIEW QUESTIONS SET

1. 0.4772 **2.** 710 m^2 **3.** 26.21 cm^3 **4.** (*a*) 3.814 rad/sec (*b*) 6.725 rad/sec^2
5. 7.74 **6.** (*a*) 115 (*b*) 98 **7.** 1.1873 **8.** (*a*) 0.6933, 0.693
8. (*b*) 0.4054 **9.** 1.342 **10.** .23108, error = – 0.00003 **11.** h = .001, n = 500
12. .663 **13.** 1.289 **14.** 0.6931

11.6. NUMERICAL SOLUTIONS OF ORDINARY DIFFERENTIAL EQUATIONS

Many problems in science and technology can be reduced to the problems of solving differential equations under certain given conditions. But all differential equations cannot be solved analytically and, hence, the numerical solutions plays a very important role. Many techniques have been developed for approximate solutions of ordinary differential equations by numerical methods though all of them lack the generality of the nature that an analytic approach would yield, that is, it does not yield the general solution but instead a set of points which lie close to a particular solution of the equation. The important methods of solving ordinary differential equations are:

(*i*) Picard Method
(*ii*) Taylor Series Method
(*iii*) Euler Method
(*iv*) Modified Euler Method
(*v*) Runge-Kutta Method

The solution may take one of the two forms:

(*a*) an expression for y in terms of x from which the values of y are obtained by direct substitution;
(*b*) a series of values of y corresponding to some specific values of x.

The methods of Taylor and Picard belong to form (*a*) and of Euler, Modified Euler and Runge-Kutta belong to form (*b*). The methods of (*b*) are called step-by-step methods or marching methods because the values of y are computed by short steps ahead of equal intervals h of the independent variables.

We shall restrict our consideration for the solution of first order initial value problem of the type $\frac{dy}{dx} = f(x, y)$ with the initial condition $y(x_0) = y_0$.

Picard's Method

Consider the following differential equation:

$$\frac{dy}{dx} = f(x, y) \qquad ...(1)$$

with initial condition $y(x_0) = y_0$.

Integrating the equation (1) between the limits x_0 and x,

$$\int_{y_0}^{y} dy = \int_{x_0}^{x} f(x, y)\, dx \qquad \text{or} \qquad y - y_0 = \int_{x_0}^{x} f(x, y)\, dx$$

or $$y = y_0 + \int_{x_0}^{x} f(x, y)\, dx \qquad ...(2)$$

Equation (2), in which the unknown function y appears under the integral sign, is called an integral equation. For a first approximation of y, replace y by y_0 in $f(x, y)$ of (2) to get

$$y^{(1)} = y_0 + \int_{x_0}^{x} f(x, y_0)\, dx, \text{ where } y^{(1)} \text{ denotes the first approximation of } y.$$

For second approximation, replace y by $y^{(1)}$ in $f(x, y)$ of (2) to get

$$y^{(2)} = y_0 + \int_{x_0}^{x} f(x, y^{(1)})\, dx$$

and so on.

In general, $y^{(n)} = y_0 + \int_{x_0}^{x} f(x, y^{(n-1)})\, dx$

The process is terminated when two consecutive values of y, viz., $y^{(n-1)}$ and $y^{(n)}$ are same up to the desired degree of accuracy.

Picard's method is of considerable theoretical value, but, in practice, the repeated evaluation of integral involved is not always simple and sometime becomes impossible. This limits the use of the method.

Example 11.36. *Use Picard's method to find the value of y when $x \le 0.2$ correct up to three decimal places given that $y(0) = 1$ and $\frac{dy}{dx} = x - y$.*

Solution. Given $\frac{dy}{dx} = x - y$

Hence, $$y = y_0 + \int_{x_0}^{x} f(x, y)\, dx$$

$$= 1 + \int_0^x (x - y)\, dx$$

Now, $$y^{(1)} = 1 + \int_0^x (x - 1)\, dx = 1 + \frac{x^2}{2} - x$$

When $x = 0.2$, $$y^{(1)} = 1 + \frac{(0.2)^2}{2} - 0.2 = 0.82$$

Second approximation is

$$y^{(2)} = 1 + \int_0^x (x - y^{(1)})\, dx = 1 + \int_0^x \left(x - 1 - \frac{x^2}{2} + x\right) dx$$

$$= 1 + \int_0^x \left(2x - \frac{x^2}{2} - 1\right) dx$$

$$= 1 + x^2 - \frac{x^3}{6} - x$$

When $x = 0.2$,

$$y^{(2)} = 1 + (0.2)^2 - \frac{(0.2)^3}{6}\ 0.2$$

$$= 0.83867$$

Third approximation is

$$y^{(3)} = 1 + \int_0^x (x - y^{(2)})\,dx = 1 + \int_0^x x - \left(1 + x^2 - \frac{x^3}{6} + x\right)dx$$

$$= 1 + \int_0^x \left(2x + \frac{x^3}{6} - x^2 - 1\right)dx$$

$$= 1 + x^2 + \frac{x^4}{24} - \frac{x^3}{3} - x$$

When $x = 0.2$,

$$y^{(3)} = 1 + (0.2)^2 + \frac{(0.2)^4}{24} - \frac{(0.2)^3}{3} - 0.2$$

$$= 1 + 0.04 + .000067 - .002667 - 0.2$$

$$= 0.83740$$

Since the value of $\frac{x^4}{24}$ at $x = 0.2$ contains four zeros after decimal, the value of higher power of x at $x = 0.2$ will have no effect up to three decimal places.

Fourth approximation is

$$y^{(4)} = 1 + \int_0^x (x - y^{(3)})\,dx = 1 + \int_0^x x - \left(1 + x^2 + \frac{x^4}{24} - \frac{x^3}{3} - x\right)dx$$

$$= 1 + \int_0^x \left(2x - \frac{x^4}{24} + \frac{x^3}{3} - x^2 - 1\right)dx$$

$$= 1 + x^2 - \frac{x^5}{120} + \frac{x^4}{12} - \frac{x^3}{3} - x$$

When $x = 0.2$, $\quad y^{(4)} = 0.83746$

Since the values of $y^{(3)}$ and $y^{(4)}$ are same up to three decimal places, $y = 0.837$ when $x = 0.2$. The result is correct up to three decimal places.

Example 11.37. Using Picard's method, find the solution of $\frac{dy}{dx} = 1 + xy$ *at* $x = 0.1, 0.2$, *given that* $y = 1$ *at* $x = 0$ *such that* y *is correct to three decimal places.*

Solution. Given $\frac{dy}{dx} = 1 + xy$

and $$y = y_0 + \int_{x_0}^x f(x, y)\,dx$$

$$= 1 + \int_0^x (1 + xy)\,dx$$

Now, $$y^{(1)} = 1 + \int_0^x (1 + xy_0)\,dx = 1 + \int_0^x (1 + x)\,dx$$

$$= 1 + x + \frac{x^2}{2}$$

When $x = 0.1$, $\quad y^{(1)} = 1 + 0.1 + \frac{(0.1)^2}{2} = 1 + .1 + .005 = 1.105$

$x = 0.2$, $\quad y^{(1)} = 1 + 0.2 + \frac{(0.2)^2}{2} = 1 + 0.2 + .02 = 1.22$

Second approximation is

$$y^{(2)} = 1 + \int_0^x (1 + xy^{(1)})\,dx = 1 + \int_0^x 1 + x\left(1 + x + \frac{x^2}{2}\right)dx$$

$$= 1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{8}$$

When $x = 0.1$, $y^{(2)} = 1.10534$

$x = 0.2$, $y^{(2)} = 1.2229$

Hence, $y = 1.105$ when $x = 0.1$. The result is correct up to three decimal places.

Third approximation is

$$y^{(3)} = 1 + \int_0^x (1 + xy^{(2)})\,dx$$

$$= 1 + \int_0^x \left(1 + x\left(1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{8}\right)\right)dx$$

$$= 1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{8} + \frac{x^5}{15} + \frac{x^6}{48}$$

When $x = 0.2$, $y^{(3)} = 1.2229$

Hence, $y = 1.223$, when $x = 0.2$. The result is correct up to three decimal places.

Taylor's Series Method

Let us consider a differential equation

$$\frac{dy}{dx} = f(x, y) \quad \text{with the initial condition } y(x_0) = y_0.$$

If $y(x)$ is the exact solution of the above equation, then the Taylor's series for $y(x)$ around $x = x_0$ is given by

$$y(x) = y_0 + (x - x_0)\,y_0' + \frac{(x - x_0)^2}{2!}\,y_0'' + \ldots.$$

where y_0', y_0'', ... are the first, second, ... derivatives of y at $x = x_0$.

The value of y at x can be determined to any desired degree of accuracy provided the initial value of various derivatives of y as used in the equation are known.

The method is best understood by the following examples:

Example 11.38. *Find the value of* y *at* $x = 0.1$ *correct to four places of decimal, given that* $\frac{dy}{dx} = x - y^2$ *with* $y(0) = 0$.

Solution. Given $\frac{dy}{dx} = x - y^2$

i.e., $y' = x - y^2$

$y_0' = x_0 - y_0^2 = 0 - 0 = 0$

$y'' = 1 - 2yy'$

$y_0'' = 1 - 2y_0\,y_0' = 1 - 2 \times 0 \times 0 = 1$

$y''' = 0 - 2(y')^2 - 2yy''$

$y_0''' = -2(y_0')^2 - 2y_0y_0'' = 0$

$y^{iv} = -4y'\,y'' - 2y'y''' - 2yy'''$

$y_0^{iv} = -4y_0'y_0'' - 2y_0'y_0''' - 2y_0y_0''' = 0$

$y^{v} = -6(y'')^2 - 6y'y''' - 2y'y''' - 2yy^{iv}$

$y_0^{v} = -6\,(y_0'')^2 - 6y_0'y_0''' - 2y_0'y_0''' - 2y_0y_0^{iv}$

$= -6 \times 1 - 0 - 0 - 0 = -6$

Therefore, Taylor's series about $x = 0$ gives

$$y = y_0 (x - 0)y_0' + \frac{(x - 0)^2}{2!} y_0'' + \frac{(x - 0)^3}{3!} y_0''' + \ldots.$$

$$= \frac{x^2}{2!} - 6\frac{x^5}{5!} + \ldots.$$

Hence, $$y(0.1) = \frac{(0.1)^2}{2} - 6\frac{(0.1)^2}{5!} + \ldots.$$

$$= 0.005 - 0.0000005$$

$$= 0.0050 \quad \text{correct up to 4 decimal places.}$$

Example 11.39. *Using Taylor's series method, find y correct up to five places of decimal when $x = 1.02$, given that $\frac{dy}{dx} = xy - 1$ with $y(1) = 2$.*

Solution. Given $$\frac{dy}{dx} = xy - 1$$

i.e., $$y' = xy - 1$$

$$y_0' = x_0y_0 - 1 = 1 \times 2 - 1 = 1$$

$$y'' = y + xy'$$

$$y_0'' = y_0 + x_0y_0' = 2 + 1 \times 1 = 3$$

$$y''' = y' + y' + xy'' = 2y' + xy''$$

$$y_0''' = 2y_0' + x_0y_0'' = 2 \times 1 + 1 \times 3 = 5$$

$$y^{iv} = 2y'' + y'' + xy''' = 3y'' + xy'''$$

$$y_0^{iv} = 3y_0'' + x_0y_0''' = 3 \times 3 + 1 \times 5 = 14$$

By Taylor's series,

$$y = y_0 + (x - x_0)\, y_0' + \frac{(x - x_0)^2}{2!} y_0'' + \frac{(x - x_0)^3}{3!} y_0''' + \frac{(x - x_0)^4}{4!} y_0^{iv} + \ldots.$$

$$= 2 + (x - 1) \times 1 + \frac{(x - 1)^2}{2!} \times 3 + \frac{(x - 1)^3}{6} \times 5 + \frac{(x - 1)^4}{24} \times 14$$

At $x = 1.02$,

$$y = 2 + (1.02 - 1) + \frac{3}{2}(1.02 - 1)^2 + \frac{5}{6}(1.02 - 1)^3 + \frac{7}{12}(1.02 - 1)^4$$

$$= 2 + .02 + .0006 + .000006 + .00000009$$

$$= 2.02061, \text{ correct up to five places of decimal.}$$

To find the result correct up to five places of decimal, the terms up to $\frac{5}{6}(x - 1)^3$ should be considered, since at $x = 1.02$, the term $\frac{7}{12}(x - 1)^4$ contains seven zeros after decimal which will have no effect in the result up to five places of decimal.

Euler's Method

Consider the differential equation

$$\frac{dy}{dx} = f(x, y) \text{ with initial condition } y(x_0) = y_0.$$

Let $y = \phi(x)$ be the solution.

Let y have value y_{n+1} when $x = x_{n+1}$.

$$y_{n+1} = \phi(x_{n+1}) = \phi(x_n + h)$$

We write $y_{n+1} = \phi(x_n + h)$

or $$y_{n+1} = \phi(x_n) + h\phi'(x_n) + \frac{h^2}{2!}\phi''(x_n) + \ldots. \quad \text{(from Taylor's series)}$$

$$= y_n + h\phi'(x_n) + \frac{h^2}{2!}\phi''(x_n) + \ldots.$$

$$= y_n + hf(x_n, y_n) + \frac{1}{2}h^2\phi''(x_n) + \ldots.$$

If the step length h is small, the term containing h^2, h^3, h^4, etc., are still smaller.

Therefore, $y_{n+1} = y_n + hf(x_n, y_n)$ ...(1)

Equation (1) is known as first order Euler's method. Putting $n = 0, 1, 2, \ldots.$, we obtain $y_1, y_2, y_3, \ldots.$

Example 11.40. Given $\frac{dy}{dx} = \frac{y - x}{y + x}$, $y(0) = 1$. *Using Euler's method, obtain y for* $x = 0.1$.

Solution. Here $x_0 = 0$, $y_0 = 1$, $h = 0.1$ and $f(x_n, y_n) = \frac{y_n - x_n}{y_n + x_n}$.

We have $y_{n+1} = y_n + hf(x_n, y_n)$ ...(1)

Putting $n = 0$ in (1),

$$y_1 = y_0 + hf(x_0, y_0)$$

$$= y_0 + h\frac{y_0 - x_0}{y_0 + x_0}$$

$$= 1 + (0.1)\frac{1 - 0}{1 + 0} = 1 + .1 = 1.1$$

Better accuracy can be obtained by taking h smaller (say .02), i.e., by dividing the interval 0 to 0.1 into five steps. The calculations and results at each step are shown in the following table:

n	x_n	y_n	$f(x_n, y_n) = \frac{y_n - x_n}{y_n + x_n}$	$y_{n+1} = y_n + hf(x_n, y_n)$
0	0	1	1	$y_1 = 1 + 0.2 \times 1 = 1.02$
1	.02	1.02	.9615	$y_2 = 1.02 + 0.2 \times .9615 = 1.0392$
2	.04	1.0392	.9259	$y_3 = 1.0392 + 0.2 \times .9259 = 1.0577$
3	.06	1.0577	.8926	$y_4 = 1.0577 + 0.2 \times .8926 = 1.0756$
4	.08	1.0756	.8615	$y_5 = 1.0756 + 0.2 \times .8615 = 1.0928$
5	.10	1.0928		

Hence, the approximate value of y at $x = .1$ is 1.0928.

Example 11.41. Solve $\frac{dy}{dx} = x + y$ *with the initial condition* $x_0 = 0$, $y_0 = 1$ *and* $h = 0.1$, *using Euler's method for* $x = 0(0.1)0.4$.

Solution. Here $x_0 = 0$, $y_0 = 1$, $h = 0.1$ and $f(x_n, y_n) = x_n + y_n$.

We have $y_{n+1} = y_n + hf(x_n, y_n)$ ...(1)

The calculation and results at each step are shown in the following table:

n	x_n	y_n	$f(x_n, y_n) = x_n + y_n$	$y_{n+1} = y_n + hf(x_n, y_n)$
0	0	1	1.00	$y_1 = 1 + 0.1 \times 1.00 = 1.10$
1	0.1	1.10	1.20	$y_2 = 1.10 + 0.1 \times 1.20 = 1.22$
2	0.2	1.22	1.42	$y_3 = 1.22 + 0.1 \times 1.42 = 1.362$
3	0.3	1.362	1.662	$y_4 = 1.362 + 0.1 \times 1.662 = 1.5282$
4	0.4	1.5282		

Hence, approximate values of y corresponding to $x = 0.1, 0.2, 0.3$ and 0.4 are 1.10, 1.22, 1.362, 1.5282, respectively.

Example 11.42. *Solve the following differential equation by Euler's method:*

$$\frac{dy}{dx} = x^2 - y. \text{ with initial condition } y(0) = 1.$$

Solution is required over the interval [0, 0.2] with $h = 0.05$. **(AMIE, W '95)**

Solution. Here $x_0 = 0$, $y_0 = 1$, $h = 0.05$ and $f(x_n, y_n) = x_n^2 - y_n$.

We have $y_{n+1} = y_n + hf(x_n, y_n)$ ...(1)

The calculation and results at each step are shown in the following table:

n	x_n	y_n	$f(x_n, y_n) = x_n^2 - y_n$	$y_{n+1} = y_n + hf(x_n, y_n)$
0	0	1	–1	$y_1 = 1 + 0.05\ (-1) = 0.95$
1	.05	0.95	– .9475	$y_2 = .95 + 0.05\ (-.9475) = .9026$
2	.10	.9026	– .8926	$y_3 = .9026 + 0.05\ (-.8926) = .8579$
3	.15	.8579	– .8354	$y_4 = .8579 + 0.05\ (-.8354) = .8161$
4	.20	.8161		

Modified Euler's Method

It may be noted that the numerical solution by Euler's method gives a poor approximation to the true value. For obtaining better solution, Euler's modified method may be used. The slope value is calculated by taking the average of the slope values at the start and end points of the interval.

Starting with the initial value y_0, an approximate value for y_1 is computed from the relation

$$y_1^{(1)} \approx y_0 + \left(\frac{dy}{dx}\right)_{x=x_0} h = y_0 + f(x_0, y_0).h \quad \text{...(1)}$$

where $y_1^{(1)}$ is the first approximation of y at $x = x_1$.

After obtaining the value of $y_1^{(1)}$, substitute it in the differential equation $\frac{dy}{dx} = f(x, y)$ so as to get the approximate value of $\frac{dy}{dx}$ at the end of first interval, viz.,

$$\left(\frac{dy}{dx}\right)_1^{(1)} = f(x_1, y_1^{(1)}) \quad \text{...(2)}$$

Now, the second approximation of y_1 is given by

$$y_1^{(2)} = y_0 + \frac{\left(\frac{dy}{dx}\right)_{x_0} + \left(\frac{dy}{dx}\right)_{x_1}^{(1)}}{2} h = y_0 + \frac{f(x_0, y_0) + f(x_1, y_1^{(1)})}{2} h$$

The improved value of $y_1^{(2)}$ is now substituted in the equation $\frac{dy}{dx} = f(x, y)$ so as to get the second approximation of $\left(\frac{dy}{dx}\right)_1$, viz. $\left(\frac{dy}{dx}\right)_1^{(2)} = f(x_1, y_1^{(2)})$.

The third approximation of y is, therefore, given by

$$y_1^{(3)} = y_0 + \frac{\left(\frac{dy}{dx}\right)_{x_0} + \left(\frac{dy}{dx}\right)_{x_1}^{(2)}}{2} h = y_0 + \frac{f(x_0, y_0) + f(x_1, y_1^{(2)})}{2} h$$

The process is terminated, when the value of $y_1^{(n)}$ and $y_1^{(n+1)}$ are same up to the required degree of accuracy. Hence,

$$y_1^{(n+1)} = y_0 + \frac{\left(\frac{dy}{dx}\right)_{x_0} + \left(\frac{dy}{dx}\right)_{x_1}^{(n)}}{2} h = y_0 + \frac{f(x_0, y_0) + f(x_1, y_1^{(n)})}{2} h.$$

Example 11.43. *Solve* $\frac{dy}{dx} = x + y$ *from 0 to* $x = 0.2$ *by Euler's modified method with initial condition* $y(0) = 1$.

Solution. $y_1^{(n+1)} = y_0 + \frac{1}{2} h [f(x_0, y_0) + f(x_1, y_1^{(n)}], \quad n = 0, 1, 2, \ldots.$

where $y_1^{(n)}$ is nth approximation of y at $x = x_1$.

Take $h = .05$ and start with initial values,

$$y_1^{(1)} = y_0 + hf(x_0, y_0) \approx 1 + .05\ (0 + 1) = 1.05$$

$$y_1^{(2)} = y_0 + \frac{1}{2} h [f(x_0, y_0) + f(x_1, y_1^{(1)})]$$

or $$y_1^{(2)} = 1 + \frac{.05}{2} [1 + .05 + 1.05] = 1.0525$$

$$y_1^{(3)} = y_0 + \frac{1}{2} h [f(x_0, y_0) + f(x_1, y_1^{(2)})]$$

or $$y_1^{(3)} = 1 + \frac{.05}{2} [1 + .05 + 1.0525] = 1.05256$$

$$y_1^{(4)} = 1 + \frac{.05}{2} [1 + .05 + 1.05256] = 1.052564$$

Since $y_1^{(3)}$ and $y_1^{(4)}$ are same up to five decimal places, $y_1 = 1.05256 = 1.0526$ (up to four decimal).

We take $y_1 = 1.05256$ and $f(x_1, y_1) = x_1 + y_1 = .05 + 1.05256 = 1.10256 = 1.1026$

For the second value of y (first approximation),

$$y_2^{(1)} = y_1 + hf(x_1, y_1) = 1.0526 + .05\ (1.1026) = 1.1077$$

$$y_2^{(2)} = y_1 + \frac{h}{2} [f(x_1, y_1) + f(x_2, y_2^{(1)})]$$

$$= 1.0526 + \frac{.05}{2} [1.1026 + 1.1077] = 1.1104$$

$$y_2^{(3)} = y_1 + \frac{h}{2} [f(x_1, y_1) + f(x_2, y_2^{(2)})]$$

$$= 1.0526 + \frac{.05}{2}[1.1026 + .10 + 1.1104]$$

$$= 1.1104$$

Since $y_2^{(2)}$ and $y_2^{(3)}$ are same up to four decimal places, $y_2 = 1.1104$ (correct up to four decimal places).

Note : As **soon as** two consecutive values of y are known, the first approximation to succeeding **y's can** be obtained more accurately by the formula

$$y_{n+1} = y_{n-1} + 2h f(x_n, y_n)$$

which can be obtained by making use of Taylor's series in the neighbourhood of x_n.

Example 11.44. *Use Euler's modified method to determine the value of y when $x = 0.05$ and 0.1 correct up to four decimal places, given that $\frac{dy}{dx} = x^2 + y$ with initial condition $y(0) = 1$.*

Solution. Take $h = .05$

$$y_1^{(1)} = y_0 + hf(x_0, y_0) = 1 + .05\ (0 + 1) = 1.05$$

$$y_1^{(2)} = y_0 + \frac{1}{2}h\,[f(x_0, y_0) + f(x_1, y_1^{(1)})]$$

$$= 1 + \frac{.05}{2}[1 + x_1^2 + y_1^{(1)}]$$

$$= 1 + \frac{.05}{2}[1 + (.05)^2 + 1.05] = 1.0513$$

$$y_1^{(3)} = y_0 + \frac{1}{2}h\,[f(x_0, y_0) + f(x_1, y_1^{(2)}]$$

$$= 1 + \frac{.05}{2}[1 + (.05)^2 + 1.0513] = 1.0513$$

Since $y_1^{(2)}$ and $y_1^{(3)}$ are same up to four decimal places, $y_1 = 1.0513$ (correct up to four decimal places).

To find out the value of y at $x = .1$, we take $x_1 = .05$, $y_1 = 1.0513$ and $f(x_1, y_1) = x_1^2 + y_1 = (.05)^2 + 1.0513 = 1.0538$.

To obtain the first approximation of y_2, use the following formula:

$$y_{n+1} = y_{n-1} + 2hf(x_n, y_n)$$

Hence,

$$y_2^{(1)} = y_0 + 2hf(x_1, y_1)$$

$$= 1 + 2 \times .05 \times 1.0538$$

$$= 1.1054$$

$$y_2^{(2)} = y_1 + \frac{1}{2}h\,[f(x_1, y_1) + f(x_2, y_2^{(1)})]$$

$$= 1.0513 + \frac{.05}{2}[1.0538 + (.1)^2 + 1.1054]$$

$$= 1.1055$$

$$y_2^{(3)} = y_1 + \frac{1}{2}h\,[f(x_1, y_1) + f(x_2, y_2^{(2)})]$$

$$= 1.0513 + \frac{.05}{2}[1.0538 + (.1)^2 + 1.1055]$$

$$= 1.1055$$

Since $y_2^{(2)}$ and $y_2^{(3)}$ are same up to four decimal places, $y_2 = 1.1055$ (correct up to four decimal places).

Runge-Kutta Method

The Runge-Kutta methods is probably the most widely used method for numerical solution of first order, first degree differential equations. The advantages of this method are:

(*i*) They are self-starting,
(*ii*) Easy to program for a digital computer, and
(*iii*) More computer storage space is available, since earlier values need not be stored.

In order to use Runge-Kutta method, consider the differential equation $\frac{dy}{dx} = f(x, y)$ with initial condition $y(x_0) = y_0$.

In the modified Euler's method,

$$y_1 = y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1)] \quad ...(1)$$

If we substitute $y_1 = y_0 + hf(x_0, y_0)$ on the right side of the equation,

$$y_1 = y_0 + \frac{h}{2}[f_0 + f(x_0 + h, y_0 + hf_0)]$$

where $f_0 = f(x_0, y_0)$.

If we now set $k_1 = hf_0$ and $k_2 = hf(x_0 + h, y_0 + k_1)$, then the above equation becomes

$$y_1 = y_0 + \frac{1}{2}[k_1 + k_2]$$

which is the **second order Runge-Kutta formula.** The error in this formula is of the order h^3. This is only just one of the several second-order Runge-Kutta formulae that exist.

The above formula can be written in general as

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2)$$

where $k_1 = hf(x_n, y_n)$

and $k_2 = hf(x_n + h, y_n + k_1)$

Higher order Runge-Kutta formulae exist, of which we mention only the **fourth-order formula,** the most commonly used in practice:

If
$$k_1 = hf(x_0, y_0)$$
$$k_2 = hf\left(x_0 + \tfrac{1}{2}h, y_0 + \tfrac{1}{2}k_1\right)$$
$$k_3 = hf\left(x_0 + \tfrac{1}{2}h, y_0 + \tfrac{1}{2}k_2\right)$$
$$k_4 = hf(x_0 + h, y_0 + k_3)$$

then $y_1 = y_0 + (1/6)(k_1 + 2k_2 + 2k_3 + k_4)$,

in which the error is of order h^5.

In general, $y_{n+1} = y_n + \frac{1}{6}(k_1 + k_2 + k_3 + k_4)$ ...(1)

where $k_1 = hf(x_n, y_n)$ ...(2)

$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + k_1/2\right) \quad ...(3)$$

$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + k_2/2\right) \quad ...(4)$$

$$k_4 = hf(x_n + h, y_n + k_3) \quad ...(5)$$

The use of the method is as follows:

(*i*) Compute the values of k from equations (2), (3), (4) and (5);

(*ii*) Substitute the values obtained from (*i*) in (1); and
(*iii*) Repeat the process at the required points in the interval, if any.

Example 11.45. *Consider* $\frac{dy}{dx} = x - y$ *with initial condition* $y(1) = 1$. *Find* $y(1.1)$ *using second order Runge-Kutta method.*

Solution. Take $h = .1$

For $n = 0$, $k_1 = hf(x_0, y_0) = .1(x_0 - y_0)$, **where** $f(x, y) = x - y$

$= .1(1 - 1) = 0$

$k_2 = hf(x_0 + h, y_0 + k_1) = hf(1 + .1, 1 + 0)$

$= hf(1.1, 1)$

$= .1 \times (1.1 - 1) = .1 \times .1 = .01$

Hence, $y_1 = y_0 + \frac{1}{2}(k_1 + k_2)$

$= 1 + \frac{1}{2}(0 + .01) = 1 + .005 = 1.005$

or $y(1.1) = 1.005$

Example 11.46. *Apply Runge-Kutta (fourth order) method to find an approximate value of* y *for* $x = 0.2$ *in steps of* 0.1 *if* $\frac{dy}{dx} = x + y^2$, *given that* $y = 1$, *when* $x = 0$. **(AMIE, S '96)**

Solution. Here $f(x, y) = x + y^2$, $x_0 = 0$, $y_0 = 1$. Take $h = 0.1$.

For $n = 0$,

$$k_1 = hf(x_0, y_0) = 0.1 \times (x_0 + y_0^2) = 0.1 \times (0 + 1^2) = 0.1$$

$$k_2 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right) = 0.1 \times f\left(0 + \frac{0.1}{2}, 1 + \frac{0.1}{2}\right)$$

$$= 0.1 \times f(0.5, 1.05) = 0.1 \times [0.5 + (1.05)^2] = 0.1152$$

$$k_3 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right) = 0.1 \times f\left(0.5, 1 + \frac{0.1152}{2}\right)$$

$$= 0.1 \times f(0.5, 1.0576) = 0.1 \times [0.5 + (1.0576)^2] = 0.1168$$

$$k_4 = hf(x_0 + h, y_0 + k_3) = 0.1 \times f(0.1, 1 + 0.1168)$$

$$= 0.1 \times f(0.1, 1.1168) = 0.1 \times [0.1 + (1.1168)^2] = 0.1347$$

Hence, $y_1 = y(0.1) = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 1.1164$

For $n = 1$,

$$k_1 = hf(x_1, y_1) = 0.1 \times (x_1 + y_1^2) = 0.1 \times [.1 + (1.1164)^2] = 0.1346$$

$$k_2 = hf\left(x_1 + \frac{h}{2}, y_1 + \frac{k_1}{2}\right) = 0.1 \times f\left(0.1 + \frac{0.1}{2}, 1.1164 + \frac{0.1346}{2}\right)$$

$$= 0.1 \times f(0.15, 1.837) = 0.1 \times [.15 + (1.837)^2] = 0.1551$$

$$k_3 = hf\left(x_1 + \frac{h}{2}, y_1 + \frac{k_2}{2}\right) = 0.1 \times f\left(0.15, 1.1164 + \frac{.1551}{2}\right)$$

$$= 0.1 \times f(0.15, 1.9395) = 0.1 \times [.15 + (1.9395)^2] = 0.1576$$

$$k_4 = hf(x_1 + h, y_1 + k_3) = 0.1 \times f(0.1 + 0.1, 1.1164 + 0.1576)$$

$$= 0.1 \times f(0.2, 1.2740) = 0.1 \times [0.2 + (1.2740)^2] = 0.1823$$

Hence, $y_2 = y(0.2) = y_1 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$$= 1.1164 + \frac{1}{6}(0.1346 + 2 \times 0.1551 + 2 \times 0.1576 + 0.1823)$$

$$= 1.2734$$

Example 11.47. *Solve the equation by Runge-Kutta method:*

$$\frac{dy}{dx} = \frac{y - x}{y + x} \quad \text{for } x = 0.1,\ 0.2 \text{ and } 0.3,\ x_0 = 0,\ y_0 = 1.$$

(AMIE, S '93)

Solution. Here $f(x, y) = \dfrac{y - x}{y + x}$, $x_0 = 0$, $y_0 = 1$. Take $h = 0.1$

For $n = 0$,

$$k_1 = hf(x_0, y_0) = 0.1\left(\frac{y_0 - x_0}{y_0 + x_0}\right) = 0.1\left(\frac{1 - 0}{1 + 0}\right) = 0.1$$

$$k_2 = hf\left(x_0 + \frac{h}{2},\ y_0 + \frac{k_1}{2}\right) = 0.1 \times f\left(0 + \frac{.1}{2},\ 1 + \frac{.1}{2}\right)$$

$$= 0.1 \times f(.05,\ 1.05) = 0.1 \times \frac{1.05 - 0.5}{1.05 + 0.5} = 0.0909$$

$$k_3 = hf\left(x_0 + \frac{h}{2},\ y_0 + \frac{k_2}{2}\right) = 0.1 \times f\left(.05,\ 1 + \frac{.0909}{2}\right)$$

$$= 0.1 \times f(.05,\ 1.04545) = 0.1 \times \frac{1.04545 - .05}{1.04545 + .05} = 0.0909$$

$$k_4 = hf(x_0 + h,\ y_o + k_3) = 0.1 \times f(0 + .1,\ 1 + .0909)$$

$$= 0.1 \times f(.1,\ 1.0909) = 0.1 \times \frac{1.0909 - 0.1}{1.0909 + 0.1} = 0.0832$$

Hence, $y_1 = y(1.1) = y_0 + \dfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$$= 1 + \frac{1}{6}(0.1 + 2 \times 0.0909 + 2 \times 0.0909 + 0.0832) = 1.0911$$

For $n = 1$,

$$k_1 = hf(x_1, y_1) = 0.1\left(\frac{y_1 - x_1}{y_2 + x_2}\right) = 0.1\left(\frac{1.0911 - .1}{1.0911 + .1}\right) = 0.0832$$

$$k_2 = hf\left(x_1 + \frac{h}{2},\ y_1 + \frac{k_1}{2}\right) = 0.1 \times f\left(.1 + \frac{.1}{2},\ 1.0911 + \frac{0.0832}{2}\right)$$

$$= 0.1 \times f(.15,\ 1.1327) = 0.1 \times \frac{1.1327 - .15}{1.1327 + .15} = 0.0766$$

$$k_3 = hf\left(x_1 + \frac{h}{2},\ y_1 + \frac{k_2}{2}\right) = 0.1 \times f\left(.15,\ 1.0911 + \frac{0.0766}{2}\right)$$

$$= 0.1 \times f(.15,\ 1.1294) = 0.1 \times \frac{1.1294 - .15}{1.1294 + .15} = 0.0765$$

$$k_4 = hf(x_1 + h,\ y_1 + k_3) = 0.1 \times f(.1 + .1,\ 1.0911 + 0.0765)$$

$$= 0.1 \times f(.2,\ 1.1676) = 0.1 \times \frac{1.1676 - .2}{1.1676 + .2} = 0.0708$$

Hence, $y_2 = y(1.2) = y_1 + \dfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$$= 1.1 + \frac{1}{6}(0.0832 + 2 \times 0.0766 + 2 \times 0.0765 + 0.0708)$$

$$= 1.1678$$

For $n = 2$, $k_1 = hf(x_2, y_2) = 0.1\left(\dfrac{y_2 - x_2}{y_2 + x_2}\right) = 0.1\left(\dfrac{1.1678 - .2}{1.1678 + .2}\right) = 0.0708$

$$k_2 = hf\left(x_2 + \frac{h}{2},\ y_2 + \frac{k_1}{2}\right) = 0.1 \times f\left(.2 + \frac{.1}{2},\ 1.678 + \frac{.0708}{2}\right)$$

$$= 0.1 \times f(.25,\ 1.2032) = 0.1 \times \frac{1.2032 - .25}{1.2032 + .25} = .0656$$

$$k_3 = hf\left(x_2 + \frac{h}{2},\ y_2 + \frac{k_2}{2}\right) = 0.1 \times f\left(.25,\ 1.1678 + \frac{.0656}{2}\right)$$

$$= 0.1 \times f(.25,\ 1.2006) = 0.1 \times \frac{1.2006 - .25}{1.2006 + .25} = 0.0655$$

$$k_4 = 0.1 \times f(x_2 + h,\ y_2 + k_3) = 0.1 \times f(.2 + .1,\ 1.678 + .0655)$$

$$= 0.1 \times f(.3,\ 1.2333) = 0.1 \times \frac{1.2333 - .3}{1.2333 + .3} = 0.0609$$

Hence, $y_3 = y(0.3) = y_2 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$$= 1.1678 + \frac{1}{6}(0.0708 + 2 \times .0656 + 2 \times .0655 + .0609)$$

$$= 1.2334$$

11.6.1. Errors in Solution of Ordinary Differential Equation

While solving the ordinary differential equation of the type $\frac{dy}{dx} = f(x, y)$, $y(x0) = y0$ by numerical methods, there arise two fundamental sources of error, viz., the truncation error and the round-off error.

Error in Euler's Method

Euler method provides a simple procedure for computing approximation y1 to the exact solution $y(x_1)$. The error in the approximation is the difference between the exact solution at $x = x_1$ and the solution value y_1 determined from $y_1 = y_0 + hy_0'$, using exact arithmetic and is called the local truncation error (E). Hence,

$$E = y_{exact} - y_{approx}$$

Now,

$$E_1 = y(x_1) - y_1$$

$$= y(x_0 + h) - (y_0 + hy_0')$$

$$= y_0 + hy_0' + \frac{h^2}{2!}y''(c) - (y_0 + hy_0')$$

(considering up to h^2 term in Taylor's series expansion)

$$= \frac{h^2}{2}y''(c), \text{ where } x_0 < c < x_0 + h$$

Thus, the order of the local truncation error is $O(h^2)$.

Error in Modified Euler's Method

It can be shown that the local truncation error in modified Euler's formula is proportional to h^3 instead of h^2 as in the case of Euler's method. This fact shows that the modified Euler formula does represent an improvement over the simple Euler formula.

Error in Runge-Kutta Method

Runge-Kutta second order formula has an error of the order of h3 and fourth order formula has an error of the order of h^5.

REVIEW QUESTIONS SET

1. Use Picard's method to solve the differential equation $\frac{dy}{dx} = x + y$; $y(0) = 1.0$ at $x = 0.1$. Correct up to four decimal places.
2. Use Picard's method to obtain correct to four decimal places of the solution of $\frac{dy}{dx} = x_2 + y_2$ at $x = 0.4$, given that $y(0) = 0$.
3. Use Picard's method to solve the differential equation $\frac{dy}{dx} = \frac{x^2}{y^2 + 1}$; $y(0) = 0$ at $x = 0.25$ and 0.50, correct to three decimal places.
4. Apply Picard's method to solve the differential equation $\frac{dy}{dx} = y - x$, $y(0) = 2$ at $x = 0.5$ and 1.0 correct to four decimal places.
5. Find an approximate value of y when $x = 0.1$ if $\frac{dy}{dx} = x - y_2$ with $y(0) = 1$, using Taylor's series method.
6. Given that $\frac{dy}{dx} = y - x$ and $y(0) = 2$. Use Taylor's series to obtain $y(0.1)$, correct to four decimal places.
7. Use Taylor's series method to solve the differential equation $\frac{dy}{dx} = x_2 + y_2$ for $x = 1.1$ with $y(1.0) = 0$.
8. Given that $\frac{dy}{dx} = 1 + y_2$, where $y(0) = 0$. Use the fourth order Runge-Kutta formula to find $y(0.2)$.
9. Use the fourth order Runge-Kutta formula to solve the differential equation $\frac{dy}{dx} = (1 - x)\, y2$ with $y(0) = 1.5$ to obtain y at $x = 0.1$ and 0.2.
10. Use Runge-Kutta method of fourth order to solve the equation $\frac{dy}{dx} = \frac{y}{x}$; $y(1) = 1$ to obtain y at $x = 1$ (0.1) 1.2.
11. (*a*) Use Runge-Kutta method to solve the differential equation $10\frac{dy}{dx} = x2 + y2$, $y(0) = 1$ for y in the interval $0 < x \leq 0.2$ with $h = 0.1$.
    (b) Solve the following differential equation by fourth-order Runge-Kutta method with initial conditions $x_0 = 0$ and $y_0 = 1.5$, $h = 0.1$ at $x = 0.3$:
    $$\frac{dy}{dx} = (1 - x)\, y_2$$ (AMIE, W '97)
12. Solve the following differential equation by Euler's method over the range $0 \leq x \leq 2.0$, using a step length $h = 0.5$:
    $$\frac{dy}{dx} = -y$$
    with the initial condition $y(0) = 1$.
13. Find $y(4.4)$, by Euler's modified method, taking $h = 0.1$, from the differential equation $\frac{dy}{dx} = \frac{2 - y^2}{5x}$, $y = 1$, where $x = 4$. Correct to 5-significant figures.
14. Calculate $y(0.05)$, by Euler's modified method, from the differential equation $\frac{dy}{dx} = x + y$, with the initial condition x0 = 0, y0 = 1 correct to 6 significant figures.
15. Solve $\frac{dy}{dx} = xy$ with the initial condition $x_0 = 0$, $y_0 = 1$, for $x = 0$ (0.1) 0.4 using Euler's method.
16. Given that $\frac{dy}{dx} = \log(x + y)$ with the initial condition that $y = 1$, when $x = 0$, find y for $x = 0.2$ correct up to four decimal places, using modified Euler's method.

17. Using modified Euler's method, find an approximate value of y at $x = 0.3$ given that $\frac{dy}{dx} = x + y$ and $y(0) = 1$. The result should be correct up to three decimal places.
18. Using modified Euler's method, find an approximate value of y at $x = 0.2$ given that $\frac{dy}{dx} = y$ and $y = 1$ at $x = 0$. The result should be correct up to three decimal places.
19. Obtain y at $x = 0.2$ in two steps of 0.1 each by modified Euler's method of the equation $\frac{dy}{dx} = -xy2$ given that $y(0) = 2$.
20. Solve the following differential equation by Euler's method over the interval [0, 1] with step size h = 0.1:

$$\frac{dy}{dx} = 2x + y$$

with initial condition $y(0) = 3$. (AMIE, S '97)

ANSWERS TO REVIEW QUESTION'S SET

1. 1.1102 2. 0.0214 3. 0.005, 0.042 4. 3.149, 4.718 5. 0.9138 6. 2.2052
7. 0.1107 8. 0.2027 9. 1.7492, 2.0547 10. 1.1, 1.2 11. (*a*) 1.0101, 1.0207, (*b*) 2.429
12. 0.5, 0.25, 0.125, 0.0625 13. 1.0187 14. 1.05256
15. (0, 1), (.1, .1), (.2, 1.01), (.3, 1.0302), (4, 1.0611)16. 1.0082 17. 1.3997 18. 1.222
19. 1.9237

20.

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	3	3.3	3.65	4.055	4.521	5.053	5.658	6.344	7.118	7.99	8.969

11.7. MATRIX INVERSION

A square matrix A is said to be the inverse of the square matrix B if $AB = BA = I$, I being the unit matrix. In order the product AB and BA are both defined and are equal, A and B should both be the square matrices of the same order. Thus, non-square matrices cannot have an inverse. It is to be noted that a matrix A can be inverted iff it is non-singular i.e., det $A \neq 0$.

There are several methods to find out the inverse of a matrix. Few methods are:

1. Computing the adjoint of the matrix
2. Gaussian method
3. Gauss-Jordan method
4. By finding the characteristic equation of the matrix
5. Iterative method.

The fourth method is discussed in the section 'Eigenvalues and Eigenvectors'.

Inverse by Computing Adjoint of Matrix

The inverse of a square matrix A is given by

$$A^{-1} = \frac{\text{adj } A}{|A|}$$

This gives the following procedure for the evaluation of the inverse of a matrix:

(*i*) Find the value of det A. If det $A = 0$, inverse does not exist.

(*ii*) Take the matrix A and replace each element by its cofactor.

(*iii*)Transform the result to form the adjoint matrix A.

(*iv*)Find $\frac{\text{adj } A}{|A|}$, which is the inverse of A.

Example 11.48. *Find inverse of the matrix* $\begin{bmatrix} 1 & 1 & 3 \\ 1 & 3 & -3 \\ -2 & -4 & -4 \end{bmatrix}$.

Solution. Suppose $A = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 3 & -3 \\ -2 & -4 & -4 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$

then $|A| = \begin{vmatrix} 1 & 1 & 3 \\ 1 & 3 & -3 \\ -2 & -4 & -4 \end{vmatrix}$ operating R2 – R1, R3 + 2R1

$= \begin{vmatrix} 1 & 1 & 3 \\ 0 & 2 & -6 \\ 0 & -2 & 2 \end{vmatrix}$ expanding by first column

$= \begin{vmatrix} 2 & -6 \\ -2 & 2 \end{vmatrix} = -8$

Therefore, $|A| = -8 \neq 0$. Hence, inverse of the matrix A exists.

Suppose capital letters denote the cofactors of corresponding small letters in $|A|$. Then

$A_1 = \begin{vmatrix} 3 & -3 \\ -4 & -4 \end{vmatrix} = -24$ $\quad B_1 = \begin{vmatrix} 1 & -3 \\ -2 & -4 \end{vmatrix} = 10$ $\quad C_1 = \begin{vmatrix} 1 & 3 \\ -2 & -4 \end{vmatrix} = 2$

$A2 = -\begin{vmatrix} 1 & 3 \\ -4 & -4 \end{vmatrix} = -8$ $\quad B2 = \begin{vmatrix} 1 & 3 \\ -2 & -4 \end{vmatrix} = 2$ $\quad C2 = \begin{vmatrix} 1 & 1 \\ -2 & -4 \end{vmatrix} = 2$

$A3 = \begin{vmatrix} 1 & 3 \\ 3 & -3 \end{vmatrix} = -12$ $\quad B3 = -\begin{vmatrix} 1 & 3 \\ 1 & -3 \end{vmatrix} = 6$ $\quad C3 = \begin{vmatrix} 1 & 1 \\ 1 & 3 \end{vmatrix} = 2$

adj A = transpose of $\begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & B_3 \\ C_1 & C_2 & C_3 \end{bmatrix}$

$= \begin{bmatrix} -24 & -8 & -12 \\ 10 & 2 & 6 \\ 2 & 2 & 2 \end{bmatrix}$

Hence, $A^{-1} = \frac{1}{|A|} \text{adj } A = -\frac{1}{8}\begin{bmatrix} -24 & -8 & -12 \\ 10 & 2 & 6 \\ 2 & 2 & 2 \end{bmatrix}$

$= \begin{bmatrix} 3 & 1 & 3/2 \\ -5/4 & -1/4 & -3/4 \\ -1/4 & -1/4 & -1/4 \end{bmatrix}$

Example 11.49. *If* $A = \begin{bmatrix} 3 & -3 & 4 \\ 2 & -3 & 4 \\ 0 & -1 & 1 \end{bmatrix}$, find A–1.

Solution. $A = \begin{bmatrix} 3 & -3 & 4 \\ 2 & -3 & 4 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$

$$|A| = \begin{vmatrix} 3 & -3 & 4 \\ 2 & -3 & 4 \\ 0 & -1 & 1 \end{vmatrix} \text{ operating } C_2 + C_3$$

$$= \begin{vmatrix} 3 & 1 & 4 \\ 2 & 1 & 4 \\ 0 & 0 & 1 \end{vmatrix} \text{ expanding by third row}$$

$$= \begin{vmatrix} 3 & 1 \\ 2 & 1 \end{vmatrix} = 1$$

Hence, $|A| = 1 \neq 0$. Therefore, A^{-1} exists. Now,

$$A_1 = \begin{vmatrix} -3 & 4 \\ -1 & 1 \end{vmatrix} = 1 \qquad B_1 = -\begin{vmatrix} 2 & 4 \\ 0 & 1 \end{vmatrix} = -2 \qquad C_1 = \begin{vmatrix} 2 & -3 \\ 0 & -1 \end{vmatrix} = -2$$

$$A2 = \begin{vmatrix} -3 & 4 \\ -1 & 1 \end{vmatrix} = -1 \qquad B2 = -\begin{vmatrix} 3 & 4 \\ 0 & 1 \end{vmatrix} = 3 \qquad C2 = -\begin{vmatrix} 3 & -3 \\ 0 & -1 \end{vmatrix} = 3$$

$$A3 = \begin{vmatrix} -3 & 4 \\ -3 & 4 \end{vmatrix} = 0 \qquad B3 = -\begin{vmatrix} 3 & 4 \\ 2 & 4 \end{vmatrix} = -4 \qquad C3 = \begin{vmatrix} 3 & -3 \\ 2 & -3 \end{vmatrix} = -3$$

$$\text{Hence, adj } A = \text{transpose of} \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & B_3 \\ C_1 & C_2 & C_3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 & 0 \\ -2 & 3 & -4 \\ -2 & 3 & -3 \end{bmatrix}$$

$$\text{Now,} \qquad A^{-1} = \frac{1}{|A|} \text{adj } A = \begin{bmatrix} 1 & -1 & 0 \\ -2 & 3 & -4 \\ -2 & 3 & -3 \end{bmatrix} \text{ (since } |A| = 1)$$

Gaussian Method

This method consists of the following steps:

1. Take a unit matrix I of the same order as that of the given matrix A.
2. Append I to A row-wise.
3. Starting from the top row A, successively divide the row by its diagonal element, leaving a_1 on the diagonal, and then subtract a multiple of the row from each row below it leaving zeros in the column below the diagonal element 1 to transform the matrix A to upper triangular matrix. This is known as forward elimination. In the elimination process if $a_{kk}^{k-1} = 0$ and $a_{lk}^{k-1} \neq 0$ for some $l > k$, then interchange the kth and lth rows, to get the non-zero diagonal element.
4. Subtract multiples of lower rows from the higher rows, leaving zeros in all positions but those on the diagonal which contain 1's. This is known as *backward elimination.*

Each operation done on A is also performed on the appended unit matrix I.

Hence, $[A : I] \xrightarrow{\text{Forward elimination}} [U : B] \xrightarrow{\text{Backward elimination}} [I : A^{-1}]$

i.e., when finally A reduces to a unit matrix, the appended unit matrix will be transformed into the inverse of A, namely A^{-1}.

Example 11.50. Find the inverse of $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix}$ *by Gaussian elimination.*

Solution. $[A : I] = \begin{bmatrix} 1 & 1 & 1 & . & 1 & 0 & 0 \\ 1 & 2 & 3 & . & 0 & 1 & 0 \\ 1 & 4 & 9 & . & 0 & 0 & 1 \end{bmatrix}$ ...(1)

Forward elimination: operating R2 – R1, R3 – R1 on (1)

$$\sim \begin{bmatrix} 1 & 1 & 1 & . & 1 & 0 & 0 \\ 0 & 1 & 2 & . & -1 & 1 & 0 \\ 0 & 3 & 8 & . & -1 & 0 & 1 \end{bmatrix} \text{ Operating R3 – 3R2}$$

$$\sim \begin{bmatrix} 1 & 1 & 1 & . & 1 & 0 & 0 \\ 0 & 1 & 2 & . & -1 & 1 & 0 \\ 0 & 0 & 2 & . & 2 & -3 & 1 \end{bmatrix} \text{ Operating } \frac{1}{2} \text{ R3}$$

$$\sim \begin{bmatrix} 1 & 1 & 1 & . & 1 & 0 & 0 \\ 0 & 1 & 2 & . & -1 & 1 & 0 \\ 0 & 0 & 1 & . & 1 & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \quad ...(2)$$

Backward elimination: operating R2 – 2R3, R1 – R3 on (2)

$$\sim \begin{bmatrix} 1 & 1 & 0 & . & 0 & \frac{3}{2} & -\frac{1}{2} \\ 0 & 1 & 0 & . & -3 & 4 & -1 \\ 0 & 0 & 1 & . & 1 & \frac{3}{2} & \frac{1}{2} \end{bmatrix} \text{ Operating R1 – R2}$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & . & 3 & \frac{5}{2} & \frac{1}{2} \\ 0 & 1 & 0 & . & -3 & 4 & -1 \\ 0 & 0 & 1 & . & 1 & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \text{ which is of the form } [I : A^{-1}]$$

Hence, $A^{-1} = \begin{bmatrix} 3 & -\frac{5}{2} & \frac{1}{2} \\ -3 & 4 & -1 \\ 1 & -\frac{3}{2} & \frac{1}{2} \end{bmatrix}$

Example 11.51. *Using Gauss elimination method, determine the inverse of the matrix*

$$\begin{bmatrix} 1 & -1 & 1 \\ 5 & -4 & 3 \\ 2 & 1 & 1 \end{bmatrix}$$ **(AMIE, S '94)**

Solution. Let $A = \begin{bmatrix} 1 & -1 & 1 \\ 5 & -4 & 3 \\ 2 & 1 & 1 \end{bmatrix}$

Now, $A = AI = \begin{bmatrix} 1 & -1 & 1 & . & 1 & 0 & 0 \\ 5 & -4 & 3 & . & 0 & 1 & 0 \\ 2 & 1 & 1 & . & 0 & 0 & 1 \end{bmatrix}$ ...(1)

Forward elimination : operating $R_2 - 5R_1$, $R_3 - 2R_1$ on (1)

$$\sim \begin{bmatrix} 1 & -1 & 1 & \cdot & 1 & 0 & 0 \\ 0 & 1 & -2 & \cdot & -5 & 1 & 0 \\ 0 & 3 & -1 & \cdot & -2 & 0 & 1 \end{bmatrix} \text{Operating } R_3 - 3R_2$$

$$\sim \begin{bmatrix} 1 & -1 & 1 & \cdot & 1 & 0 & 0 \\ 0 & 1 & -2 & \cdot & -5 & 1 & 0 \\ 0 & 0 & 5 & \cdot & 13 & -3 & 1 \end{bmatrix} \text{Operating } \frac{1}{5} R_3$$

$$\sim \begin{bmatrix} 1 & -1 & 1 & \cdot & 1 & 0 & 0 \\ 0 & 1 & -2 & \cdot & -5 & 1 & 0 \\ 0 & 0 & 1 & \cdot & \frac{13}{5} & -\frac{3}{5} & \frac{1}{5} \end{bmatrix} \quad \ldots(2)$$

Backward elimination: operating $R_2 + 2R_3$, $R_1 - R_3$ on (2)

$$\sim \begin{bmatrix} 1 & -1 & 0 & \cdot & -8/5 & 3/5 & -1/5 \\ 0 & 1 & 0 & \cdot & 1/5 & -1/5 & 2/5 \\ 0 & 0 & 1 & \cdot & 13/5 & -3/5 & 1/5 \end{bmatrix} \text{Operating R1 + R2}$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & \cdot & -7/5 & -2/5 & 1/5 \\ 0 & 1 & 0 & \cdot & 1/5 & -2/5 & 2/5 \\ 0 & 0 & 1 & \cdot & 13/5 & -3/5 & 1/5 \end{bmatrix}$$

Hence, $$A^{-1} = \begin{bmatrix} -7/5 & 2/5 & 1/5 \\ 1/5 & -1/5 & 2/5 \\ 13/5 & -3/5 & 1/5 \end{bmatrix}$$

Gauss Jordan Method

This method differs from Gaussian method only because elements above the diagonal elements are eliminated at the same time as those below, thus making back substitution unnecessary.

Example 11.52. Find the inverse of A, where

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & -1 \\ 2 & -1 & 3 \end{bmatrix}, \text{ by Gauss Jordan method.}$$

Solution. $$[A : I] = \begin{bmatrix} 1 & 2 & 1 & \cdot & 1 & 0 & 0 \\ 2 & 3 & -1 & \cdot & 0 & 1 & 0 \\ 2 & -1 & 3 & \cdot & 0 & 0 & 1 \end{bmatrix} \text{Operating } R_2 - 2R_1, R_3 - 2R_1$$

$$\sim \begin{bmatrix} 1 & 2 & 1 & \cdot & 1 & 0 & 0 \\ 0 & -1 & -3 & \cdot & -2 & 1 & 0 \\ 0 & -5 & 1 & \cdot & -2 & 0 & 1 \end{bmatrix} \text{Operating } (-1)\, R_2$$

$$\sim \begin{bmatrix} 1 & 2 & 1 & \cdot & 1 & 0 & 0 \\ 0 & 1 & 3 & \cdot & 2 & -1 & 0 \\ 0 & -5 & 1 & \cdot & -2 & 0 & 1 \end{bmatrix} \text{Operating } R_3 + 5R_2 \text{ and } R_1 - 2R_2$$

$$\sim \begin{bmatrix} 1 & 0 & -5 & \cdot & -3 & 2 & 0 \\ 0 & 1 & 3 & \cdot & 2 & -1 & 0 \\ 0 & 0 & 16 & \cdot & 8 & -5 & 1 \end{bmatrix} \text{Operating } \frac{1}{16} R_3$$

$$\sim \begin{bmatrix} 1 & 0 & -5 & \cdot & -3 & 2 & 0 \\ 0 & 1 & 3 & \cdot & 2 & -1 & 0 \\ 0 & 0 & 1 & \cdot & 1/2 & -5/16 & 1/16 \end{bmatrix} \text{Operating } R_2 - 3R_3, R_1 + 5R_3$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & \cdot & -1/2 & 7/16 & 5/16 \\ 0 & 1 & 0 & \cdot & 1/2 & 1/16 & -3/16 \\ 0 & 0 & 1 & \cdot & 1/2 & -5/16 & 1/16 \end{bmatrix}$$

Hence, $A^{-1} = \begin{bmatrix} -1/2 & 7/16 & 5/16 \\ 1/2 & -1/16 & 3/16 \\ 1/2 & -5/16 & 1/16 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} -8 & 7 & 5 \\ 8 & -1 & -3 \\ 8 & -5 & 1 \end{bmatrix}$

Iterative Method

Suppose the approximate inverse is B of matrix A. The error matrix E is given by

$$E = AB - I$$

or $$AB = I + E$$

or $$(AB)^{-1} = (I + E)^{-1}$$

or $$B^{-1} A^{-1} = (I + E)^{-1}$$

or $$A^{-1} = B(I + E)-1$$

or $$A^{-1} = B(I - E + E2 - ....) \quad ...(1)$$

If $\| E \| < 1$, approximating (1),

$$A^{-1} \approx B(I - E) = B(2I - AB)$$

Thus, $$A^{-1} = B\,(2I - AB)$$

This gives an iterative method

$$B^{(k+1)} = B^{(k)}\,(2I - AB^{(k)}),\ k = 0, 1, 2, ...$$

Example 11.38. *Find the inverse of the matrix*

$$A = \begin{bmatrix} 5 & 2 \\ 3 & -1 \end{bmatrix}$$

using the iterative method, given that its approximate inverse is $\begin{bmatrix} 0.1 & 0.2 \\ 0.3 & -0.4 \end{bmatrix}$. *Perform two iterations.*

Solution. Here $B^{(0)} = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & -0.4 \end{bmatrix}$, the initial approximate inverse of A.

From iterative formula, B(k+1) = B(k) (2I – AB(k))

$$B^{(1)} = B(0)\,(2I - AB(0))$$

$$= \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & -0.4 \end{bmatrix} \times \left(\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} - \begin{bmatrix} 5 & 2 \\ 3 & -1 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & -0.4 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & -0.4 \end{bmatrix} \times \left(\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} - \begin{bmatrix} 1.1 & 0.2 \\ 0 & 1.0 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & -0.4 \end{bmatrix} \times \left(\begin{bmatrix} .9 & -0.2 \\ 0 & 1.0 \end{bmatrix} \right) = \begin{bmatrix} 0.09 & 0.18 \\ 0.27 & -0.46 \end{bmatrix}$$

$$B^{(2)} = B(1)\,(2I - AB(1))$$

$$= \begin{bmatrix} 0.09 & 0.18 \\ 0.27 & 0.46 \end{bmatrix} \times \left(\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} - \begin{bmatrix} 5 & 2 \\ 3 & -1 \end{bmatrix} \times \begin{bmatrix} 0.09 & 0.18 \\ 0.27 & -0.46 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0.09 & 0.18 \\ 0.27 & 0.46 \end{bmatrix} \times \left(\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} - \begin{bmatrix} .99 & -.02 \\ 0 & 1.0 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0.09 & 0.18 \\ 0.27 & 0.46 \end{bmatrix} \times \begin{bmatrix} 1.01 & .02 \\ 0 & 1.0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.0909 & 0.1818 \\ 0.2727 & -0.4546 \end{bmatrix}$$

11.8. SOLUTIONS OF LINEAR SIMULTANEOUS EQUATIONS

In many engineering and scientific applications, it is often required to solve a system of simultaneous linear equations. The general form of linear simultaneous n equations with n unknowns is of the form

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n &= b_2 \\ \ldots\ldots\ldots\

1. The methods of first type are called elimination methods, reduction methods, or direct methods. In this section, we will consider the following methods:
   (*a*) Gauss elimination method
   (*b*) Gauss-Jordan method, and
   (*c*) Matrix inversion method.
2. The methods of second type are called indirect or iterative methods. In this section, we will consider the following methods:
   (*a*) Gauss-Siedel method, and
   (*b*) Jacobi's method.

11.8.1. Direct Methods

Gauss Elimination Method: The Gauss elimination method is a systematic process of eliminating variables from the linear equations. This method is divided into two parts: (a) Triangularization, (b) Back substitution.

The variable x_1 in (1) is eliminated in each equation, i = 2, 3, ..., n, by subtracting a multiple $m_{i1} = a_{i1}/a_{11}$ of the first equation from it. This gives the first derived set as

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots & \\ a_{n2}^{(1)}x_2 + \ldots + a_{nn}^{(1)}x_n &= b_n^{(1)} \end{aligned}$$

The variable x2 is now eliminated from each equation i = 3, ..., n by subtracting a multiple mi2 = $a_{i2}^{(1)}/a_{22}^{(1)}$ of the second row from it. After n – 1 such steps, we obtain an equivalent derived system of the following form:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n &= b_3^{(2)} \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots & \\ a_{nn}^{(n-1)}x_n &= b_n^{(n-1)} \end{aligned}$$

or, in matrix form, $Ux = b^{(n-1)}$, where U is *upper triangular*. This triangular set may now be solved by *back substitution*, computing x_n from the nth equation, x_{n-1} from the $(n-1)$th equation, ..., x_1 from the first.

At each step in the above triangularisation process, we assumed that the term a_{kk}, called the pivot which is used to eliminate x_k from the rows $(k+1)$ to n, is non-zero. Thus, it is required to further verify that the pivots akk are non-zero. If $a_{kk}^{(k-1)} = 0$ and $a_{ik}^{(k-1)} \neq 0$ for some l > k, then interchanging the k-th and l-th rows of the matrix [A, b](k–1), we obtain a non-zero pivot element. But if $a_{kk}^{(k-1)} = a_{k+1,k}^{(k-1)} = \ldots\ldots = a_{nk}^{(k-1)} = 0$, then it means that A is singular.

In order to minimize the rounding errors, we should avoid dividing by small pivots because it results in larger error propagation. For this reason, if $a_{kk}^{(k-1)} \neq 0$, then to find the largest pivot, there exists two well-known strategies known as partial pivoting and complete pivoting.

In partial pivoting, choose the pivot element $a_{rk}^{(k-1)}$ such that

$$a_{rk}^{(k-1)} = \max_{k \le i \le n} |a_{ik}^{(k-1)}|$$

Then interchange the rows k and r, and continue the *k*-th step of the elimination process.

In complete pivoting, choose the element $a_{rm}^{(k-1)}$ as the pivot element, where

$$a_{rm}^{(k-1)} = \max_{k \le i, j \le n} |a_{ij}^{(k-1)}|$$

Now, interchange the rows *r* and *k*, columns *m* and *k* and continue the *k*-th step of the elimination process. Note that interchanging of columns means the order of the unknowns is changed. Complete pivoting is safe as errors are never magnified unreasonably.

After $[\mathbf{A}, \mathbf{b}]^{(k-1)}$ has been constructed and applying the pivoting strategy, the non-zero elements of $[\mathbf{A}, \mathbf{b}]^{(k)}$ are computed using the formula:

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - (a_{ik}^{(k-1)} / a_{kk}^{(k-1)})\, a_{kj}^{(k-1)} \quad \begin{cases} k = 1, \ldots, (n-1) \\ i = (k+1), \ldots, n \\ j = (k+1), \ldots, (n+1) \end{cases}$$

where $a_{ij}^{(0)} = a_{ij}$ are the coefficients, of the original matrix *A*.

Example 11.54. Solve the following set of simultaneous equations using Gaussian elimination method:

$$3x - y + 5z = 16$$
$$x - 3y + 2z = 1$$
$$4x + y + z = 9$$

(AMIE, W '93)

Solution. The augmented matrix $[A : b] = \begin{bmatrix} 3 & -1 & 5 & 16 \\ 1 & -3 & 2 & 1 \\ 4 & 1 & 1 & 9 \end{bmatrix}$. To eliminate x from second and third rows, multiply the first row successively by $m_{21} = \frac{a_{21}}{a_{11}} = \frac{1}{3}$ and $m_{31} = \frac{a_{31}}{a_{11}} = \frac{4}{3}$ and subtract from second and third rows, respectively, i.e., operating $R_2 - \frac{1}{3} R_1$ and $R_3 - \frac{4}{3} R_1$,

$$\begin{bmatrix} 3 & -1 & 5 & 16 \\ 0 & -\frac{8}{3} & \frac{1}{3} & -\frac{13}{3} \\ 0 & \frac{7}{3} & -\frac{17}{3} & -\frac{37}{3} \end{bmatrix}$$

$$m_{32} = \frac{a_{32}^{(1)}}{a_{22}^{(1)}} = \frac{\frac{7}{3}}{-\frac{8}{3}} = -\frac{7}{8}$$

Operating $R_3 - \left(-\frac{7}{8}\right) R_2$ to eliminate y from third row,

$$\sim \begin{bmatrix} 3 & -1 & 5 & 16 \\ 0 & -\frac{8}{3} & \frac{1}{3} & -\frac{13}{3} \\ 0 & 0 & -\frac{129}{24} & -\frac{387}{24} \end{bmatrix}$$

Performing back substitution,

$$-\frac{129}{24}z = -\frac{387}{24} \qquad \text{(from third row)}$$

$$\Rightarrow z = 3$$

$$-\frac{8}{3}y + \frac{1}{3}z = -\frac{13}{3} \qquad \text{(from second row)}$$

or $$y = \frac{\left(-\frac{13}{3} - \frac{1}{3} \times 3\right)}{(-8/3)} = 2$$

and $3x - y + 5z = 16$ (from first row)

or $x = (16 + 2 - 5 \times 3)/3 = 1$

The required solutions are $x = 1$, $y = 2$, $z = 3$.

Example 11.40. *Find the solution of the following system of equations*

$$2x + 8y + 2z = 14$$
$$x + 6y - z = 13$$
$$2x - y + 2z = 5$$

using the Gauss elimination method.

Solution. The augmented matrix $[A : b] = \begin{bmatrix} 2 & 8 & 2 & 14 \\ 1 & 6 & -1 & 13 \\ 2 & -1 & 2 & 5 \end{bmatrix}$

Here $m_{21} = \frac{a_{21}}{a_{11}} = \frac{1}{2}$ and $m_{31} = \frac{a_{31}}{a_{11}} = \frac{2}{2} = 1$

Operating $R_2 - \frac{1}{2} R_1$ and $R_3 - R_1$ on the augmented matrix,

$$[A : b] \sim \begin{bmatrix} 2 & 8 & 2 & 14 \\ 0 & 2 & -2 & 6 \\ 0 & 9 & 0 & -9 \end{bmatrix} \qquad m_{32} = \frac{a_{32}^{(1)}}{a_{22}^{(1)}} = \frac{9}{2} \text{ and operating } R_3 - \frac{9}{2}R2$$

$$\sim \begin{bmatrix} 2 & 8 & 2 & 14 \\ 0 & 2 & -2 & 6 \\ 0 & 0 & -9 & 18 \end{bmatrix}$$

Finally, through back substitution, beginning with the third row, the following values are obtained:

$$-9z = 18 \Rightarrow z = -2$$

$$y = (6 + 2z)/2 = (6 - 4)/2 = 1$$

and $$x = (14 - 2z - 8y)/2 = (14 - 2 \times -2 - 8 \times 1)/2 = 5$$

Hence, the required solutions are $x = 5$, $y = 1$ and $z = -2$.

Algorithm: Gaussian elimination with partial pivoti

1. For $n \times n$ coefficient matrix, construct an augment matrix of order $n \times (n + 1)$.
2. (Perform partial pivoting for (1, 1) position). Int change rows, if necessary, to make the value of a_{11}, the largest magnitude of any coefficient in the first column; i.e., scan the column of the matrix from the diagonal down in order to find the element that is largest in magnitude and bring it to (1, 1) position.
3. Eliminate x_1 in equations – rows 2 to n.
4. Repeat the steps 2 and 3 for the second row through $(n - 1)$ rows.
5. Solve xn from the nth equation.
6. Solve x_i, $i = n - 1, n - 2, \ldots 1$ using back substitution.

Example 11.56. Find the solution of the following system of equations:

$$\begin{aligned} 5x_1 + 3x_2 - x_3 &= 11 \\ 2x_1 + 4x_3 + x_4 &= 1 \\ -3x_1 + 3x_2 - 3x_3 + 5x_4 &= -2 \\ 6x2 - 2x_3 + 3x_4 &= 9 \end{aligned}$$

by Gaussian elimination with partial pivoting.

Solution. The augmented matrix $[A : b] = \begin{bmatrix} (5) & 3 & -1 & 0 & 11 \\ 2 & 0 & 4 & 1 & 1 \\ -3 & 3 & -3 & 5 & -2 \\ 0 & 6 & -2 & 3 & 9 \end{bmatrix}$

Here 5 is the largest coefficient in the first column. Then

$$m_{21} = \frac{a_{21}}{a_{11}} = \frac{2}{5} = 0.4\ ,\ m_{31} = \frac{a_{31}}{a_{11}} = \frac{-3}{5} = -0.6$$

and $$m_{41} = \frac{a_{41}}{a_{11}} = \frac{0}{5} = 0.0$$

Operating $R_2 - (0.4)\ R_1$, $R_3 - (-0.6)\ R_1$ and $R_4 - (0)R_1$ on the augmented matrix,

$$\sim \begin{bmatrix} 5 & 3 & -1 & 0 & 11 \\ 0 & -1.2 & 4.4 & 1.0 & -3.4 \\ 0 & 4.8 & -3.6 & 5.0 & 4.6 \\ 0 & 6.0 & -2.0 & 3.0 & 9.0 \end{bmatrix}$$

Since the numerically largest value below the second diagonal element is 6.0 and belongs to the 4th row, we interchange 2nd and 4th row,

$$\sim \begin{bmatrix} 5 & 3 & -1 & 0 & 11 \\ 0 & (6.0) & -2.0 & 3.0 & 9.0 \\ 0 & 4.8 & -3.6 & 5.0 & 4.6 \\ 0 & -1.2 & 4.4 & 1.0 & -3.4 \end{bmatrix}$$

Now, $m_{32} = \frac{a_{32}^{(1)}}{a_{22}^{(1)}} = \frac{4.8}{6.0} = 0.8$ and $m_{42} = \frac{a_{42}^{(2)}}{a_{22}^{(1)}} = \frac{-1.2}{6.0} = -0.2$. Operating $R_3 - (0.8)\ R_2$ and $R_4 - (-0.2)R_2$,

$$\sim \begin{bmatrix} 5 & 3 & -1 & 0 & 11 \\ 0 & 6.0 & -2.0 & 3.0 & 9.0 \\ 0 & 0 & -2.0 & 2.6 & -2.6 \\ 0 & 0 & 4.0 & 1.6 & -1.6 \end{bmatrix}$$

Since the numerically largest value below the third diagonal element is 4.0, interchanging 3rd and 4th row,

$$\sim \begin{bmatrix} 5 & 3 & -1 & 0 & 11 \\ 0 & 6.0 & -2.0 & 3.0 & 9.0 \\ 0 & 0 & (4.0) & 1.6 & -1.6 \\ 0 & 0 & -2.0 & 2.6 & -2.6 \end{bmatrix},\quad m_{43} = \frac{-2.0}{4.0} = -0.5$$

Operating $R_4 - (-0.5)\ R_3$

$$\sim \begin{bmatrix} 5 & 3 & -1 & 0 & 11 \\ 0 & 6.0 & -2.0 & 3.0 & 9.0 \\ 0 & 0 & 4.0 & 1.6 & -1.6 \\ 0 & 0 & 0 & 3.4 & -3.4 \end{bmatrix}$$

The values of the variables are directly obtained by using the back substitution.

$$x_4 = -3.4/3.4 = -1.0$$

$$x_3 = (-1.6 - 1.6 \times (-1.0))/4.0 = 0.0$$

$$x_2 = (9.0 + 2.0 \times 0.0 - 3.0 \times (-1.0))/6.0 = 2.0$$
$$x_1 = (11 - 3 \times 2.0 + 1 \times 0 - 0 \times (-1.0))/5.0 = 1.0$$

Gauss-Jordan Method

This method differs from the Gaussian scheme because elements above the diagonal elements are eliminated at the same time as those below, thus making back substitution unnecessary. Usually the diagonal elements are made equal to one at the same time the reduction is performed. Pivoting can be employed to preserve accuracy.

Example 11.42. Solve the following system of equations using Gauss-Jordan elimination method, clearly indicating each step.

$$\begin{aligned} x_1 + 2x_2 + x_3 + 4x_4 &= 13 \\ 2x_1 + 4x_3 + 3x_4 &= 28 \\ 4x_1 + 2x_2 + 2x_3 + x_4 &= 20 \\ -3x_1 + x_2 + 3x_3 + 2x_4 &= 6 \end{aligned}$$ (AMIE, S'97)

Solution. The augmented matrix [A : b] is

$$\begin{bmatrix} 1 & 2 & 1 & 4 & 13 \\ 2 & 0 & 4 & 3 & 28 \\ 4 & 2 & 2 & 1 & 20 \\ -3 & 1 & 3 & 2 & 6 \end{bmatrix}$$ Operating $R_2 - 2R_1$, $R_3 - 4R_1$ and $R_2 + 3R_1$

$$\sim \begin{bmatrix} 1 & 2 & 1 & 4 & 13 \\ 0 & -4 & 2 & -5 & 2 \\ 0 & -6 & -2 & -15 & -32 \\ 0 & 7 & 6 & 14 & 45 \end{bmatrix}$$ Operating $-\frac{1}{4} R_2$

$$\sim \begin{bmatrix} 1 & 2 & 1 & 4 & 13 \\ 0 & 1 & -.5 & 1.25 & -.5 \\ 0 & -6 & -2 & -15 & -32 \\ 0 & 7 & 6 & 14 & 45 \end{bmatrix}$$ Operating $R_1 - 2R_2$, $R_3 + 6R_2$ and $R_4 - 7R_2$

$$\sim \begin{bmatrix} 1 & 0 & 2 & 1.50 & 14 \\ 0 & 1 & -.5 & 1.25 & -.5 \\ 0 & 0 & -5 & -7.50 & -35 \\ 0 & 0 & 9.5 & 5.25 & 48.5 \end{bmatrix}$$ Operating $\left(-\frac{1}{5}\right) R_3$

$$\sim \begin{bmatrix} 1 & 0 & 2 & 1.50 & 14 \\ 0 & 1 & -.5 & 1.25 & -.5 \\ 0 & 0 & 1 & 1.50 & 7 \\ 0 & 0 & 9.5 & 5.25 & 48.5 \end{bmatrix}$$ Operating $R_1 - 2R_3$, $R_2 + .5R_3$ and $R_4 - 9.5R3$

$$\sim \begin{bmatrix} 1 & 0 & 0 & -1.50 & 0 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 0 & 1 & 1.50 & 7 \\ 0 & 0 & 0 & -9.00 & -18.00 \end{bmatrix}$$ Operating $\frac{1}{-9} R_4$

$$\sim \begin{bmatrix} 1 & 0 & 0 & -1.50 & 0 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 0 & 1 & 1.50 & 7 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$ Operating $R_1 + 1.50\, R_4$, $R_2 - 2R_4$ and $R_3 - 1.50\, R_4$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

Hence, the required solution is x1 = 3, x2 = –1, x3 = 4, x4 = 2.

Example 11.57. Solve by Gauss-Jordan elimination with partial pivoting:

$$x1 - x2 + 2x3 = -8$$
$$x1 + x2 + x3 = -2$$
$$2x1 - 2x2 + 3x3 = -20$$

Solution. The augmented matrix is

$$\begin{bmatrix} 1 & -1 & 2 & -8 \\ 1 & 1 & 1 & -2 \\ 2 & -2 & 3 & -20 \end{bmatrix}$$

Since 2 is the largest coefficient in the first column interchanging R_1 and R_3,

$$\sim \begin{bmatrix} 2 & -2 & 3 & -20 \\ 1 & 1 & 1 & -2 \\ 1 & -1 & 2 & -8 \end{bmatrix}$$ Operating $\frac{1}{2} R_1$

$$\sim \begin{bmatrix} 1 & -1 & 3/2 & -10 \\ 1 & 1 & 1 & -2 \\ 1 & -1 & 2 & -8 \end{bmatrix}$$ Operating $R_2 - R_1$ and $R_3 - R_1$

$$\sim \begin{bmatrix} 1 & -1 & 3/2 & -10 \\ 0 & 2 & -1/2 & 8 \\ 0 & 0 & 1/2 & 2 \end{bmatrix}$$ Operating $\frac{1}{2} R_2$

$$\sim \begin{bmatrix} 1 & -1 & 3/2 & -10 \\ 0 & 1 & -1/4 & 4 \\ 0 & 0 & 1/2 & 2 \end{bmatrix}$$ Operating $R_1 + R_2$

$$\sim \begin{bmatrix} 1 & 0 & 5/4 & -6 \\ 0 & 1 & -1/4 & 4 \\ 0 & 0 & 1/2 & 2 \end{bmatrix}$$ Operating $2R_3$

$$\sim \begin{bmatrix} 1 & 0 & 5/4 & -6 \\ 0 & 1 & -1/4 & 4 \\ 0 & 0 & 1 & 4 \end{bmatrix}$$ Operating $R_1 - \frac{5}{4} R_3$ and $R_2 + \frac{1}{4} R_3$

$$\sim \begin{bmatrix} 1 & 0 & 0 & -11 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 4 \end{bmatrix}$$

Hence, the required solution is x1 = –11, x2 = 5, x3 = 4.

Matrix Inversion Method

The system of linear equations can be written in matrix form as

$$AX = b \qquad \ldots(2)$$

where A is the coefficient matrix, X is the solution vector to be determined and b is called the right hand side vector.

Premultiplying (2) by A^{-1},

$$A^{-1}(AX) = A^{-1}b$$

or

$$(A^{-1}A)X = A^{-1}b$$

or $IX = A^{-1} b$

i.e., $X = A^{-1} b$

Hence, the following steps are to be followed to solve the system of linear equations by matrix inversion method.

1. Find the inverse of coefficient matrix, i.e., A^{-1} which can be determined by any of the methods described above.
2. Multiply A^{-1} with b, column matrix will be the solution vector.

Example 11.59. Solve the following equations using matrix inversion method:

$$3x + y + 2z = 3$$
$$2x - 3y - z = -3$$
$$x + 2y + z = 4$$

(AMIE, W '94)

Solution. Here $A = \begin{bmatrix} 3 & 1 & 2 \\ 2 & -3 & -1 \\ 1 & 2 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 3 \\ -3 \\ 4 \end{bmatrix}$ and $X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

Thus, $|A| = 3(-3 + 2) - 1(2 + 1) + 2(4 + 3) = 8 \neq 0$

$$A_1 = \begin{vmatrix} -3 & -1 \\ 2 & 1 \end{vmatrix} = -1, \quad A_2 = \begin{vmatrix} 1 & 2 \\ 2 & 1 \end{vmatrix} = 3, \quad A_3 = \begin{vmatrix} 1 & 2 \\ -3 & -1 \end{vmatrix} = 5$$

$$B_1 = -\begin{vmatrix} 2 & -1 \\ 1 & 1 \end{vmatrix} = -3, \quad B_2 = \begin{vmatrix} 3 & 2 \\ 1 & 1 \end{vmatrix} = 1, \quad B_3 = -\begin{vmatrix} 3 & 2 \\ 2 & -1 \end{vmatrix} = 7$$

$$C_1 = \begin{vmatrix} 2 & -3 \\ 1 & 2 \end{vmatrix} = 7, \quad C_2 = -\begin{vmatrix} 3 & 1 \\ 1 & 2 \end{vmatrix} = -5, \quad C_3 = \begin{vmatrix} 3 & 1 \\ 2 & -3 \end{vmatrix} = -11$$

Thus, $\text{Adj } A = \begin{bmatrix} -1 & -3 & 7 \\ 3 & 1 & -5 \\ 5 & 7 & -11 \end{bmatrix}^T = \begin{bmatrix} -1 & 3 & 5 \\ -3 & 1 & 7 \\ 7 & -5 & -11 \end{bmatrix}$

and $A^{-1} = \frac{1}{8}\begin{bmatrix} -1 & 3 & 5 \\ -3 & 1 & 7 \\ 7 & -5 & -11 \end{bmatrix}$

Hence, $X = A^{-1} B = \frac{1}{8}\begin{bmatrix} -1 & 3 & 5 \\ -3 & 1 & 7 \\ 7 & -5 & -11 \end{bmatrix} \times \begin{bmatrix} 3 \\ -3 \\ 4 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 8 \\ 16 \\ -8 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$

The required solution is x = 1, y = 2, z = –1.

11.8.2. Iterative Methods

In this method, the values of unknowns are improved by substituting directly the previous values of the unknowns in the equations. The procedure is continued till the convergence is assured (correct to required degree of accuracy). These methods are preferred over the direct method, particularly when the coefficient matrix is sparse, i.e., it has many zeros.

Iterative methods for linear simultaneous equations require presentation of the equations in such a way that each of the n equations contain one of the n unknowns on its left hand side. Thus n linear equations are to be written as

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1n}x_n)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3 - \ldots - a_{2n}x_n)/a_{22}$$
$$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$$
$$x_n = (b_n - a_{n1}x_1 - a_{n2}x_2 - \ldots - a_{nn-1}x_{n-1})/a_{nn}$$

Gauss-Siedel Method

The sequence of steps constituting the Gauss-Siedel method is as follows:

Step 1: Assign an initial value to each unknown. Usually we assign zero value to all unknowns if no initial estimates are available.

Step 2: Solve the first equation for x_1 by substituting the initial values of other unknowns.

Step 3: Solve the second equation for x_2 by substituting the current values of other unknowns.

Step 4: Proceed with the remaining equations until the final equation is solved.

This completes the first iteration. The number of iterations required depends upon the desired degree of accuracy. The iteration formula at (k + 1)th iteration is given below:

$$x_1^{(k+1)} = (b_1 - a_{12}\, x_2^{(k)} - a_{13}\, x_3^{(k)} - \ldots - a_{1n}\, x_n^{(k)})/a^{11}$$

$$x_2^{(k+1)}) = (b_2 - a_{21}\, x_1^{(k+1)} - a_{23} x_3^{(k)} - \ldots - a_{2n}\, x_n^{(k)})/a_{22}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$x_n^{(k+1)} = (b_n - a_{n1} x_{1(k+1)} - a_{n2} x_2^{(k+1)} - \ldots - a_{n\,n-1}\, x_{n-1}^{(k+1)}/a_{nn}$$

Remark 1: The Gauss-Siedel method will ensure the convergence if the matrix A is diagonally dominant, i.e.,

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad \text{for } i = 1, 2, \ldots, n$$

Remark 2: Termination of iteration if

$$|x_i(k+1) - x_i(k)| < \varepsilon_1$$

or

$$\frac{|x_i^{(k+1)} - x_i^{(k)}|}{|x_i^{(k+1)}|} < \varepsilon_2, \qquad i = 1, 2, \ldots, n$$

Example 11.60. *Solve the following system of equations by Gauss-Siedel iteration method:*

$$8x_1 + 2x_2 - 2x_3 = 8$$
$$x_1 - 8x_2 + 3x_3 = -4$$
$$2x_1 + x_2 + 9x_3 = 12$$

Correct up to two decimal places.

Solution. The given system is diagonally dominant. The Gauss-Siedel iteration scheme can be written as follows:

$$x_1^{(k+1)} = \frac{1}{8}\,[8 - 2x_2^{(k)} + 2x_3^{(k)}] \qquad \ldots(1)$$

$$x_2^{(k+1)} = \frac{1}{8}\,[4 + x_1^{(k+1)} + 3x_3^{(k)}] \qquad \ldots(2)$$

$$x_3^{(k+1)} = \frac{1}{9}\,[12 - 2x_1^{(k+1)} - x_2^{(k+1)}] \qquad \ldots(3)$$

We take the initial values as x1(0) = 0, x2(0) = 0, x3(0) = 0

First iteration: Using the initial values in (1), $x_1^{(1)}$ and $x_3^{(0)}$ in (2) and $x_1^{(1)}$ and $x_2^{(1)}$ in (3),

$$x_1^{(1)} = \frac{1}{8}\,[8 - 2 \times 0 + 2 \times 0] = 1$$

$$x_2^{(1)} = \frac{1}{8}\,[4 + 1 \times 1 + 3 \times 0] = 0.625$$

$$x_3(1) = \frac{1}{9}\,[12 - 2 \times 1 - 1 \times 0.625] = 1.0417$$

Second iteration: Using $x_2^{(1)}$ and $x_3^{(1)}$ in (1), $x_1^{(2)}$ and $x_3^{(1)}$, in (2) and $x_1^{(2)}$ and $x_2^{(2)}$ in (3),

$$x_1^{(2)} = \frac{1}{8}[8 - 2 \times 0.625 + 2 \times 1.0417] = 1.1042$$

$$x_2^{(2)} = \frac{1}{8}[4 + 1 \times 1.1042 + 3 \times 1.0417] = 1.0287$$

$$x_3^{(2)} = \frac{1}{9}[12 - 2 \times 1.1042 - 1 \times 1.0287] = 0.9736$$

Third iteration:

$$x_1^{(3)} = \frac{1}{8}[8 - 2 \times 1.0287 + 2 \times 0.9736] = 0.9862$$

$$x_2^{(3)} = \frac{1}{8}[4 + 1 \times 0.9862 + 3 \times 0.9736] = 0.9884$$

$$x_3^{(3)} = \frac{1}{9}[12 - 2 \times 0.9862 - 1 \times 0.9884] = 1.0044$$

Fourth iteration:

$$x_1^{(4)} = \frac{1}{8}[8 - 2 \times 0.9884 + 2 \times 1.0044] = 1.0040$$

$$x_2^{(4)} = \frac{1}{8}[4 + 1 \times 1.0040 + 3 \times 1.0044] = 1.0022$$

$$x_3^{(4)} = \frac{1}{9}[12 - 2 \times 1.0040 - 1 \times 1.0022] = 0.9986$$

Fifth iteration:

$$x_1^{(5)} = \frac{1}{8}[8 - 2 \times 1.0022 + 2 \times 0.9986] = 0.9991$$

$$x_2^{(5)} = \frac{1}{8}[4 + 1 \times 0.9991 + 3 \times 0.9986] = 0.9994$$

$$x_3^{(5)} = \frac{1}{9}[12 - 2 \times 0.9991 - 1 \times 0.9994] = 1.0003$$

The successive iterations computed above are shown below:

Computation Table

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	0	0	0
1	1	0.625	1.0417
2	1.1042	1.0287	0.9736
3	0.9862	0.9884	1.0044
4	1.0040	1.0022	0.9986
5	0.9991	0.9994	1.0003
⋮	⋮	⋮	⋮

Thus, $x_1 = 1.00$, $x_2 = 1.00$, $x_3 = 1.00$, correct up to two decimal places.

Jacobi Method

Jacobi's method is similar to Gauss-Siedel method, the only difference is that in Jacobi's method in each iteration, new values of xis are found using only the old values of xis, which are found in the just previous cycle. The iteration formula at $(k + 1)$th iteration is given below:

$$x_1(k+1) = (b_1 - a_{12}\, x_2^{(k)} - a_{13}x_3^{(k)} - \ldots - a_{1n}x_n^{(k)})/a_{11}$$
$$x_2(k+1) = (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \ldots - a_{2n}\, x_n^{(k)}/a_{22}$$
$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$
$$x_{n(k+1)} = (b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \ldots - a_{nn-1}\, x_{n-1}(k))/a_{nn}$$

Example **11.61.** *Solve by Jacobi method the system of equations*

$$4x + y + 3z = 17$$
$$x + 5y + z = 14$$
$$2x - y + 8z = 12$$

Compute up to fourth iteration.

Solution. Here the coefficient matrix is diagonally dominant as $|4| > |1| + |3|$, $|5| > |1| + |1|$ and $|8| > |2| + |-1|$. The Jacobi iteration formulae are:

$$x^{(k+1)} = \frac{1}{4}[17 - y(k) - 3z^{(k)}] \qquad \ldots(1)$$

$$y^{(k+1)} = \frac{1}{5}[14 - x^{(k)} - z^{(k)}] \qquad \ldots(2)$$

$$z^{(k+1)} = \frac{1}{8}[12 - 2x^{(k)} + y^{(k)}] \qquad \ldots(3)$$

We take the initial values as $x(0) = y(0) = z(0) = 0$.

First iteration : Using initial values

$$x^{(1)} = \frac{1}{4}[17 - 0 - 3(0)] = \frac{17}{4}$$

$$y^{(1)} = \frac{1}{5}[14 - 0 - 0] = \frac{14}{5}$$

$$z^{(1)} = \frac{1}{8}[12 - 2(0) + 0] = \frac{3}{2}$$

Second iteration Using the values of $x^{(1)}$, $y^{(1)}$, $z^{(1)}$ in (1), (2), (3),

$$x^{(2)} = \frac{1}{4}\left[17 - \frac{14}{5} - 3\cdot\frac{3}{2}\right] = \frac{97}{40} = 2.425$$

$$y^{(2)} = \frac{1}{5}\left[14 - \frac{17}{4} - \frac{3}{2}\right] = \frac{33}{20} = 1.65$$

$$z^{(2)} = \frac{1}{8}\left[12 - 2\cdot\frac{17}{4} + \frac{14}{5}\right] = \frac{63}{80} = 0.788$$

Third iteration:

$$x^{(3)} = \frac{1}{4}[17 - 1.65 - 3 \times .788] = 3.246$$

$$y^{(3)} = \frac{1}{5}[14 - 2.425 - .788] = 2.157$$

$$z^{(3)} = \frac{1}{8}[12 - 2 \times 2.425 + 1.65] = 1.145$$

Fourth iteration:

$$x(4) = \frac{1}{4}[17 - 2.157 - 3 \times 1.145] = 2.852$$

$$y(4) = \frac{1}{5}[14 - 3.246 - 1.145] = 1.922$$

$$z^{(4)} = \frac{1}{8}[12 - 2 \times 3.246 + 2.157] = 0.958$$

Thus, after fourth iteration, x = 2.852, y = 1.922, z = 0.958.

Ill-conditioning **(AMIE, W '95)**

A matrix A is called ill-conditioned if there exists a vector B such that for small changes in the elements of A or B will produce large change in X = A–1B, In this case, numerical methods for computing an approximate solution are prone to have more errors. Consider the two systems:

$$\left.\begin{aligned} x - y &= 1 \\ x - 1.00001\, y &= 0 \end{aligned}\right\} \quad ...(1)$$

and

$$\left.\begin{aligned} x - y &= 1 \\ x - .99999\, y &= 0 \end{aligned}\right\} \quad ...(2)$$

For system (1), $A = \begin{bmatrix} 1 & -1 \\ 1 & -1.00001 \end{bmatrix}$

and for system (2), $A = \begin{bmatrix} 1 & -1 \\ 1 & -.99999 \end{bmatrix}$

There is a small change in the elements of A.

The solution of (1) is $x = 100{,}001$, $y = 100{,}000$

and the solution of (2) is $x = -99{,}999$, $y = -100{,}000$

This shows that for small change in the coefficients of the variables, there is a large change in the solution. Hence, the system is ill-conditioned.

11.9. EIGENVALUES AND EIGENVECTORS

Let A be a square matrix of order n having elements a_{ij} ($i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$). Now, to find a vector X and a constant λ such that the equation

$$AX = \lambda X \quad ...(1)$$

then λ is called an eigenvalue and X, an eigenvector.

Equation (1) can be rewritten as

$$(A - \lambda I)X = 0 \quad (2)$$

Equation (2) represents a set of homogeneous linear equations:

$$\begin{aligned} (a_{11} - \lambda)\, x_1 + a_{12}\, x_2 + \ldots + a_1 n\, x_n &= 0 \\ a_{21}\, x_1 + (a_{22} - \lambda) x_2 + \ldots + a_2 n\, x_n &= 0 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots & \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots & \\ a_{n1}\, x_1 + \ldots + \ldots + (a_{nn} - \lambda) x_n &= 0 \end{aligned}$$

A non-trivial solution to system (2) exists if and only if

$$|A - \lambda I| = \begin{bmatrix} a_{11} - \lambda & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} - \lambda & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} - \lambda \end{bmatrix} = 0 \quad ...(3)$$

(3) is a polynomial equation of degree n in λ.

Characteristic Matrix

The matrix $A - \lambda I$ *is called characteristic matrix of given matrix A which is obtained by subtracting* λ from diagonal elements of A.

Characteristic Polynomial

The determinant $|A - \lambda I|$ *when expanded will give a polynomial of degree n in* λ which is called characteristic polynomial of matrix A.

Characteristic Equation

The equation $|A - \lambda I| = 0$ *is called characteristic equation of matrix A.*

Characteristic Roots or Eigenvalues

The roots $\lambda_1, \lambda_2, \ldots \lambda_n$ of the characteristic equation are called characteristic roots or eigenvalues.

Characteristic Vector, or Eigenvectors

Corresponding to each characteristic root λ there corresponds non-zero vector X which satisfies the equation $(A - \lambda I)\ X = 0$. *The non-zero vectors X are characteristic vectors or eigenvectors.*

Example 11.62. *Find the eigenvalues and eigenvectors of the following matrix:* **(AMIE, W '95)**

$$A = \begin{bmatrix} -2 & 1 & 1 \\ -6 & 1 & 3 \\ -12 & -2 & 8 \end{bmatrix}$$

Solution. Here characteristic equation $|A - \lambda I| = 0$ is

$$\begin{vmatrix} -2-\lambda & 1 & 1 \\ -6 & 1-\lambda & 3 \\ -12 & -2 & 8-\lambda \end{vmatrix} = 0$$

$$(-2-\lambda)(8 - 9\lambda + \lambda^{2+6}) - (-48 + 6\lambda + 36) + (12 + 12 - 12\lambda) = 0$$

or $\lambda^{3-7}\lambda^{2+14}\lambda - 8 = 0$

or $\lambda^3 - \lambda^{2-6}\lambda^{2+6}\lambda + 8\lambda - 8 = 0$

or $\lambda^{2(}\lambda - 1) - 6\lambda(\lambda - 1) + 8(\lambda - 1) = 0$

or $(\lambda - 1)(\lambda^{2-6}\lambda + 8) = 0$

or $(\lambda - 1)(\lambda - 4)(\lambda - 2) = 0$

or $\lambda = 1, 2, 4$

The eigenvalues are 1, 2, and 4.

For $\lambda = 1$, $[A - \lambda I]\ X = 0$ *becomes*

$$\begin{bmatrix} -2-1 & 1 & 1 \\ -6 & 1-1 & 3 \\ -12 & -2 & 8-1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -3 & 1 & 1 \\ -6 & 0 & 3 \\ -12 & -2 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

i.e.,

$$-3x_1 + x_2 + x_3 = 0 \quad \ldots(1)$$

$$-6x_1 + 0x_2 + 3x_3 = 0 \quad \ldots(2)$$

$$-12x_1 - 2x_2 + 7x_3 = 0 \quad \ldots(3)$$

Solving (1) and (2),

$$\frac{x_1}{3-0} = \frac{x_2}{-6+9} = \frac{x_3}{0+6}$$

i.e., $$\frac{x_1}{1} = \frac{x_2}{1} = \frac{x_3}{2} = k \text{ (say)}$$

For $k = 1$, the eigenvalue corresponding to $\lambda = 1$ is

$$X_1 = [1 \quad 1 \quad 2]^T$$

For $\lambda = 2$, $[A - \lambda I]\, X = 0$ becomes

$$\begin{bmatrix} -2-2 & 1 & 1 \\ -6 & 1-2 & 3 \\ -12 & -2 & 8-2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -4 & 1 & 1 \\ -6 & -1 & 3 \\ -12 & -2 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

i.e.,
$$-4x_1 + x_2 + x_3 = 0$$
$$-6x_1 - x_2 + 3x_3 = 0$$
$$-12x_1 - 2x_2 + 6x_3 = 0$$

Here last two equations are same. Then

$$-4x_1 + x_2 + x_3 = 0$$
$$-6x_1 - x_2 + 3x_3 = 0$$

Solving these equations, taking $x_3 = k$,

$$\frac{x_1}{3+1} = \frac{x_2}{-6+12} = \frac{k}{4+6}$$

i.e., $$\frac{x_1}{4} = \frac{x_2}{6} = \frac{k}{10}$$

or $$\frac{x_1}{2} = \frac{x_2}{3} = \frac{k}{5}$$

or $$x_1 = \frac{2k}{5}, \quad x_2 = \frac{3k}{5} \quad \text{and} \quad x_3 = k$$

If $k = 5$, $x_1 = 2$, $x_2 = 3$ and $x_3 = 5$

For $\lambda = 2$, the eigenvector x2 = [2 3 5]T.

For $\lambda = 4$, $[A - \lambda I]X = 0$ becomes

$$\begin{bmatrix} -6 & 1 & 1 \\ -6 & -3 & 3 \\ -12 & -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

i.e.,
$$-6x_1 + x_2 + x_3 = 0$$
$$-6x_1 - 3x_2 + 3x_3 = 0$$
$$-12x_1 - 2x_2 + 4x_3 = 0$$

Solving first two equations,

$$\frac{x_1}{3+3} = \frac{x_2}{-6+18} = \frac{x_3}{18+6}$$

i.e., $\frac{x_1}{1} = \frac{x_2}{2} = \frac{x_3}{4} = k$ (say)

or $x_1 = k$, $x_2 = 2k$, $x_3 = 4k$, which satisfy third equation.

Taking $k = 1$, the eigenvector is $x^3 = [1 \ 2 \ 4]T$.

Example 11.48. Find the characteristic equation of the matrix

$$A = \begin{bmatrix} 6 & -2 & 2 \\ -2 & 3 & -1 \\ 2 & -1 & 3 \end{bmatrix}$$

Also, find the eigenvalues and eigenvectors.

Solution. Given

$$A = \begin{bmatrix} 6 & -2 & 2 \\ -2 & 3 & -1 \\ 2 & -1 & 3 \end{bmatrix}.$$

The characteristic equation of A is $|A - \lambda I| = 0$, *that is,*

$$\begin{vmatrix} 6-\lambda & -2 & 2 \\ -2 & 3-\lambda & -1 \\ 2 & -1 & 3-\lambda \end{vmatrix} = 0$$

Expanding by first row,

$$(6 - \lambda)\,[(3 - \lambda)2 - 1] + 2[-2(3 - \lambda) + 2] + 2[2 - 2(3 - \lambda)] = 0$$

or $\lambda^{3} - 12\lambda^{2} + 36\lambda - 32 = 0$

or $(\lambda - 2)\,(\lambda - 2)\,(\lambda - 8) = 0$

Hence, the eigenvalues are 2, 2, and 8.

Corresponding to $\lambda = 2$, the eigenvectors are given by

$$(A - 2I)\,X = 0$$

or $$\begin{bmatrix} 4 & -2 & 2 \\ -2 & 1 & -1 \\ 2 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0 \qquad \ldots(1)$$

From relation (1), only one independent equation is obtained.

$$2x1 - x_2 + x_3 = 0$$

Thus, two variables will be assigned arbitrary values and the third will be determined in terms of these arbitrary constants.

Choosing x2 = 0, 2x1 = – x3 or $\frac{x_1}{1} = -\frac{x_3}{2}$. Hence, $\frac{x_1}{1} = \frac{x_2}{0} = \frac{x_3}{-2}$. This gives eigenvectors as [1, 0, –2]T.

Next, choose $x_3 = 0$, $2x_1 - x_2 = 0$ or $x_1 = \frac{x_2}{2}$. Hence, $\frac{x_1}{1} = \frac{x_2}{2} = \frac{x_3}{0}$. This gives the other independent eigenvector as $[1, 2, 0]^T$. Any other eigenvector corresponding to $\lambda = 2$ will be a linear combination of these two eigenvectors.

Corresponding to $\lambda = 8$,

$$\begin{aligned} -2x_1 - 2x_2 + 2x_3 &= 0 \\ -2x_1 - 5x_2 - x_3 &= 0 \\ 2x_1 - x_2 - 5x_3 &= 0 \end{aligned}$$

Solving first two equations,

$$\frac{x_1}{2} = \frac{x_2}{-1} = \frac{x_3}{1} = k, \text{ i.e., } x1 = 2k,\ x2 = -k \text{ and } x3 = k$$

which also satisfy the third equation.

This gives an eigenvector [2, –1, 1]T, when k = 1.

Example 11.49. *Find the eigenvalues and eigenvectors of the matrix* $A = \begin{bmatrix} 3 & 4 \\ 4 & -3 \end{bmatrix}$.

Solution. Given $A = \begin{bmatrix} 3 & 4 \\ 4 & -3 \end{bmatrix}$

The characteristic equation of A is $|A - \lambda I| = 0$, *that is,*

$$\begin{vmatrix} 3-\lambda & 4 \\ 4 & -3-\lambda \end{vmatrix} = 0$$

Expanding by first row

$$(3 - \lambda)(-3 - \lambda) - 16 = 0$$

or $\lambda^{2} - 9 - 16 = 0$, i.e., $\lambda = \pm 5$

Hence, the eigenvalues are 5 and –5.

Corresponding to $\lambda = 5$, the eigenvectors are given by

$$(A - 5I)X = 0$$

or $$\begin{bmatrix} -2 & 4 \\ 4 & -8 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

which gives the equations

$$-2x_1 + 4x_2 = 0$$
$$4x_1 - 8x_2 = 0$$

Only one independent equation is obtained.

Choosing $x_2 = k$, $x_1 = 2k$.

We choose, in particular, $k = \frac{1}{\sqrt{5}}$, so that $x_1 = \frac{1}{\sqrt{5}}$ and $x_2 = \frac{2}{\sqrt{5}}$ and $x_{12} + x_{22} = 1$.

The eigenvector chosen in this way is said to be normalized. We have, therefore, $X_1 = \left[\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}\right]^T$ corresponding to $\lambda = -5$. The eigenvectors are given by

$$(A + 5I)X = 0$$

or $$\begin{bmatrix} 8 & 4 \\ 4 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

which gives the equations

$$8x_1 + 4x_2 = 0$$
$$4x_1 + 2x_2 = 0$$

Choosing $x_2 = k$, $x_1 = -\frac{1}{2}k$

We choose, in particular, $k = \frac{2}{\sqrt{5}}$ so that $x_1 = -\frac{1}{\sqrt{5}}$ and $x_2 = \frac{2}{\sqrt{5}}$ and $x_{12} + x_{22} = 1$.

Corresponding to l = –5, the eigenvector $x_2 = \left[-\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}\right]^T$.

Example 11.50. *Compute the eigenvalues and eigenvectors of the following system:*

$$10x_1 + 2x_2 + x_3 = \lambda x_1$$
$$2x_1 + 10x_2 + x_3 = \lambda x_2$$
$$2x_1 + x_2 + 10x_3 = \lambda x_3$$

(AMIE, S '94)

Solution. Given

$$\begin{aligned} 10x_1 + 2x_2 + x_3 &= \lambda x1 \\ 2x_1 + 10x_2 + x_3 &= \lambda x2 \\ 2x_1 + x_2 + 10x_3 &= \lambda x3 \end{aligned}$$

These equations can be written as

$$\begin{aligned} (10 - \lambda)x_1 + 2x_2 + x_3 &= 0 \\ 2x_1 + (10 - \lambda)x_2 + x_3 &= 0 \\ 2x_1 + x_2 + (10 - \lambda)x_3 &= 0 \end{aligned}$$

The corresponding characteristic equation is given by

$$\begin{vmatrix} 10-\lambda & 2 & 1 \\ 2 & 10-\lambda & 1 \\ 2 & 1 & 10-\lambda \end{vmatrix} = 0$$

Expanding with first row

$$(10 - \lambda)[(10 - \lambda)2 - 1] - 2\,[2(10 - \lambda) - 2] + 1[2 - 2(10 - \lambda)] = 0$$

or $(10 - \lambda)3 - 7(10 - \lambda) + 6 = 0$

or $Y_3 - 7Y + 6 = 0$, where $Y = 10 - \lambda$

or $(Y_3 - Y_2) + (Y_2 - Y) + (-6Y + 6) = 0$

or $Y_2(Y - 1), Y(Y - 1) - 6(Y - 1) = 0$

or $(Y - 1)(Y_2 + Y - 6) = 0$

or $(Y - 1)(Y + 3)(Y - 2) = 0$

i.e., $Y = 1$, $Y = -3$ and $Y = 2$. Therefore, $\lambda = 9$, $\lambda = 13$ and $\lambda = 8$ which are the eigenvalues.

Corresponding to $\lambda = 13$, the eigenvectors are given by

$$\begin{bmatrix} -3 & 2 & 1 \\ 2 & -3 & 1 \\ 2 & 1 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

which gives the equations

$$\begin{aligned} -3x_1 + 2x_2 + x_3 &= 0 \\ 2x_1 - 3x_2 + x_3 &= 0 \\ 2x_1 + x_2 - 3x_3 &= 0 \end{aligned}$$

Solving first two equations,

$$\frac{x_1}{2 + 3} = \frac{x_2}{2 + 3} = \frac{x_3}{9 - 4} = k \text{ (say)}$$

Choosing $k = 1$, the eigenvector $x_1 = [1, 1, 1]T$.

Corresponding to $\lambda = 9$, the eigenvectors are given by

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

which give the equations

$$\begin{aligned} x_1 + 2x_2 + x_3 &= 0 \\ 2x_1 + x_2 + x_3 &= 0 \\ 2x_1 + x2 + x3 &= 0 \end{aligned}$$

Last two equations are same. Chopsing x3 = k, the first two equations become

$$\begin{aligned} x_1 + 2x_2 + k &= 0 \\ 2x_1 + x_2 + k &= 0 \end{aligned}$$

On solving, $x_1 = -\frac{1}{3}k$ and $x_2 = -\frac{1}{3}k$.

Choosing $k = 1$, the eigenvector $x_2 = \left[-\frac{1}{3} \; -\frac{1}{3} \; 1\right]^T$.

Corresponding to $\lambda = 8$, the eigenvectors are given by

$$\begin{bmatrix} 2 & 2 & 1 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

which give the equations

$$2x_1 + 2x_2 + x_3 = 0$$
$$2x_1 + 2x_2 + x_3 = 0$$
$$2x_1 + x_2 + 2x_3 = 0$$

First two equations are same. Choosing x3 = k, the last two equations become

$$2x_1 + 2x_2 + k = 0$$
$$2x_1 + x_2 + 2k = 0$$

On solving, $x_1 = -\frac{3}{2}k$, $x_2 = k$, $x_3 = k$.

Choosing $k = 1$, the eigenvector $x_3 = \left[-\frac{3}{2} \; 1 \; 1\right]^T$.

Properties of Eigenvalues

The following are the important properties of eigenvalues:

1. The sum of the eigenvalues of a matrix A, is the sum of the elements of its principal diagonal.
2. If λ is an eigenvalue of a matrix A, then $1/\lambda$ is the eigenvalue of A^{-1}.
3. If λ is an eigenvalue of an orthogonal matrix, then $1/\lambda$ is also its eigenvalue.
4. If $\lambda_1, \lambda_2, \ldots, \lambda n$ are the eigenvalues of a matrix A, then A^m has the eigenvalues λ_{1m}, $\lambda_{2m}, \ldots \lambda_{nm}$, ($m$ being a positive integer).
5. Eigenvectors corresponding to distinct eigenvalues of a matrix are linearly independent.
6. If a square matrix A has n linearly independent eigenvectors, then a matrix P can be found such that $P^{-1}AP$ is a diagonal matrix whose diagonal elements are the eigenvalues of A.
   The transformation of A by a non-singular matrix P to $P^{-1}AP$ is called a similarity transformation.
7. Any similarity transformation applied to a matrix leaves its eigenvalues unchanged.

Gerschgorin bounds: If λ be an eigenvalue of a matrix A, then for some k ($1 \le k \le n$),

$$|\lambda - akk| \le |ak1| + |ak2| + \ldots + |akn| = \rho_{k\,(say)}$$

i.e., all the eigenvalues of A lie in the union of the n circles with centres a_{kk} and radii ρ_k.

The eigenvalue bounds so obtained are called Gerschgorin bounds.

Inverse of Matrix using Characteristic Equation

The method of finding the inverse of a matrix by using its characteristic equation is illustrated with an example. The method is based on Cayley Hamilton theorem which states that every square matrix satisfies its characteristic equation.

Example 11.51. *Verify Cayley Hamilton theorem for the matrix*

$$A = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix}$$

Hence, compute A^{-1}.

Solution. The characteristic equation is

$$|A - \lambda I| = \begin{vmatrix} 2-\lambda & -1 & 1 \\ -1 & 2-\lambda & -1 \\ 1 & -1 & 2-\lambda \end{vmatrix} = 0$$

Expanding by first row,

$$\lambda^3 - 6\lambda^2 + 9\lambda - 4 = 0$$

To verify the theorem, it is essential to show that

$$A^3 - 6A^2 + 9A - 4I = 0 \qquad \ldots(1)$$

Here $$A^2 = AA = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix}\begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 6 & -5 & 5 \\ -5 & 6 & -5 \\ 5 & -5 & 6 \end{bmatrix}$$

$$A^3 = A^2A = \begin{bmatrix} 6 & -5 & 5 \\ -5 & 6 & -5 \\ 5 & -5 & 6 \end{bmatrix}\begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 22 & -21 & 21 \\ -21 & 22 & -21 \\ 21 & -21 & 22 \end{bmatrix}$$

Hence, $A^3 - 6A^2 + 9A - 4I$

$$= \begin{bmatrix} 22 & -21 & 21 \\ -21 & 22 & -21 \\ 21 & -21 & 22 \end{bmatrix} - 6\begin{bmatrix} 6 & -5 & 5 \\ -5 & 6 & -5 \\ 5 & -5 & 6 \end{bmatrix} + 9\begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix} - 4\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 22 & -36+18=4 & -21+30-9=0 & 21-30+9=0 \\ -21 & +30-9=0 & 22-36+18=4 & -21+30-9=0 \\ 21 & -30+9=0 & -21+30-9=0 & 22-36+18=4 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = 0$$

Thus, A satisfies its characteristic equation.

Now, multiplying relation (1) by A^{-1},

$$A^2 - 6A + 9I - 4A^{-1} = 0$$

or $$4A^{-1} = A^2 - 6A + 9I$$

$$= \begin{bmatrix} 6 & -5 & 5 \\ -5 & 6 & -5 \\ 5 & -5 & 6 \end{bmatrix} - 6\begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix} + 9\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 6-12+9 & -5+6+0 & 5-6+0 \\ -5+6+0 & 6-12+9 & -5+6+0 \\ 5-6+0 & -5+6+0 & 6-12+9 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -1 \\ 1 & 3 & 1 \\ 0 & 3 & 3 \end{bmatrix}$$

or $$A^{-1} = \frac{1}{4}\begin{bmatrix} 3 & 1 & -1 \\ 1 & 3 & 1 \\ -1 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 3/4 & 1/4 & -1/4 \\ 1/4 & 3/4 & 1/4 \\ -1/4 & 1/4 & 3/4 \end{bmatrix}$$

Method of Iteration (Power Method)

This method finds out the largest eigenvalue of a square matrix by repeated use of the basic equation

$$AX = \lambda X$$

Let $X^{(0)}$ be the initial guess of the eigenvector. Then the product $AX^{(0)}$ is computed. Let

$$AX^{(0)} = \lambda_1 X^{(1)}$$

where λ_1 is the largest element of $AX^{(0)}$ and $X^{(1)}$ is another eigenvector. Repeating the procedure with vector $X^{(1)}$, it yields

$$AX^{(1)} = \lambda_2 X^{(2)}$$

The process is continued till the difference of values of λ_i between two successive iterations is less than the required degree of accuracy. Thus, when this process is converged, li will be the largest eigenvalue and the vector $X^{(i)}$ becomes its associated eigenvector.

Iterative Method for Finding Smallest Eigenvalue

The power method can also be used to find the smallest eigenvalue of the matrix A in the following manner:

Given $AX = \lambda X$

Pre-multiplying with A^{-1} on both sides,

$$A^{-1}(AX) = A^{-1}(\lambda X)$$

or $$X = \lambda A^{-1} X$$

i.e., $$A^{-1}X = \frac{1}{\lambda} X$$

or $$BX = \lambda^* X, \text{ where } B = A^{-1} \text{ and } \lambda^* = \frac{1}{\lambda}.$$

Now, the application of the power method to B will yield the largest value of λ^* as the eigenvalue. Then $\lambda = 1/\lambda^*$ represents the smallest eigenvalue of A.

Iterative Method for Finding Intermediate Eigenvalues

The other eigenvalues between the smallest and the largest may also be found in the following manner: If λ_1 is the largest eigenvalue of A found by power method and x^1 is the corresponding eigenvector, then let

$$A_1 = A - \frac{\lambda_1 [x_1 x_1^T]}{x_1^T x_1}$$

where x1T is transpose of x1 and hence x1Tx1 is a scalar and [x1, x1T] is a symmetric square (n × n) matrix. It can be shown that the (n – 1) eigenvalues of the (n × n) matrix A1 are the same as those of A excepting λ_1, and the nth eigenvalue of A1 is zero. Thus, when the power method is applied to A1, it will yield the largest eigenvalue l2 of A1 (i.e., second largest eigenvalue of A) and the associated eigenvector x2. The third largest eigenvalue of A can then be found by applying the power method to A2 expressed as

$$A_2 = A_1 - \frac{\lambda_2 [x_2 x_2^T]}{x_2^T x_2}$$

The two eigenvalues of A^2 are zero. The power method when applied to A2 will converge to λ_3, the third largest eigenvalue of A. The procedure may thus be repeated until all n eigenvalues of A are found. This method is suitable for matrices of small size but becomes computationally inefficient and inaccurate for large matrices.

Example 11.52. Find the largest eigenvalue correct up to 2 decimal places and the corresponding eigen vector of the following matrix by power method:

$$A = \begin{bmatrix} 1 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 1 & 3 \end{bmatrix}.$$

Solution. Let the initial eigenvector $x^{(0)}$ be $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

$$AX^{(0)} = \begin{bmatrix} 1 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 1 & 3 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1. \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 7 \end{bmatrix} = 7 \begin{bmatrix} 0.714 \\ 0 \\ 1 \end{bmatrix} = \lambda_{1X(1)}$$

$$AX^{(1)} = \begin{bmatrix} 1 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 1 & 3 \end{bmatrix} \times \begin{bmatrix} 0.714 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.714 \\ 1.714 \\ 5.142 \end{bmatrix} = 5.142 \begin{bmatrix} 0.722 \\ 0.333 \\ 1 \end{bmatrix} = \lambda_{2X(2)}$$

$$AX^{(2)} = \begin{bmatrix} 1 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 1 & 3 \end{bmatrix} \times \begin{bmatrix} 0.722 \\ 0.333 \\ 1 \end{bmatrix} = \begin{bmatrix} 4.055 \\ 1.056 \\ 5.499 \end{bmatrix} = 5.499 \begin{bmatrix} 0.737 \\ 0.192 \\ 1 \end{bmatrix} = \lambda_{3X(3)}$$

$$AX^{(3)} = \begin{bmatrix} 1 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 1 & 3 \end{bmatrix} \times \begin{bmatrix} 0.737 \\ 0.192 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.929 \\ 1.353 \\ 5.403 \end{bmatrix} = 5.403 \begin{bmatrix} 0.727 \\ 0.250 \\ 1 \end{bmatrix} = \lambda_{4X(4)}$$

$$AX^{(4)} = \begin{bmatrix} 1 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 1 & 3 \end{bmatrix} \times \begin{bmatrix} 0.727 \\ 0.250 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.977 \\ 1.227 \\ 5.431 \end{bmatrix} = 5.431 \begin{bmatrix} 0.732 \\ 0.226 \\ 1 \end{bmatrix} = \lambda_{5X(5)}$$

$$AX^{(5)} = \begin{bmatrix} 1 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 1 & 3 \end{bmatrix} \times \begin{bmatrix} 0.732 \\ 0.226 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.958 \\ 1.28 \\ 5.422 \end{bmatrix} = 5.422 \begin{bmatrix} 0.730 \\ 0.236 \\ 1 \end{bmatrix} = \lambda_{6X(6)}$$

$$AX^{(6)} = \begin{bmatrix} 1 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 1 & 3 \end{bmatrix} \times \begin{bmatrix} 0.730 \\ 0.236 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.966 \\ 1.258 \\ 5.426 \end{bmatrix} = 5.426 \begin{bmatrix} 0.731 \\ 0.232 \\ 1 \end{bmatrix} = \lambda_{7X(7)}$$

Now, $|\lambda_7 - \lambda_6| = |5.426 - 5.422| = 0.004 < 0.005$

Thus, the largest eigenvalue corrected up to 2 decimal places is 5.426 and the corresponding eigenvector is

$$\begin{bmatrix} 0.731 \\ 0.232 \\ 1 \end{bmatrix}$$

REVIEW QUESTIONS SET

1. Find the adjoint and, hence, the inverse of the following matrices:

$$(a)\begin{bmatrix}2&2&4\\1&3&2\\3&1&3\end{bmatrix}\quad(b)\begin{bmatrix}1&2&1\\5&2&3\\1&1&2\end{bmatrix}\quad(c)\begin{bmatrix}0&1&2\\1&2&3\\3&1&1\end{bmatrix}$$

2. By using Gauss elimination method, obtain the inverse of the following matrices:

$$(a)\begin{bmatrix}3&-3&4\\2&-3&4\\0&-1&1\end{bmatrix}\quad(b)\begin{bmatrix}4&4&8\\2&6&4\\6&2&6\end{bmatrix}\quad(c)\begin{bmatrix}6&-6&8\\4&-6&8\\0&-2&1\end{bmatrix}$$

$$(d)\begin{bmatrix}1&1&1&1\\1&2&3&-4\\2&3&5&-5\\3&-4&-5&8\end{bmatrix}\quad(e)\begin{bmatrix}1&1&1\\2&2&3\\1&4&9\end{bmatrix}$$

(AMIE, S '97)

3. Use Gauss-Jordan elimination method to obtain the inverse of example 2.
4. Find the inverse of $A=\begin{bmatrix}1&1\\1&2\end{bmatrix}$ using the iterative method, the approximate inverse given as $B=\begin{bmatrix}1.8&-0.9\\-0.9&0.9\end{bmatrix}$. Perform two iterations.
5. Find the inverse of the following matrix by Gauss-Jordan elimination method:

$$\begin{bmatrix}1&5&3\\1&3&2\\2&4&-6\end{bmatrix}$$

(AMIE, S '97)

6. Solve the following sets of linear equations with the help of matrix inversion:
   (a) $x+4y+3z=1,\quad 2x+y+4z=4,\quad x-3y-2z=5$
   (b) $2x-y+3z=8,\quad -x+2y+z=4,\quad 3x+y-4z=0$
   (c) $x-y+z+w=4,\quad 2x+3y-2w=-5,\quad 3y-z-w=-5,\ 5x-y-z=6$
   (d) $2x+3y+z=9,\quad x+2y+3z=6,\quad 3x+y+2z=8$ **(AMIE, S'96)**
   (e) $x+y+z=4,\quad x+2y+3z=7,\quad x+4y+9z=15$ **(AMIE, W '97)**
7. Solve the following system of equations by (i) Gauss elimination, (ii) Gauss-Jordan methods:
   (a) $x-y+z=1,\quad -3x+2y-3z=-6,\quad 2x-5y+4z=5$
   (b) $3x+y-z=3,\quad 2x-8y+z=-5,\quad x-2y+9z=8$
   (c) $x+2y+3z=6,\quad 2x+4y+z=7,\quad 3x+2y+9z=14$
   (d) $x+y+2z=4,\quad 3x+y-3z=-4,\quad 2x-3y-5z=-5$
8. Given the following system of equations:

$$20x_1+x_2-2x_3=17$$
$$3x_1+20x_2-x_3=-18$$
$$2x_1-3x_2+20x_3=25$$

   Find the solution, using the Jacobi's method, correct to four significant digits.
9. Find the solution using the Gauss-Siedel method correct up to four significant digits for the following system of equations:

$$10x+y+2z=44$$
$$2x+10y+z=51$$
$$x+2y+10z=61$$

10. Solve the following system of equations by Jacobi and Gauss-Siedel methods correct to three decimal places:

$$x+y+54z=110$$
$$27x+6y-z=85$$
$$6x+15y+2z=72$$

11. Determine the eigenvalues and eigenvectors for the following matrices:

$$(a)\begin{bmatrix}2&2&1\\1&3&1\\1&2&2\end{bmatrix}\quad(b)\begin{bmatrix}5&2&1\\2&5&1\\2&1&5\end{bmatrix}\quad(c)\begin{bmatrix}5&0&1\\0&-2&0\\1&0&5\end{bmatrix}$$

(*d*) $\begin{bmatrix} 3 & 1 & 4 \\ 0 & 2 & 0 \\ 0 & 0 & 5 \end{bmatrix}$ (*e*) $\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$ (*f*) $\begin{bmatrix} 6 & -2 & 2 \\ -2 & 3 & -1 \\ 2 & -1 & 3 \end{bmatrix}$ **(AMIE, W '94, W '97)**

(*g*) $\begin{bmatrix} 2 & 2 & -1 \\ 2 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$ **(AMIE, W'98)**

12. Compute the eigenvalues and the corresponding eigenvectors of the following system:

$$10x_1 + 2x_2 + x_3 = \lambda x_1$$
$$2x_1 + 10x_2 + x_3 = \lambda x_2$$
$$2x_1 + x_2 + 10x_3 = \lambda x_3$$

(AMIE, S '94)

13. Find the largest eigenvalue corrected up to 2 decimal places and the corresponding eigenvector of the following matrix by power method:

$$A = \begin{bmatrix} 10 & 5 & 6 \\ 5 & 20 & 4 \\ 6 & 4 & 30 \end{bmatrix}$$

(AMIE, S '97)

14. Given the matrix

$$A = \begin{bmatrix} 4 & 1 & -1 \\ 2 & 3 & -1 \\ -2 & 1 & 5 \end{bmatrix}$$

Find the largest eigenvalue and the corresponding eigenvector.

15. Find the numerically largest eigenvalue of

$$A = \begin{bmatrix} 25 & 1 & 2 \\ 1 & 3 & 0 \\ 2 & 0 & -4 \end{bmatrix}$$ and the corresponding eigenvector.

16. Find the solution of the following system of equations:

$$5x_1 + 3x_2 - x_3 = 11$$
$$2x_1 + 4x_3 + x_4 = 1$$
$$-3x_1 + 3x_2 - 3x_3 + 5x_4 = -2$$
$$6x_2 - 2x_3 + 3x_4 = 9$$

by Gaussian elimination with partial pivoting.

17. Solve the system of equations

(*a*)
$$x_1 + 3x_2 - 2x_3 = 7$$
$$4x_1 - x_2 + 3x_3 = 10$$
$$-5x_1 + 2x_2 + 3x_3 = 7$$

(*b*)
$$x_1 + 2x_2 + x_3 + 4x_4 = 13$$
$$2x_1 + 4x_3 + 3x_4 = 28$$
$$4x_1 + 2x_2 + 2x_3 + x_4 = 20$$
$$-3x_1 + x_2 + 3x_3 + 2x_4 = 6$$

(AMIE, W '97)

by Gaussian elimination with full pivoting.

ANSWERS TO REVIEW QUESTIONS SET

1. (a) $\begin{bmatrix} 7 & -2 & -8 \\ 3 & -6 & 0 \\ -8 & 4 & 4 \end{bmatrix}$, $\frac{1}{-12}\begin{bmatrix} 7 & -2 & -8 \\ 3 & -6 & 0 \\ -8 & 4 & 4 \end{bmatrix}$ (*b*) $\begin{bmatrix} 1 & -3 & 4 \\ -3 & 1 & 0 \\ 1 & 1 & -4 \end{bmatrix} = \begin{bmatrix} -\frac{1}{4} & \frac{3}{4} & -1 \\ \frac{3}{4} & -\frac{1}{4} & 0 \\ -\frac{1}{4} & -\frac{1}{4} & 1 \end{bmatrix}$

(*c*) $\begin{bmatrix} -1 & 1 & -1 \\ 8 & -6 & 2 \\ -5 & 3 & -1 \end{bmatrix}$, $\begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ -4 & 3 & -1 \\ \frac{5}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix}$

2. (a) $\begin{bmatrix} 1 & -1 & 0 \\ -2 & 3 & -4 \\ -2 & 3 & -3 \end{bmatrix}$ (*b*) $\begin{bmatrix} -\frac{7}{24} & \frac{1}{12} & \frac{2}{6} \\ -\frac{1}{8} & \frac{1}{4} & 0 \\ \frac{2}{6} & -\frac{1}{6} & -\frac{1}{6} \end{bmatrix}$ (*c*) $\begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{5} & \frac{3}{10} & -\frac{4}{5} \\ -\frac{2}{5} & \frac{3}{5} & -\frac{3}{5} \end{bmatrix}$

(*d*) $\frac{1}{18}\begin{bmatrix} 2 & 16 & -6 & 4 \\ 22 & 41 & -30 & -1 \\ -10 & -44 & 30 & -2 \\ 4 & -13 & 6 & -1 \end{bmatrix}$ (*e*) $\begin{bmatrix} -2 & \frac{5}{3} & -\frac{1}{3} \\ 5 & -\frac{8}{3} & \frac{1}{3} \\ -2 & 1 & 0 \end{bmatrix}$

4. $\begin{bmatrix} 1.998 & -0.9999 \\ -0.9999 & 0.9999 \end{bmatrix}$ **5.** $\begin{bmatrix} -\frac{13}{9} & \frac{7}{3} & \frac{1}{18} \\ \frac{5}{9} & -\frac{2}{3} & \frac{1}{18} \\ -\frac{1}{9} & \frac{1}{3} & -\frac{1}{9} \end{bmatrix}$

6. (*a*) (3, −2, 2) (*b*) (2, 2, 2) (*c*) (1, −1, 0, 2) (*d*) (35/18, 29/18, 5/18) (*e*) (2, 1, 1)

7. (*a*) (−2, 3, 6) (*b*) (1, 1, 1) (*c*) (1, 1, 1) (*d*) (1, −1, 2)

8. 0.9999, − 0.9999, 0.9999 **9.** 2.999, 4.000, 5.000 **10.** 2.425, 3.573, 1.926

11. (*a*) 5, 1, 1, $x_1 = [k\ k\ k]^T$, $x^2 = [c_1, c_2, -(c_1 + c_2)]^T$

(*b*) 3, 4, 8, $x_1 = \left[-\frac{3}{2}k\ \ k\ \ k\right]^T$, $x_2 = [k\ \ k\ -3k]^T$, $x_3 = [k\ k\ k]^T$

(*c*) −2, 4, 6, $x_1 = [0\ k\ 0]^T$, $x_2 = [k\ 0, -k]^T$, $x_3 = [k\ 0\ k]^T$

(*d*) 2, 3, 5, $x_1 = [k\ -k\ 0]^T$, $x_2 = [k\ 0\ 0]^T$, $x_3 = [2k\ 0\ k]^T$

(*e*) 2, $2 + \sqrt{2}$, $2 - \sqrt{2}$, x1 $= [-k\ 0\ k]^T$, $x_2 = [k\ \sqrt{2}\ k\ k]^T$, $x_3 = [0\ \ 0\ \ 0]^T$

(*f*) 2, 2, 8; (1, 0, −2), (1, 2, 0), (2, −1, −1)

(*g*) 1, −1, 4; $[0, 1, 2]^T$, $[1\ \ -1\ \ 1]^T$, $\left[1\ \ \frac{2}{3}\ \ -\frac{2}{3}\right]^T$

12. 8, 9, 13, $x_1 = [-3k, 2k, 2k]^T$, $x_2 = [k\ k\ -3k]^T$, $x_3 = [3k\ k\ k]^T$

13. $\lambda = 33.714$, $x = [0.341\ \ 0.417\ \ 1.000]^T$

14. $\lambda = 6$ $x = [1\ \ 1\ \ -1]^T$

15. $\lambda = 25.1821$ $x = [10.0451\ \ 0.0685]^T$

16. $x_1 = 1.0$ $x_2 = 2.0$, $x_3 = 0.0$, $x_4 = -1.0$

17. (*a*) $x_1 = 1.5000$, $x_2 = 3.5001$, $x_3 = 2.5000$ (*b*) $x_1 = 3$, $x_2 = -1$, $x_3 = 4$, $x_4 = 2$

12

ALGORITHMS AND FLOW CHARTS

12.0. INTRODUCTION

For solving any problem on a computer, the necessary instructions describing a scheme of solution for the problem are to be prepared first. Irrespective of the computers used, all the basic operations described by the instructions of the scheme remain same. The steps of basic operations by following which a solution to a problem can be obtained constitute an algorithm for the problem. The graphical, pictorial or schematic representation that illustrates the sequence of operations to be performed to arrive at the solution is called a *flow chart.*

12.1. ALGORITHMS

A computer need precise and well defined instructions for finding solutions of problems. An algorithm is a step-by-step procedure for solving a problem. An algorithm may have the following properties :

1. It must be finite (Finiteness).
2. It must be unambiguous (Definiteness).
3. It must be executable (Non-intuitiveness).
4. It must have 0 or more input (Input).
5. It must have 1 or more output (Output).
6. It should be general so that it can solve any problem of a particular type for which it is constructed (completeness/generality).

Algorithm presents the basic frame which is transformed into the program using data structures as the building blocks. To explain the technique for algorithm development, some examples are given:

Example 12.1. *Write an algorithm to compute the factorial of a positive number read from the input stream.*

Solution.

```
Step 1          Start
Step 2          (Input)
                Read (N)
Step 3          (Initialize Variables)
                PROD = 1
                MULT = N
Step 4          (Loop)
                Repeat through step 6 while MULT ≥ 1
Step 5          (Partial Product)
                PROD = PROD * MULT
Step 6          (Reset)
                MULT = MULT - 1
Step 7          (Output)
                Write ('Factorial of', N, 'is' PROD)
Step 8          Stop
```

The algorithm can be illustrated by taking $N = 5$.

INPUT	$N = 5$	
INITIALIZATION	PROD = 1,	MULT = 5
ITERATION 1	PROD = 1 × 5 = 5,	MULT = 5 − 1 = 4
ITERATION 2	PROD = 5 × 4 = 20,	MULT = 4 − 1 = 3
ITERATION 3	PROD = 20 × 3 = 60,	MULT = 3 − 1 = 2
ITERATION 4	PROD = 60 × 2 = 120,	MULT = 2 − 1 = 1
ITERATION 5	PROD = 120 × 1 = 120	MULT = 1 − 1 = 0

Since MULT = 0 ≠ 1, now iteration will stop here.

OUTPUT : Factorial of 5 is 120.

12.2. VARIABLES AND OPERATIONS

In writing algorithms, one should have knowledge about the variables and the different operations which are explained below:

12.2.1. Variables

The variables refer to the quantities which are not fixed. They are used to refer input data, intermediate results of computation and final results for writing algorithms. Variables are normally named with the symbols of English alphabet. In Example 12.1, we have used N, PROD, MULT as variables.

The equality sign (=) is used for assigning values to variables. The value of the expression in the right hand side is computed first and then assigned to the variable that occurs in the left hand side. Since a variable can have only one value at a time, after a new value is assigned to it, its old value is lost. In Example 12.1, in initialization PROD = 1, and MULT = 5, but after 1st iteration PROD = 5, and MULT = 4. In step 5, the expression PROD × MULT in right hand side is evaluated first by taking initial values and the new computed value 5 is assigned to PROD.

12.2.2. Decision-Making Operations

Decision-making operations are required to specify various alternative steps of operations when certain condition is satisfied.

12.2.3. Branching Operations

The steps of operation in an algorithm are normally performed in sequence. Sometimes, however, it is necessary to jump to some other part of the program, then altering normal sequence of operations. The necessary reference to such a step is made by mentioning its step number after a GO TO instruction in the step from where the reference is needed. Such a jump operation is known as an unconditional branching operation.

12.2.4. Looping Operations

Looping operation involves repeating some portion of the program either a specified number of times or until some particular condition has been satisfied. The repeated portion of the program (the loop) may contain a conditional branching operation that determines whether the loop will be terminated or executed at least one more time.

In example 12.1, the same sequence of steps from 5 to 6 is repeated as many times as the value of MULT ≥ 1.

Example 12.2. *Write an algorithm to prepare a table giving the radius and area of circles with radius 5, 10, 15, ..., 50.*

Solution. Step 1 Start

Step 2	(Initialization)
	$R = 0$
Step 3	(LOOP)
	Repeat through 6 until $R = 50$
Step 4	$R = R + 5$
Step 5	$A = (22/7) \times R^2$
Step 6	Write ('Area is', A, 'when Radius', R)
Step 7	Stop

Example 12.3. *Write an algorithm for finding the solution of the following pair of equations in X and Y:*

$$a_1 X + a_2 Y = b_1 \quad ...(1)$$

$$a_3 X + a_4 Y = b_2 \quad ...(2)$$

Solution. Assuming that a_1, a_2, a_3, a_4, b_1 and b_2 are non-zero. We write from equation (1).

$$X = (b_1 - a_2 Y) / a_1 \quad ...(3)$$

Substituting the above value of X in equation (2),

$$a_3 (b_1 - a_2 Y) / a_1 + a_4 Y = b_2$$

or $(a_4 - a_2 a_3 / a_1) Y = b_2 - a_3 b_1 / a_1$

Putting $P = a_4 - a_2 a_3 / a_1$ and $Q = b_2 - a_3 b_1 / a_1$,

$Y = Q/P$

The following is the algorithm for the above method of solving the equations:

Step 1	Start
Step 2	Read (a_1, a_2, a_3, a_4, b_1, b_2)
Step 3	$P = a_4 - a_2 a_3 / a_1$
Step 4	$Q = b_2 - a_3 b_1 / a_1$
Step 5	If $P = 0$, Write ('No Solution') , GO to step 3
Step 6	$Y = (Q/P)$
Step 7	$X = (b_1 - a_2 y) / a_1$
Step 8	Write (X, Y)
Step 9	Stop

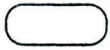
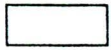
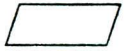
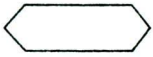
12.3 FLOW CHART

Flow chart is the pictorial representation of an algorithm. The operation instructions are placed in different boxes which are connected by arrows, known as directional arrows, to indicate the flow of logic in the entire process. Boxes of different shapes are used to indicate different kinds of operations.

Why Flow Chart is Drawn?

1. It is easier to understand at a glance than a narrative description.
2. The flow chart assists the programmer when he actually starts writing the program.
3. If the programmer himself or someone else wishes to correct or modify the program after sometime, the flow chart may be more clear and easy to understand than the actual program.
4. Provide effective program documentation.

The six most frequently used symbols are:

Symbol	*Shape*	*Name*
	Oval	Terminal
	Rectangle	Assertion/Process
	Parallelogram	Input/Output
	Diamond	Decision
	Hexagon	Preparation / Group Instruction
	Circle	Connector

Terminal Box

This symbol is used to indicate the beginning or end of an algorithm by (Start) or (Stop).

Assertion Box

This symbol is used to indicate a processing operation like arithmetic operation, assigning value to a variable.

$D = B^2 - 4AC$ SUM = 0

Input/Output Box

This box is used to indicate the reading of the initial data on which the subsequent operations are to be performed and the writing of the intermediate and final results. Input statement begins with the word Read followed by variable names separated by commas.

Read X, Y

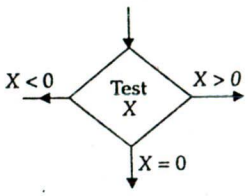
Output statement begins with the word write followed by variable names separated by commas.

Write A, B

Decision Box

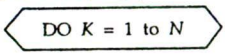
This box is used when a logical decision has to be taken, i.e., it indicates a conditional transfer in an algorithm. It can have one or more entries but it will have two or more exits. We may be interested in different sets of computations in the three cases defined by the condition of:

X viz., X < 0, X = 0 or X > 0



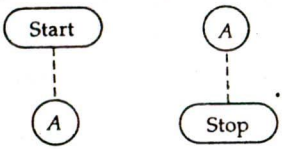
Preparation Box

This box indicates the beginning of a looping procedure.



Connector Box

This box is used to indicate the connection of two portions of flow chart. Normally these are used for long flow chart. One box acts as an exit connector and other as entry connector. Connectors connecting the corresponding points in the flow chart should have the same name and different parts of the flow chart should have different names. This is also used to link two points that involve repeating a process.



Example 12.4. *Draw a flow chart to find the average of four numbers stored in variables P, Q, R, S.*

Solution. The flow chart is shown in Fig. 12.1.

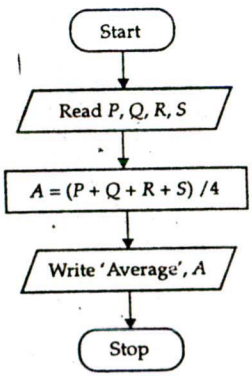


Fig. 12.1

Example 12.5. *Draw a flow chart for converting the given Fahrenheit temperature to Centigrade temperature.*

Solution. The formula of converting Fahrenheit temperature to Centigrade temperature is

$$C = \frac{5}{9}(F - 32)$$

where C = temperature in centigrade
F = temperature in fahrenheit.

The flow chart is shown in Fig. 12.2.

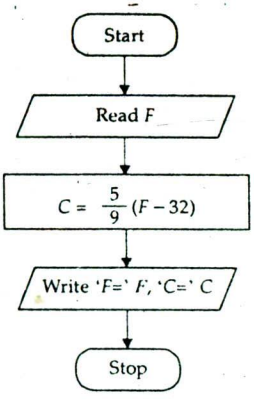


Fig. 12.2

Example 12.6. *Draw the flow chart to calculate* $Y = AX^2 + BX + C$ *when* $A = 10$, $B = 20$, $C = 30$ *and* $X = .1(.1)\ 2.5$.

Solution. The flow chart is shown in Fig. 12.3.

The values of Y are calculated for different values of X starting with $X = .1$ with increment of .1. The loop is terminated when $x = 2.5$.

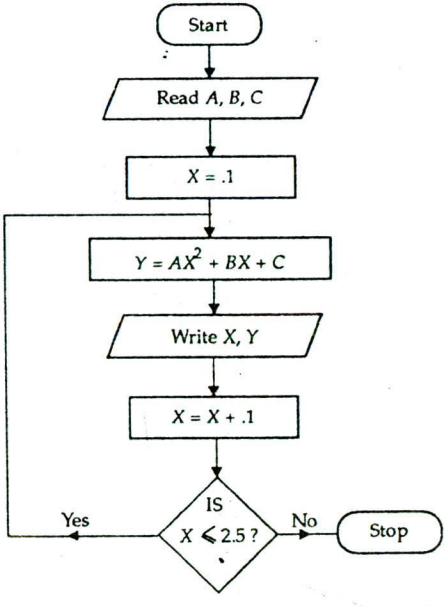


Fig. 12.3

Example 12.7. *Draw a flow chart of factorial* N $(N \geq 1)$.

Solution. We know $N\ ! = 1 \times 2 \times 3 \times ....(N - 1) \times N$. Let F holds the value of $N!$ and it is initialised to 1. The counter A is initialised to 0 and is incremented by 1. The loop is terminated when $A = N$.

The flow chart is shown in Fig. 12.4.

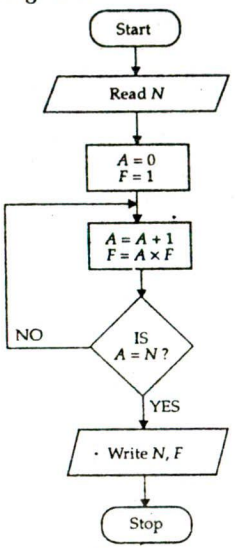


Fig. 12.4

Example 12.8. *Draw a flow chart that calculates*

$$1 + \frac{1}{3} + \frac{1}{5} + \ldots + \frac{1}{N}$$ *, where one first reads N.*

Solution. The flow chart is shown in Fig. 12.5.

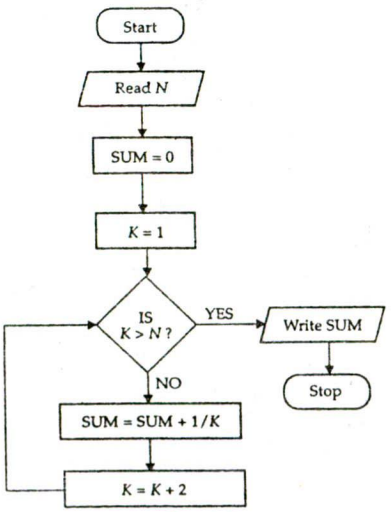


Fig. 12.5

Example 12.9. *Draw a flow chart of summing up of first N positive integers.*
Solution. The flow chart is shown in Fig. 12.6.

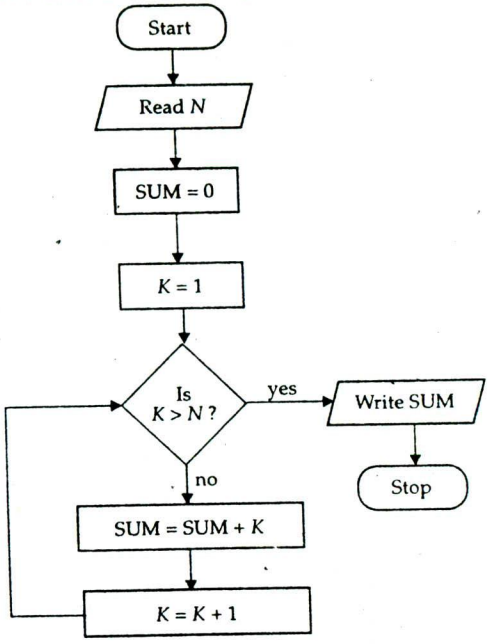


Fig. 12.6

Example 12.10. *Draw the flow chart for obtaining the sum of the series*
$1 + x + x^2 + x^3 + \ldots + x^k$.
Solution. The flow chart is shown in Fig. 12.7.

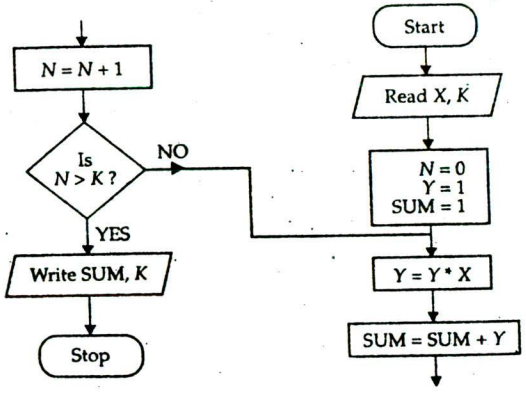


Fig. 12.7

Example 12.11. Draw a flow chart to solve a general quadratic equation

$$AX^2 + BX + C = 0.$$

Solution. $AX^2 + BX + C = 0$ gives $X = \dfrac{-B \pm \sqrt{B^2 - 4AC}}{2A}$. The flow chart is shown in Fig. 12.8.

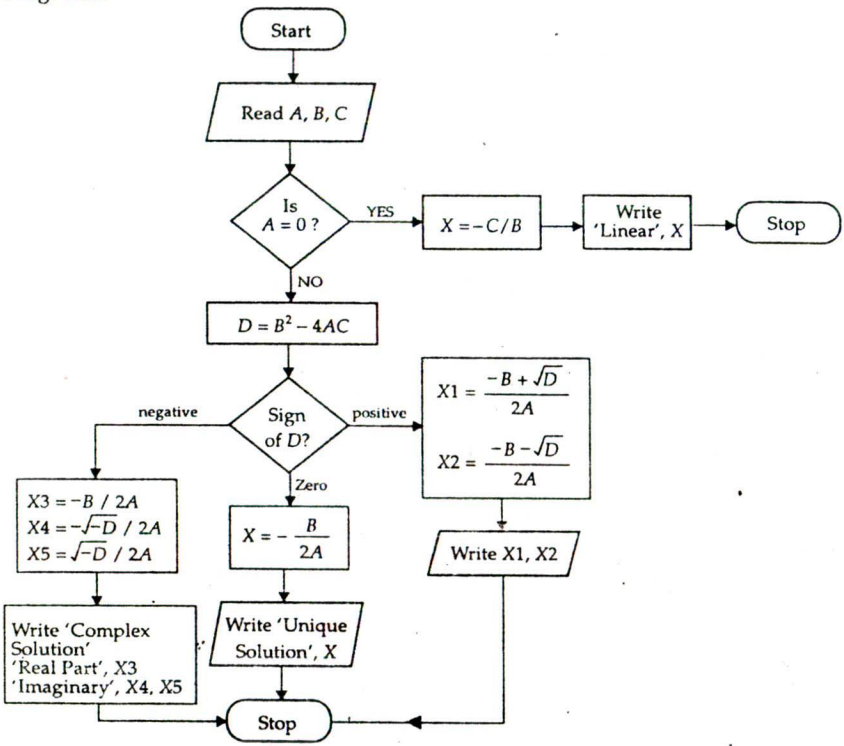
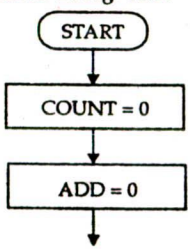


Fig. 12.8

Example 12.12. Give a flow chart for the following problem: 'Read data values and keep on adding them until negative value is entered. After that output the sum and number of data values read and added'.

Solution. **The flow chart is shown in Fig. 12.9.**



Contd.,

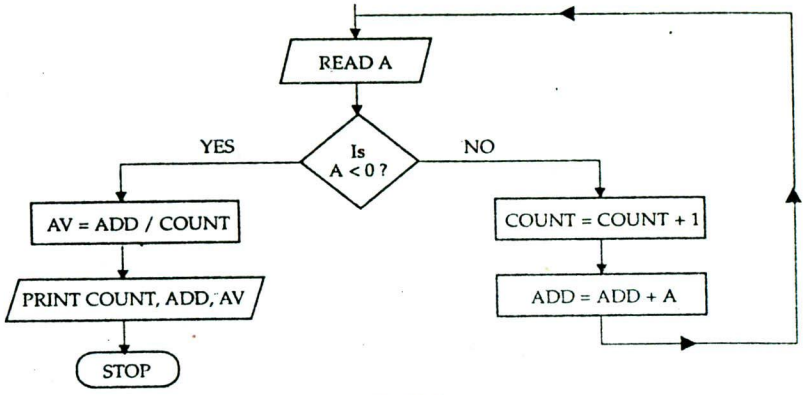


Fig. 12.9

Example 12.13. *Give a flow chart for the following problem: 'A set of non-zero numbers are to be read from which the sum of all even numbers has to be found out, along with their average. The program terminates when a zero or a negative number is read.'*

Solution. The flow chart is shown in Fig. 12.10.

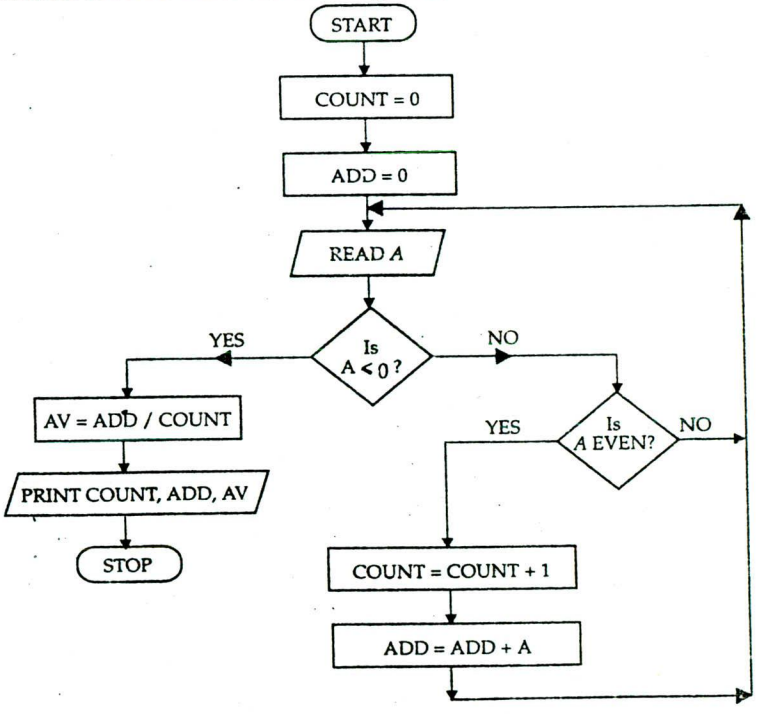


Fig. 12.10

Example 12.14. *Draw a flow chart to evaluate* $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots$ *for a given x in radians. The computation should be stopped when the absolute value of a term becomes less than* 10^{-6} *or adding 10 terms whichever is earlier.*

Solution. Here T_n (nth term of cos x) $= \frac{(-1)^{n-1} \cdot x^{2n-2}}{(2n-2)!}$

and $$T_{n+1} = \frac{(-1)^n x^{2n}}{(2n)!} = -\frac{x^2}{2n(2n-1)} T_n \quad \ldots(1)$$

Thus, if $T_1 = 1$, the successive terms can be evaluated using formula (1). The flow chart is shown in Fig. 12.11.

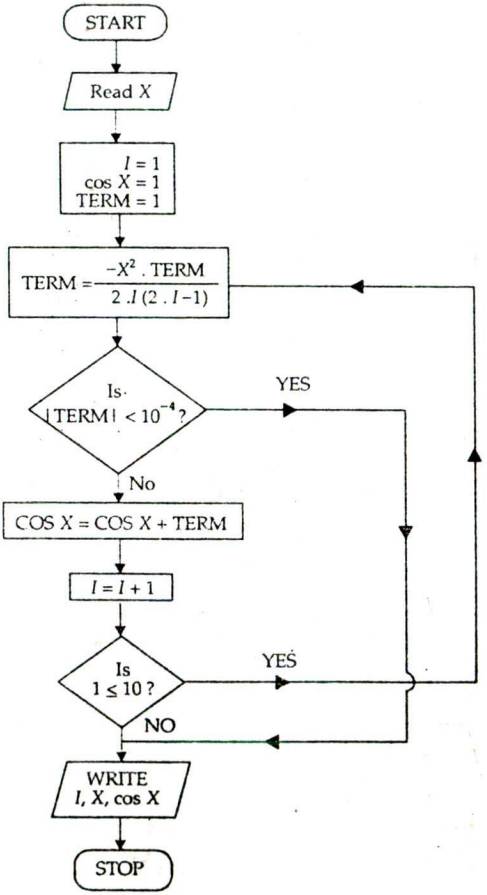


Fig. 12.11

REVIEW QUESTIONS SET

1. Write an algorithm to compute GCD of two positive integer numbers.
2. Write an algorithm to compute the largest of any three positive numbers A, B and C.
3. Write an algorithm to find whether a given number N can be expressed as a perfect square.
4. Write an algorithm to compute the value of K, where $K = \sum_{j=1}^{100} j^2$.
5. Write an algorithm to find the average height of N students in a class.
6. Name and describe the five basic symbols of a flow chart.
7. Draw a flow chart for finding the sum of the series
   $1^3 + 3^3 + 5^3 + ... + 49^3$
8. Consider the equation $Y = 8X^2 - 5X + 4$. Draw a flow chart which calculates Y for values of X from -2 to 2 with step length 0.5.
9. Draw a flow chart that calculates
   SUM = 1/2 + 2/3 + 3/4 + . . . + $(N - 1)/N$, where one first reads N.
10. Draw a flow chart to find the sum of first N odd integers.
11. The sine of an angle may be obtained from
    $$\sin X = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \ldots$$
    where X is measured in radians. Draw a flow chart for a computation that uses the first five terms of this series to calculate the sine of 0.2 radian.
12. Draw a flow chart to produce a table giving the radius and area of circles with radius 5, 10, 15, ..., 50.
13. Draw a flow chart that will compute the following after reading a value of X:
    $$\begin{aligned} Y &= (X^2 + 2X - 1)/5 && \text{if } X < 0 \\ &= X - 3 && \text{if } X = 0 \\ &= X - 14 && \text{if } X > 0 \end{aligned}$$
14. Prepare a flow chart which will read the lengths of three sides of a triangle into three variables A, B and C and then print the area of the triangle.
15. Draw a flow chart for a given set of 50 positive integers to find'
    (*a*) the total number of odd integers
    (*b*) the total number of even integers
    (*c*) the sum of all odd integers
    (*d*) the sum of all even integers
16. Draw a flow chart to find the largest and smallest values of n given numbers.

13

PROGRAMMING LANGUAGES

13.0 INTRODUCTION

A programming language is a set of written symbols governed by a set of rules that instruct the computer hardware to perform specified tasks. A programming language that allows instruction formats, close to English sentences, is called a high level programming language. Major objectives of this chapter are to discuss the distinctive features of some of the important high level programming languages including FORTRAN, ALGOL, COBOL and PL/I.

13.1. FORTRAN

FORTRAN is the acronym of FORmula TRANslation. It is one of the first high level languages to gain widespread use by the scientific and engineering community. FORTRAN was developed for IBM-704 computers by John Backus and a team of 13 other programmers at IBM over a period of three years (1954-57). Thereafter, it has undergone several revisions and improvements. Successive versions have been called FORTRAN II, FORTRAN IV and FORTRAN 77. The documentation of the latest version FORTRAN 77 was started in 1977 and finally approved in March 1978.

Character Set

FORTRAN 77 consists of the following characters:

Character	*Meaning*	*Character*	*Meaning*
0,1,2,....,9	Decimal digits	#	Blank or Space
A,B,C,....,Z	Upper case letters	+	Plus sign
$	Dollar sign	–	Minus sign
'	Apostrophe (single quote)	/	Slash
(	Left parenthesis	,	Comma
)	Right parenthesis	.	Period (decimal point)
*	Asterisk	:	Colon
=	Equal sign		

The digits, together with upper case (capital) letters, are called alpha-numeric characters. Others are special characters. Some FORTRAN 77 compilers accept lower case alphabets (*a, b, ..., z*).

Key Word (Reserved Word)

A set of letters built in the definition of the language is known as key word. Key words are used for special purposes defined by the language. PRINT, STOP, END are key words used in FORTRAN.

13.1.1. Constants

A constant is an entity (i.e., data) which does not change during the execution of a program. A constant may be classified as arithmetic or logical or character.

Arithmetic (Numeric) Constants

An arithmetic constant has a numerical value which may be signed or unsigned. The arithmetic constants may be of the data type: Integer, real, double precision and complex.

Integer constant : An integer constant is a whole number (signed or unsigned) without a decimal point, comma or exponent.

25 is an integer but 25. is not.

The maximum limit to the number of digits varies from one computer system to another. For example, 3154678121 is an invalid integer constant since it exceeds the maximum value $2^{31} - 1$ assuming a 32 bit memory word.

Real constant (Floating point constant) : A real constant is constant with a decimal point and may have a fractional part. A real constant can be written without an exponent and with an exponent.

(a) Without an exponent : A basic real constant consists of a string of decimal digits with a decimal point, prefixed by an optional + or – sign.

Examples of valid real constants are:

(*i*) 1.235
(*ii*) – 0.04
(*iii*) 0.0

Examples of invalid real constants are:

(*i*) 1,234 (commas are not allowed),
(*ii*) 57 (no decimal point is provided),
(*iii*) 3.147689156789 (too many significant digits for a 32 bit word lengtn).

(b) With an exponent : Real constant can be represented in exponent form as $\pm cE \pm e$, where leading sign is optional; *c* represents an integer or a real constant in regular form; $E \pm e$ represents a decimal exponent, *i.e.*, $10^{\pm e}$, *e* is an integer constant (optionally signed) in the range permitted on a particular computer.

Some valid real constants in exponent form are:

(*i*) 11.0 *E* + 07
(*ii*) 1.2 *E* – 8
(*iii*) – 45. *E* + 01

The following are invalid real constants:

(*i*) 0.25 *E* – 420 (exponent too large),
(*ii*) E3 (*E* should precede a real or integer constants),
(*iii*) –7.5 *E* 2.7 (*E* cannot follow a real constant).

Double precision real constants : A real constant is stored in one word and gives an accuracy of seven decimal digits for a 32 bit long memory word. When a real constant is stored in one word, it is called a single precision real constant. For example : 4.472136 given the square root of 20; 0.3333333 gives the approximate value of 1/3 correct up to 7 significant digits. If we require more accurate results, we store a real constant in two consecutive words. Such constants are called double precision real constants.

Double precision real constant represents a real constant stored in two consecutive words and results in at least double the number significant decimal digits (usually 16). A double precision real constant can be written either in basic form or in exponent form. When written in exponent form, character *D* is used in place of *E*.

Complex constants : A complex constant is written as two real or integer constants separated by a comma, surrounded by parentheses pair. First one of these two constants is the real part and second one is imaginary part of the complex number. Following are the examples of valid complex number :

Complex Constant	*Interpretation*
(1.6, 2.6)	1.6 + *i* 2.6
(2.1, 3.1 *E* + 01)	2.1 + *i* 31.0
(0, 5)	+ *i* 5

Logical Constants

There are two logical constants represented by .TRUE. and .FALSE. These are represented internally by the integer values 1 and 0, respectively. The periods before and after the words TRUE and FALSE are required.

Character Constants

A string of alphanumeric and/or special characters placed within apostrophes constitute a character constant. If an apostrophe is to be included in a character string, then two consecutive apostrophes are to be put without a gap. The length of a character string is the number of characters in between the enclosing apostrophes including the blank as well, but presence of two consecutive apostrophes is counted as a single character. The length must be at least one and less than the permitted number (usually 255).

'FOUND' is of length 5
and 'DON'T is of length 5
'SUM = ' is of length 6

13.1.2. Variable

A quantity whose numerical value is temporarily unknown or which may vary during program execution is called a variable and expressed by a variable name. A variable name identified the memory address which stores the current value assigned to the variable. The value assigned to the variable is made available to program at any time during its execution, whenever a reference is made to that variable. The value of the variable can be assigned by Assignment Statement or Data Statement. The names of the Fortran variables are written according to certain set of rules.

1. The name of a variable must begin with an alphabetic character.
2. Subsequent letters may be combination of letters and digits.
3. No special characters are allowed in a name.
4. Variable name must not exceed six characters. (Blank spaces in a variable name are not counted towards the total number of characters). Some FORTRAN compilers permit more than six characters but only first six are significant.
5. Key (Reserved) words are not accepted as variable name.

Valid variable names	*Invalid variable names*
I	N + M (no special character can be used).
C 47	2INT (cannot begin with a number).

13.1.3. Data Type

In FORTRAN, the information or data is divided into six categories. They are termed as data types of the language. Six data types are INTEGER, REAL, DOUBLE, PRECISION, COMPLEX, LOGICAL and CHARACTER.

Type Declaration

Any FORTRAN variable has an implicit type associated with the first letter of its type. A variable beginning with any of the letters, *I, J, K, L, M* and *N* is an integer variable and a variable beginning with any other letters is a real variable.

Integer Variable	*Real Variable*
K	CENTER
ME	DELTA
LAST	AREA

The classification of variable names in integer and real type variables on the basis of the first character is known as **default** or **predefined convention**. To override this predefined convention to have variables of other types, specification statements are used.

13.1.4. FORTRAN Statements

A FORTRAN program is composed of a sequence of step-by-step instructions called statements. There are two general categories of FORTRAN statements: executable and non-executable.

(a) Executable Statements: *Executable statements* instruct the computer to perform (execute) some specific action; e.g., read a value, assign a value, print a result, or stop execution. Assignment statements, transfer statements, control statements, and input/output statements are all examples of executable statements.

(b) Non-executable Statements : *Non-executable statements* direct no specific action but provide information essential to the proper execution of some executable statement. Declaration statements, DATA statements, and FORMAT statements are examples of non-executable statements. With the exception of the FORMAT statement, non-executable statements should appear in the program sequence before any executable statement.

Type Specification Statement

Type specification statements are non-executable statements and are placed in the beginning of the program before the first executable statement. Type specification statements are of two kinds: (*a*) The IMPLICIT statement, (*b*) The EXPLICIT statement.

FORTRAN provides one IMPLICIT type declaration statement which may be used to alter the default first letter rule. The general form is IMPLICIT type (L_1, L_2, ..., L_k) which declares that any variable name starting with any of the letter L_1, L_2, ..., L_k is of the declared type.

The statement IMPLICIT INTEGER (*A* - G, X, Y) means all variable names beginning with *A* through G and X and Y are of integer type.

FORTRAN provides EXPLICIT type declaration statement which can be used to declare a variable name to desired type. The general form of the statement is

Type v_1, v_2, ..., v_k, where v_1, v_2, ..., v_k are variable names whose types are being declared and type is one of INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL or CHARACTER. The following are examples of EXPLICIT type declaration:

INTEGER	Y,	BETA
REAL	LENGTH	I
COMPLEX	C,	ROOT
DOUBLE PRECISION	THETA,	MASS
LOGICAL	FLAG,	STAGE
CHARACTER	A * 2,	NAME * 8

The first statement declares that Y, BETA will be treated as integer variable although the beginning letter is Y and B. The second statement declares LENGTH and I will be treated as real variable although the first letters are L and I. The third statement declares THETA and MASS are double precision real variables. From the fourth statement FLAG and STAGE are variables of logical type. The only value each of them can have either TRUE or FALSE. The fifth statement declares variable A and NAME to be of character type. The length of A and NAME have 2 and 8 characters, respectively. If the length

descriptor is omitted, the length specification given for the list is taken to one. An EXPLICIT declaration overrides not only the predefined convention but also an IMPLICIT statement. Whenever a program has both the IMPLICIT and EXPLICIT statements, the IMPLICIT statement must proceed the EXPLICIT declaration. The statements

```
IMPLICIT        REAL,     (A, B)  LOGICAL (L)
INTEGER         ALPHA,    BETA
```

declare all variable names starting with the alphabet A or B except the variable names ALPHA and BETA of the type REAL.

13.1.5. Mathematical Function

Mathematical expressions often require the evaluation of several common functions, such as square root, sine, cosine, exponential, etc., FORTRAN allows the use of many such functions if the preassigned symbolic name of the function is written and enclosed its argument within parentheses. In fact, the programs already written for evaluation of these functions and incorporated in the functions library of FORTRAN implementations. These are called an intrinsic or built in or Library functions. The advantage is that the programmer need not write program for these commonly used functions. Following are some common examples of built-in functions available in FORTRAN 77.

Functions	*Mathematical notation*	*Symbolic name with arguments*	*Type of function*
Absolute value	$\|x\|$	ABS (*x*)	real
		IABS (*x*)	integer
Square root (arguments ≥ 0)	$\sqrt{x}$	SQRT (*x*)	integer
Exponential	e^x	EXP (*x*)	real
Natural Logarithm (argument ≥ 0)	$\log x$	ALOG (*x*)	real
Logarithm to base 10	$\log_{10} x$	ALOG10 (*x*)	real
Trigonometric Sine (argument in radian)	$\sin x$	SIN (*x*)	real
Trigonometric Cosine (argument in radian)	$\cos x$	COS (*x*)	real
Trigonometric Tangent (argument in radian)	$\tan x$	TAN (*x*)	real
Arc Sine	$\sin^{-1} x$	ASIN (*x*)	real
Arc Cosine	$\cos^{-1} x$	ACOS (*x*)	real
Arc Tan	$\tan^{-1} x$	ATAN (*x*)	real
Hyperbolic Sine	$\sinh x$	SINH (*x*)	real
Hyperbolic Cosine	$\cosh x$	COSH (*x*)	real
Hyperbolic Tan	$\tanh x$	TANH (*x*)	real

13.1.6. Expression

An expression consists of a sequence of operands (constants and variables) separated by suitable operators and parentheses which conform to certain rules for formulating expression. An expression can be any of the following type :

(*i*) Arithmetic expression
(*ii*) Relational expression
(*iii*) Logical expression
(*iv*) Character expression.

Arithmetic Expression

Arithmetic expression is a combination of constants, subscripted or non-subscripted variables, functions, function names, arithmetic operators and parentheses.

The operators used in FORTRAN arithmetic expression are shown as follows:

	Mathematical operator	*FORTRAN operator*
Addition	+	+
Subtraction	−	−
Multiplication	×	*
Division	/	/
Exponential		* *

1. **Integer Expression :** The mathematical expression obtained by combining integer variables and constants with the help of arithmetic operators is known as integer expression. 3 * 4 + *I*, *I* * *J* + 5 and *I* ** *x* + *J* ** 2 are examples of integer expression.
2. **Real Expression :** The mathematical expression obtained by combining real variables and constants with the help of arithmetic operators is known as real expression. *A* + *D* / *E* + *D* ** *F*, *A* ** 3 + *B* ** 3 and *U* + *V*/(*R* + *S*) ** 2 are examples of real expression.
3. **Mixed Mode Expression :** The mathematical expression obtained by combining integer constant and variable and real constant and variable with the help of arithmetic operators is known as mixed mode expression. *I* * *A* + 3.5, *A* * 2 + *K* ** 2 are examples of mixed mode expressions.

The following rules are observed while converting mathematical expression to FORTRAN.

1. Operands must be separated by an arithmetic operator. No operator should be assumed to be present between two operators. For example,

   5 *B* is an invalid expression

   5**B* is an valid expression.

2. Two of the arithmetic operators must not appear side by side. For example,

   3 + − X is an invalid expression.

   3 + (−X) is a valid expression.

Note : An integer variable or integer constant cannot be raised to a real power. 2 ** 2.5, *I* ** 3.0 are invalid.

Non-negative real values can be raised to both real and integer powers. 6.0 ** 3, 7.5 ** 6.0 are valid.

Following are few valid arithmetic expressions in FORTRAN with their mathematical expressions :

Mathematical Expression	*FORTRAN Equivalent*
$(a + b)^3$	(*A* + *B*) * * 3
$\sqrt{a^2 + b^2}$	(*A* ** 2 + *B* ** 2) ** 0.5
$\frac{a+b}{c+d}$	(*A* + *B*) / (*C* + *D*)
$\frac{\sin^3 x}{\cos^2 y}$	sin (*X*) ** 3 / cos (*Y*) ** 2
$\frac{10^6 p}{q(d-r)}$	1.0 *E* + 06 * *P*/(*Q* * (*D* − *R*))
log (|sec *x* + tan *x*|)	ALOG (ABS (1.0/cos (*X*) + TAN (*X*)))

$\frac{\pi r^2 h}{3}$	3.14159 * R ** 2 * H / 3.0
$\frac{e^{x+y}}{4!}$	E X P (X + Y) / (4 * 3 * 2 * 1)
$\frac{1}{\lvert xy \rvert} + \frac{ab}{cd}$	1.0/ABS (X * Y) + (A * B) / (C * D)

Evaluation of Arithmetic Expressions

There is a well defined order in which the various operations are evaluated. Parenthesis can be used to override this order of evaluation. First, the quantities inside the sets of parentheses are computed beginning with the innermost pair of parenthesis. When parenthesis do not completely resolve the ambiguity, the order of computation is carried out from left to right (except for ** where expressions of the form *A* ** *B* ** *C* are evaluated from right to left) with the following priorities for the arithmetic operations:

1. Function references.
2. Exponentiations.
3. Multiplication and division in the order they appear from left to right.
4. Addition and subtraction in the order they appear from left to right.

This is called the **precedence rule** or the **priority rule** or the **hierarchy of operators.**

In evaluating expression of the form *A* op *B*, where op is any of the mathematical operator +, -, *, /, **, the evaluation performed depends on the type of operands *A* and *B*.

(*i*) When *A* and *B* are both integer type, the result is also integer type. This is called integer-mode. At each step of calculation, fractional part is lost. For instance,

```
10/4          =  2
10/5 * 2      =  2 * 2 = 4
2 ** (4/3)    =  2 ** 1 = 2
2 ** 3 ** 2   =  2 ** 9 = 512 (from right to left)
```

(*ii*) When *A* and *B* are real type, the operation will be performed in real mode and the result will be real.

(*iii*) Expressions involving different type of numeric operands are called mixed-mode expressions. When one operand is real and the other operand is an integer, then integer operand is first converted to the type real and calculation is done in real mode. For instance,

```
3.0 + 6/5 = 3.0 + 1 = 3.0 + 1.0 = 4.0
3.0 + 6.0/5 = 3.0 + 6.0/5.0 = 3.0 + 1.2 = 4.2
```

Example **13.1.** *Show the order of evaluation of the following expressions:*

*(i) A ** 3 + 5.0 * B − D / (C + 2.8)*
*(ii) X * Y ** I − ((A − B) / (C − D))*
*(iii) X * Y + Z / (J + K) − I ** (J/4)*

Solution. (*i*) Given

1.	(C + 2.8)	=	e_1
2.	A * * 3	=	e_2
3.	5.0 * B	=	e_3
4.	D / e_1	=	e_4
5.	e_2 + e_3	=	e_5
6.	e_5 − e_4	=	e_6 Final result

(*ii*) 1. $(A - B) = e_1$
2. $(C - D) = e_2$
3. $e_1 / e_2 = e_3$
4. $Y * * I = e_4$
5. $X * e_4 = e_5$
6. $e_5 - e_3 = e_6$ Final result

(*iii*) 1. $(J + K) = e_1$
2. $(J / 4) = e_2$
3. $I * * e_2 = e_3$
4. $X * Y = e_4$
5. $Z / e_1 = e_5$
6. $e_4 + e_5 = e_6$
7. $e_6 - e_3 = e_7$ Final result

***Example* 13.2.** *Given the values I = 4, J = 5 and K = 10, find the value of the expression*

$$I * I/J - K * 2/I$$

Solution. Given expression is

$$\begin{aligned} & I * I/J - K * 2/I \\ = \ & 4 * 4/5 - 10 * 2/4 \\ = \ & 16/5 - 10 * 2/4 \\ = \ & 3 - 10 * 2/4 \\ = \ & 3 - 20/4 \\ = \ & 3 - 5 \\ = \ & -2 \end{aligned}$$

***Example* 13.3.** *Given the values x = 2.0, y = 3.0, C = 0.0, D = 5.0, E = 4.0, find the value of the expression.*

$$\frac{XY^2}{\cos C} + D - E$$

Solution. The FORTRAN equivalent of the given arithmetic expression is

```
    X * Y ** 2 / COS (C) + D - E
=   2.0 * 3.0 ** 2/COS (0.0) + 5.0 - 4.0
=   2.0 * 3.0 ** 2/1.0 + 5.0 - 4.0
=   2.0 * 9.0/1.0 + 5.0 - 4.0
=   18.0/1.0 + 5.0 - 4.0
=   18.0 + 5.0 - 4.0
=   23.0 - 4.0
=   19.0
```

Arithmetic Assignment Statement

The general form of arithmetic assignment statement is

Variable Name = Arithmetic Expression

The numerical evaluation of the arithmetic expression in the right hand side is done and assign the value to the variable on the left hand side.

The 'equals' symbol is not to be interpreted in the same sense in mathematics but must be read as 'is assigned'. The statement $A = B$ assigns the value of B to A leaving B unchanged. The assignment statement

$$Y = A + B$$

instruct the computer to add the values stored in *A* and *B* location and store the sum in the location denoted by *Y* leaving the values of *A* and *B* unchanged. The unique storage location will be assigned to each variable name, the first time that name is used. The important points to be noted are:

(*a*) The mode of the expression and mode of variable name need not be same;
(*b*) The mode of the variable name determines the mode of the result of expression. The expression is first computed in its own mode and then the computed value is converted to the type of the variable on the left if required before assigning to the variable.

The following table illustrates the above points:

Assignment statement	*Value assigned to variable*
C = 5 * 4 + 1	C = 21.0
I = 5.0 / 4.0	I = 1
K = 5 * 3.0/2.0	K = 7
Y = 5.0 * 3.0 ** 2	Y = 45.0
M = 2 * 5 + 3	M = 13

Example 13.4. What will be the value of L after executing the following statement:

```
I = 2
J = 3
K = I * J
K = K + 5
K = K/3
L = I + J + K
```

Solution.

```
I = 2
J = 3
K = I * J = 2 * 3 = 6
K = K + 5 = 6 + 5 = 11
K = K/3 = 11/3 = 3
L = I + J + K = 2 + 3 + 3 = 8
```

Example 13.5. What is the final value of K in the following program: **(AMIE, S '94)**

```
K = 5
I = 3
IA = 252
M = I * 1000 + IA * 10
K = M/1000 + K
```

where K, I, IA, M are integers.

Solution.

```
K = 5
I = 3
IA = 252
M = 3 * 1000 + 252 * 10 = 5520
K = 5520/1000 + 5 = 5 + 5 = 10
```

Example 13.6. *If A = 3.0, B = 5.0, K = 16 and J = 5, the following three statements are executed in the given order:*

```
I = A / B
K = K * 2/B
X = K/J
```

What will be the values of I, K and X after the above three statements are executed? **(AMIE, W '96)**

Solution. $I = 3.0/5.0 = 0$

$K = 16 * 2/5.0 = 32/5.0 = 32.0/5.0 = 6$

$X = 6/5.= 1.0$

Example 13.7. *What is the value of R in each of the following statements:*

(i) $R = I * (J/B)$ *(ii)* $R = (I + 2)/A$

given that $I = 2.0$, $A = 5.0$, $J = 9.0$, $B = 3.0$ **(AMIE, W '94)**

Solution. Since I and J are integer variables and A and B are real in FORTRAN, I and J will store 2.0 and 9.0 as 2 and 9, respectively.

(i)
$R = I * (J/B)$
$= 2 * (9/3.0)$
$= 2 * (9.0/3.0)$
$= 2 * 3.0$
$= 2.0 * 3.0$
$R = 6.0$

(ii)
$R = (I + 2)/A$
$= (2 + 2)/5.0$
$= 4/5.0$
$= 4.0/5.0$
$R = 0.8$

Relational Expression

A *relational expression* is formed when two arithmetic expressions are connected with a relational operator. The arithmetic expression can be an integer, a real or a double precision quantity but cannot be a complex quantity. The six *relational operators* are:

Operator	*Meaning*
.GT.	>
.LT.	<
.GE.	≥
.LE.	≤
.EQ.	=
.NE.	≠

Rules for Forming Relational Expressions

While writing relational expressions, the following rules may be observed:

1. Both the arithmetic expressions, forming a relational expression, must be of the same numeric type, i.e., both of them must be integer or both real or both double precision quantities. For example,

   I.GT. 4 and A.EQ. 5.1

   are valid relational expressions but I.LT. 3.2 is invalid because the arithmetic expression (I) is in integer mode whereas the arithmetic expression (3.2) is in real mode. However, it would be valid on a computer system that accepts mixed mode expressions.
2. A relational expression is a condition and can, therefore, take only two values, TRUE or FALSE.
3. The operators (GT, LT, GE, LE, etc., must be preceded and followed by a period (.).
4. All the relational operators are at the same level of hierarchy and no relational operator has a preference over the other. However, the expression is evaluated from left to right. In a hierarchial order, the relational operators appear next to arithmetic operators.

Logical Expression

The logical expression is formed by combining logical constants, logical variables, relational expressions and logical operators. The values of the logical expression is either .TRUE. or .FALSE.

The logical operators available in FORTRAN are shown in the following table:

Table 13.1 : Logical Operators

Symbol	*Expression*	*Logical Value*
.NOT.	.NOT. *A*	.TRUE., if *A* is .FALSE.
.AND.	*A* .AND. *B*	.TRUE., if *A* and *B* are both .TRUE.
.OR.	*A* .OR. *B*	.TRUE., if at least one of *A* and *B* is .TRUE.
.EQV.	*A* .EQV. *B*	.TRUE., if *A* and *B* are both .TRUE., or are both .FALSE.
.NEQV.	*A* .NEQV. *B*	.TRUE., if one of *A* or *B* is .TRUE. and the other is .FALSE.

Here .NOT. denotes negation operation, .AND. denotes conjunction operator, .OR. denotes disjunction operator, .EQV. denotes equivalence operator and .NEQV. denotes non-equivalence operator.

The order of evaluation of a logical expression is as follows:

(*i*) Evaluation of functions
(*ii*) Arithmetic expressions
(*iii*) Relational operators in the order they appear from left to right. There is no priority among relational operators.
(*iv*) Logical operators in the order .NOT., .AND., .OR., .EQV. or .NEQV. from left to right.

The order of evaluation can be overridden by the use of parentheses.

While writing logical expression, the following rules must be observed:

(*a*) Two logical operators must not be together except in the case when the first operator is .AND. or .OR. or .EQV. or .NEQV. and second operator is .NOT,
(*b*) The operator .NOT. must be followed by an operand but not preceded by an operand, whereas the other operators must be preceded by an operand and followed by an operand or by the operator .NOT.

Example 13.8. *Show step by step evaluation of the expression.*

X .AND Y .OR. *(Z* ** *2 + Z)* .GT. *(3.1 – P)*

Solution.

Z ** $2 + Z = e_1$
$3.1 - P = e_2$
e_1 . GT. $e_2 = e_3$
X .AND. $Y = e_4$
e_4 .OR. $e_3 = e_5$ Final logical result

Example 13.9. *Evaluate*

(*a*) *M* ** *N – M/L* .NE. *A * B * C*
where M = 4, N = 3, L = 7, A = 2.0, B = 4.0 and C = 6.0

(*b*) .NOT. *R* .AND. .NOT. (*K .LE.*
where R = TRUE, K = 4 and B 5'94)

Solution.

(*a*) *M* ** *N – M/L* .NE. *A * B * C*
= 4 ** 3 – 4/7 .NE. 2.0 * 4.0 * 6.0
= 64 – 4/7 .NE. 2.0 * 4.0 * 6.0

= 64 – 0 .NE. 2.0 * 4.0 * 6.0
= 64 – 0 .NE. 48.0
= 64 .NE. 48.0
= .TRUE.

(*b*) .NOT. *R* .AND.NOT. (*K*.LE.*B*)
= .NOT. TRUE. AND. NOT. (4. LE. 2.0)
= .NOT. TRUE. AND. NOT. FALSE.
= .FALSE.AND.TRUE.
= .FALSE.

Example 13.10. *Evaluate*

X + Y .LT. Z + I .OR. I. LT. J .AND. X.NE. Z/K

where X = 10.0, Y = 15.0, I = 3, J = 2, Z = 25.0 *and* K = 5.

Solution. Given expression is

X + Y .LT. Z + I .OR. I .LT. J .AND. X .NE. Z/K
= 10.0 + 15.0 .LT. 25.0 + 3 .OR. 3 .LT. 2 .AND. 10.0 .NE. 25.0/5
= 25.0 .LT. 25.0 + 3 .OR. 3 .LT. 2 .AND. 10.0 .NE. 25.0/5
= 25.0 .LT. 28.0 .OR. 3 .LT. 2 .AND. 10.0 .NE. 25.0/5
= 25.0 .LT. 28.0 .OR. 3 .LT. 2 .AND. 10.0 .NE. 5.0
= .TRUE. OR. 3 .LT. 2 .AND. 10.0 .NE. 5.0
= .TRUE. OR. FALSE. AND. 10.0. NE. 5.0
= .TRUE. OR. FALSE. AND. TRUE.
= .TRUE. OR. FALSE.
= .TRUE.

Logical Assignment Statement

The general form of the logical assignment statement is

Variable Name = Logical Expression

where variable name is declared logical. The execution of this statement causes the logical expression to be evaluated as either .TRUE. or .FALSE. and the value obtained is assigned to the logical variable name. As for example

LOGICAL *A,B,C.*
A = .TRUE.
B = *A*

Character Expression

A character expression is a string of character, constants, a character variable name or a standard function. For example, a character expression can be written as

'HOW' '*b*' DO' '*b*YOU' '*b*DO'

and after evaluation produces the character result

HOW DO YOU DO

The statements

CHARACTER * 10 TM
CHARACTER * 5 WORD

declare variable TM of length 10 and WORD of length 5.

Character Assignment Statement

The general form of a character assignment statement is

Variable = Character Expression

where variable on the left is of character type.

As for example

```
CHARACTER * 3 CH
CH = 'VALUE'
```

will assign 'VAL' to the variable CH.

13.1.7. Input and Output Statements

In order to solve a problem on a digital computer, it may be necessary to transfer the required data to its memory from one of its input units. After a program is executed and the problem is solved, the results are transferred from memory unit to one of its output units to display or print out.

FORTRAN provides two types of Input/Output statements, formatted and list-directed (unformatted or format-free). In the first type, programmer can specify for an input data to be read in a suitable manner and similarly output data can be printed in a desired manner by specifying a suitable format. In the second type, values will be read as presented and printed as available in memory without any conversion and as such the statements are also called list directed I/O statements.

Format-free Input and Output Statements

READ statement : The READ statement is an executable input statement whose general format is

READ *, INPUT list

where list represents a set of variable names, each separated from the other by comma and * defines list directed input and data to the read from a computer terminal and assigned to the variables named in the list. When the statement

READ *, *A*, *I*, *B*

is encountered and 3.8, 2 and 6.10 are assigned values to *A*, *I* and *B*, respectively, they should be typed as

3.8, 2, 6.10

PRINT statement : The PRINT statement is an executable output statement whose general format is

PRINT *, Output list

where the list can include variables, constants, character strings and arithmetic expressions and * signifies list directed output and specifies output to an installation defined output device, normally terminal or line printer.

The values are printed in a standard preformatted way. The number of values that can be printed on one line is also predetermined. The exact format and spacing used in displaying these values are compiler-dependent. However, the computer always begins to print on a new line whenever it encounters a new PRINT statement.

Assuming $X = 4.2$, $Y = 10.6$, $I = 8$, some PRINT statement and the output are shown below:

Statement	*Output*
PRINT *, *X*, *Y*	4.2, 10.6
PRINT *, '*I* =', *I*	*I* = 8
PRINT *, '*X* + *Y* =', *X* + *Y*	*X* + *Y* = 14.8
PRINT *, *I*, 'TIMES'	8 TIMES

STOP Statement

STOP is the executable statement in the program that indicates the logical end of the

program. A program can have more than one STOP statements and it is not necessary for a STOP statement to appear at the end of the program. It may appear anywhere in the body of a program. The STOP statement returns control to the operating system program so that the terminal can be used to execute other jobs.

END Statement

This statement is the last statement of every program unit to indicate the physical end of the program unit. This is a non-executable statement. The statement signals to the compiler that there is no more statement left to be translated.

The following are the differences between the STOP statement and the END statement:

	STOP	*END*
1.	To instruct the computer to stop the execution of the program.	To inform the computer that this is the last line of the program.
2.	STOP statement may occur anywhere in the program.	END statement is the last statement of the program.
3.	One or more STOP statements may occur in the program.	Only one END statement must occur in the program.

FORTRAN Coding Sheet

FORTRAN coding sheet is a sheet of paper with 80 columns in each line, i.e., 80 characters can be written in one line. Programs can be written on the coding sheet first for easy feeding to the computer.

Columns 1 to 5 are used for giving line numbers in FORTRAN, every statement need not be given a line number. Only the statements which are later referred can be given line numbers. The line number is a positive integer with at the most 5 digits.

Column 6 is used for the continuation symbol. If one statement is continued in the next line, then a character must be placed at the sixth column of the later line.

Columns 7 to 72 are used for the statements.

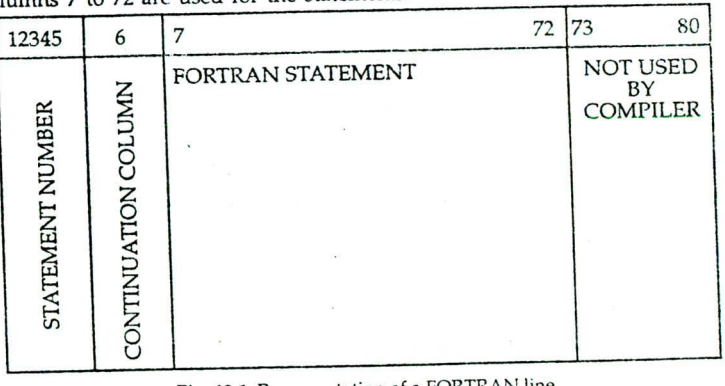


Fig. 13.1: Representation of a FORTRAN line

Columns 73-80 may be used by the programmer for identification of the program or for any other purpose. These columns are ignored by the compiler.

If the character C is written on the first column, the line from column 2 to 80 is treated as a comment line. Comment lines make the program more readable and are used

for the purpose of documentation. The comment lines can be placed at any part of the program and are ignored by the compiler but are printed in the listing of the program.

13.1.8. Transfer of Control Statements

Normally a program is executed statement after statement in sequence until the last statement is encountered. In many situations it becomes necessary to skip certain statements in the program under one set of conditions or to repeat a segment of a program a finite number of times or until a specified condition is met. As a result, a class of statement which makes decisions and act on these decisions are called control statements.

A control/transfer statement is an executable statement that transfers or branches to an another statement designated by a unique label. A control statement may be unconditional or conditional. An unconditional transfer statement always branches to a single designated statement, while a conditional transfer statement may branch to one of several different designated statements, depending upon conditions of the data.

Control Statements

The various control statements can be classified into the following types:

(*i*) Branching Statements

- Unconditional GO TO
- Computed GO TO
- Assigned GO TO
- Arithmetic IF
- Logical IF
- Block IF
  - (*a*) IF – THEN — ENDIF
  - (*b*) IF – THEN — ELSE — ENDIF
  - (*c*) IF – THEN — ELSEIF ... ELSE – ENDIF

(*ii*) Looping Statements

DO

- CONTINUE
- WHILE

Unconditional GO TO Statement

The general form of the statement is

GO TO n

When n is an unsigned non-zero integer number identifying the statement label of the executable statement to which the transfer is to be made. After statement, n is executed control continues execution with the statements following the statement n. Indiscriminate use of GO TO statement may take a program difficult to debug and understand and, therefore, it should be used with care. The blank between GO and TO is optional.

Computed GO TO Statement

The computed GO TO statement causes the transfer of control depending upon the value of an unsigned integer variable or integer expression.

The general form is

GO TO (n_1, n_2, ..., n_k), I

where I is the integer variable (can be integer expression) and n_1, n_2, ..., n_k are statement numbers.

This statement transfers the control to any one of the statements n_1, n_2 ..., n_k depending on the value of I.

If $I = 1$, the control is transferred to statement n_1
If $I = 2$, the control is transferred to statement n_2 ....
If $I = k$, the control is transferred to statement n_k, If $I < 1$ or $I > k$, then this statement itself is ignored and the control goes to the next line.

The Statement

GO TO (10, 20, 30, 40, 50), *A–B*
transfers the control to statement 40 if $A - B = 4$.

Assigned GO TO Statement

Its general form is

GO TO I, (n_1, n_2,...., n_k)

where I is an integer variable and n_1, n_2, n_3,...., n_k are statement numbers.

An ASSIGN statement must be used to assign the value of the integer variable I. The general form of the ASSIGN statement is

ASSIGN n TO I

where n is one of the statements from the set (n_1, n_2,....,n_k)

The statements

ASSIGN 99 TO INDEX
GO TO INDEX (15, 20, 99)

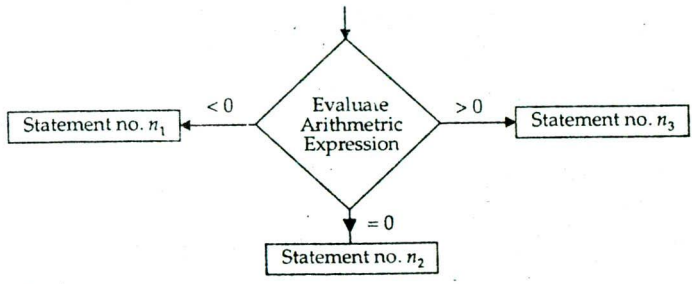
This is equivalent to GO TO 99.

Arithmetic IF Statement

The general form of this statement is

IF (e) n_1, n_2, n_3

where e is a variable or an arithmetic expression of the type integer, real or double precision and n_1, n_2, n_3 (not necessarily distinct) are unsigned statement numbers of executable statements. The control goes to statement labelled n_1, if the value of the variable or expression is negative, to n_2 if the value is zero, and to n_3 if the value is positive.



For example, when the statement

IF ($X - Y$) 10, 20, 30

is executed, the control is transferred to the

statement no. 10 if $X - Y < 0$
statement no. 20 if $X - Y = 0$
statement no. 30 if $X - Y > 0$

Example 13.11. Find the sum of first hundred natural numbers using arithmetic IF statement.

Solution.

```
      ISUM = 0
      ITERM = 1
10    ISUM = ISUM + ITERM
      ITERM = ITERM + 1
20    IF (ITERM - 100) 10, 10, 30
30    PRINT *, ISUM
      STOP
      END
```

Explanation: Sum of the natural number S is initialised as 0 (I SUM = 0), the first term is 1 (I TERM = 1) and a loop is set up for summing all the natural numbers up to and including 100. When the value of I TERM is 101, the control is transferred to 30 to print, the result.

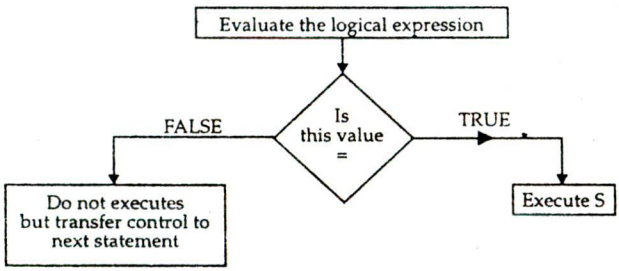
Logical IF Statement

The general form of the statement is

IF (logical expression) S

where S is any executable statement except a DO, another logical IF, IF THEN, ELSE IF, ELSE or ENDIF statement.

First, the logical expression is evaluated. IF the value is .TRUE., the statement S is executed and then goes to the next statement. If the value is .FALSE., the control goes to the next statement after the logical IF statement.



Example 13.12. Write logical IF statement to compute y defined by

$y = 4x + 2.5$ *if* $X \leq 4.1$

$= 8x + 7.1$ *if* $X > 4.1$

Solution.

```
      IF (X .LE. 4.1)  Y = 4.0 * X + 2.5
      IF (X .GT. 4.1)  Y = 8.0 * X + 7.1
```

Example 13.13. Write a FORTRAN program to find the largest of three given numbers A,B,C.

Solution.

```
      READ *, A, B, C
      IF  (A .GT. B) GO TO 20
      IF  (B .GT. C) GO TO 10
      PRINT *, C
10    PRINT *, B
20    IF (A .GT. C) GO TO 30
30    PRINT *, A
      STOP
      END
```

Alternatively,

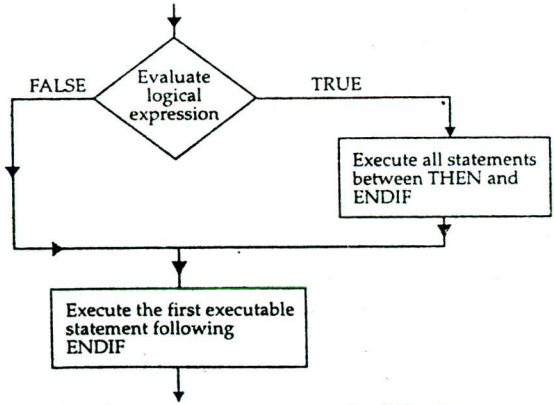
```
READ *, A, B, C
BIG = A
IF (B .GT. BIG) BIG = B
IF (C .GT. BIG) BIG = C
PRINT *, BIG
STOP
END
```

IF–THEN–ENDIF Statement

The general form of this statement is

```
IF (logical expression) THEN
        Statement – 1
        Statement – 2
           ⋮
        Statement – n
ENDIF
```

If the value of the logical expression is .TRUE., then all the statements between THEN and ENDIF are executed and the first executable statement following ENDIF is executed next. But if the expression is .FALSE. then all the statements between THEN and ENDIF are ignored and the control is passed to the next executable statement following ENDIF.



For example, when the given statements are the following:

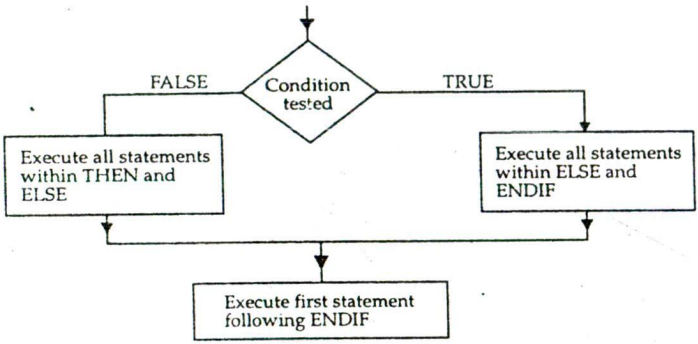
```
IF  (A .GT. 4. 5) THEN
    B = 6.1
    C = 2.0
    E = B/C – A
ENDIF
```

$B = 6.1$, $C = 2.0$ and $E = B/C–A$ will be executed if $A > 4.5$ is .TRUE.

IF–THEN–ELSE–ENDIF Statement (Block IF Statement)

The general form of this statement is

```
IF (condition) THEN
        statements for IF Block
ELSE
        statements for ELSE Block
ENDIF
```



Example :

```
IF (MARKS : LE.50) THEN
    WRITE *, 'FAILED'
ELSE
    WRITE *, 'PASSED'
ENDIF
```

Explanation : If marks ≤ 50, then the value of the logical expression is TRUE, so IF block is executed and the word 'FAILED' is printed. If MARKS > 50, then the value of the expression is FALSE, so ELSE block is executed and 'PASSED' is printed.

Nested Block IF Structures

The structure in which one or more block IF statements are nested inside the IF block and/or ELSE block of a block IF statement is known as Nested block IF structure. This is required when to select one from more than two alternative actions. An example of nested block IF structure is given.

```
IF (P MARK .GE. 80.0) THEN
  PRINT ★, 'EXCELLENT'
ELSE
  IF (P MARK .GE. 70.0) THEN
    PRINT ★, 'VERYGOOD'
    ELSE
      IF (P MARK .GE. 60.0) THEN
          PRINT ★, 'GOOD'
      ELSE
        IF (P MARK .GE. 50.0) THEN
            PRINT ★, 'SATISFACTORY'
        ELSE
            PRINT ★, 'NIL'
```

```
        ENDIF
      ENDIF
    ENDIF
  ENDIF.
```

IF–THEN–ELSEIF .... ELSE–ENDIF Statement

The statement has the form

```
IF (e1) THEN
    block–1
ELSE IF (e2) THEN
    block–2
ELSE IF (e3) THEN
    block–3
        ⋮
ELSE IF (en) THEN
    block – n
ELSE
    block – (n + 1)
ENDIF
```

If the logical expression e_1 has the value .TRUE., the statements within block-1 will be executed, other blocks will be ignored and the control will be transferred to the statement following ENDIF. IF e_1 has the value .FALSE. the control passes to ELSE IF (e_2) THEN. Statements within block-2 are executed if e_2 has the value .TRUE. and the control passes on to the statement following the ENDIF, otherwise the control passes to the next ELSE IF statement and so on.

***Example* 13.14.** *Consider the above example and rewrite the lines by using ELSE IF statement.* (AMIE, S'97)

Solution.

```
IF (P MARK .GE. 80.0) THEN
    PRINT *, 'EXCELLENT'
ELSE IF (P MARK .GE. 70.0) THEN
    PRINT *, 'VERY GOOD'
ELSE IF (P MARK .GE. 60.0) THEN
    PRINT *, 'GOOD'
ELSE IF (P MARK .GE. 50.0) THEN
    PRINT *, 'SATISFACTORY'
ELSE
    PRINT *, 'NIL'
ENDIF
END
```

DO Statement

It is a loop control statement and is used whenever a certain portion of the program is to be repeated a number of times. The general form of the statement is

DO n $I = m_1, m_2, m_3$

where n is the state number of an executable statement, called the DO terminal statement, called the DO terminal statement appearing below the DO statement in the same segment.

The set of all statements following the DO statement and up to and including the DO terminal statement forms the range of the DO statement.

I is called the running variable, control variable or index variable and can be of the type integer or real or double precision; m_1, m_2 and m_3 (optional) are constant, variable or arithmetic expressions of type integer, real, or double precision; m_1 represents the initial value assigned to the DO variable n; it is called initial parameter; m_2 represents the limit (or terminal) value for the DO variable; it is called terminal parameter; m_3 is optional and it represents the increment for the DO variable; it is called incrementing parameter; if m_3 is omitted, it is assigned a default value of one; it is important to note that m_3 must not be assigned a zero value; these parameters may be negative.

The number of times the DO loop will be executed can be found out by using the formula

$$\text{NR} = \frac{m_2 - m_1}{m_3} + 1, \text{ truncated to integer}$$

where NR denotes the number of repetitions (or iterations) of the DO loop. NR will be zero when

$$m_1 > m_2 \text{ and } m_3 > 0$$

and $$m_1 < m_2 \text{ and } m_3 < 0$$

It is a good practice to use CONTINUE statement as a dummy executable statement as the last statement in DO loop domain.

For example, the statement

```
      NSUM = 0
      DO 10 I = 1, 20, 2
      NSUM = NSUM + I
   10 CONTINUE
```

will compute the sum of all odd integers from 1 to 20.

But the statement

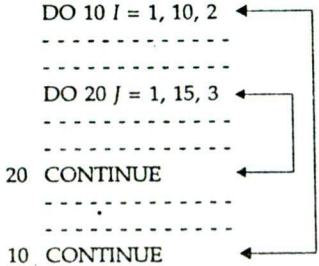
```
      NSUM = 0
      DO 10 I = 1, 20
   10 NSUM = NSUM + I
```

will also compute the sum of all integers from 1 to 20. Note that CONTINUE statement has not been used.

NESTED DO LOOPS

When one DO loop is constructed within another DO loop, it is called a nested DO loop.

Consider the loops

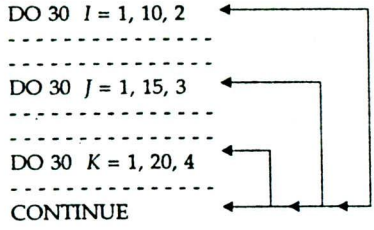


There are two loops here, one lies completely inside the other.

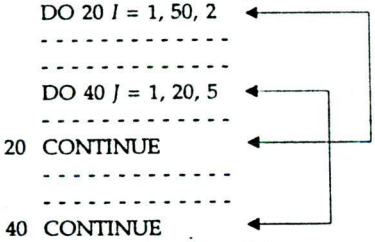
The following rules should be strictly adhered while using nested DO loops.

Rule 1 : In the case of nested DO loops, the same statement can be the terminal statement for more than one DO loop.

Consider the DO loop



Rule 2: DO loops should not overlap. Consider the DO loop



Since the loops overlap, this is invalid.

Rule 3 : The running variable name of the outer DO loop should not be used as running variable for an inner nested DO loop.

For example, consider the loop

```
DO 10 I = 1, 20, 2
- - - - - - - - - - - - -
DO 20 I = 1, 30, 2
- - - - - - - - - - - - -
20 CONTINUE
- - - - - - - - - - - - -
10 CONTINUE
```

Here *I*, the running variable, is used in the outer as well as inner loops. This is invalid.

13.1.9. Arrays and Subscripted Variables

An array is a group of similar elements. The arrays may be one dimensional or multi-dimensional. An array is described by a common variable name. The individual elements of an array are indicated by writing a subscript (or subscripts) in parentheses after the common variable name.

The students in a class may be represented by a common variable name, say ROLL, and different students in the same class may be distinguished by writing different subscripts in parenthesis, after the variable name ROLL. Thus, ROLL (1), ROLL (2), ..., ROLL (100) represents 100 different students in a class. The subscripted variable name

ROLL has only one subscript to differentiate one member of the array from the other and hence are examples of a one-dimensional array.

Consider the tabular form of marks of 3 boys in 4 subjects:

	Maths	*Physics*	*Chemistry*	*Biology*
Siddhartha	98	89	92	90
Arjun	90	85	90	87
Udit	92	87	83	79

If the whole block of data is given the name, MARKS then MARKS (1, 2) represents 89 which is the marks of Siddhartha in Physics, M (3, 3) represents 83 marks of Udit in Chemistry. In general, MARKS (*I*, *J*) represents the marks of *I*th student in the *J*th subject. The variable MARKS (*I*, *J*) is a two-dimensional subscripted variable with two subscripts *I* and *J*.

The subscript must be non-zero unsigned positive integer constant or an integer variable or an expression. When the subscript is an integer expression, it should be in one of the following forms:

(*i*) variable ± constant, (*ii*) constant * variable, (*iii*) constant * variable ± constant.

Examples of valid subscripted variables are: *A*(5), *A*(5 * *N*), *B* (*I*), GRADE (4 * *K* – 2), KOUNT (4 * *I*, *J* + 4).

An array is stored in contiguous words of memory, and that permits an array to be treated as a single entity and facilitates the selection of successive elements of the array.

13.1.10. DIMENSION Statement

The DIMENSION statement is a non-executable statement which specifies the name, dimension and size of those subscripted variables used in the program. It is usually placed at the beginning of the program.

The general format of this statement

DIMENSION v_1, v_2, ...

where *v*'s stand for subscripted variable names followed by parenthesis enclosing integer constants, separated by comma which gives the maximum number of elements in the corresponding array.

Consider the DIMENSION statement

DIMENSION X (10), *M* (6, 7)
COMPLEX *X*

This statement declares that X is a one-dimensional array with the subscript varying from 1 to 10 and instruct the computer to reserve 10 locations in the memory for values of X and X is declared complex. *M* is a two-dimensional array with the first subscript varying from 1 to 6 and the second from 1 to 7 and reserve locations $6 \times 7 = 42$ in the memory.

The matrix $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ can be expressed as

DIMENSION *A* (3, 3)

which implies that the matrix *A* has (3 × 3) elements and reserve locations in the memory as *A* (1, 1), *A* (1, 2), *A* (1, 3), *A* (2, 1) *A*(2, 2), *A* (2, 3), *A* (3, 1), *A* (3, 2) and *A* (3, 3).

Instead of DIMENSION statement, the following statements can also be used to declare the arrays.

1. INTEGER Statement
2. REAL Statement
3. COMPLEX Statement
4. LOGICAL Statement
5. CHARACTER Statement
6. DOUBLE PRECISION Statement

The following are few examples:

```
INTEGER A (50, 50)
```

A is an integer variable with two subscripts. Both subscripts have the maximum value 50.

```
CHARACTER * 25 A (20), B (50)
```

A and *B* are declared as character variables with subscripts. *A* has maximum 20 items and *B* has maximum 50 items each holding 25 character positions.

Implied DO Loops

To read or write subscripted variables, each subscripted variable may be listed, but this is cumbersome. FORTRAN has a very useful feature which simplifies input/output of subscripted variables by allowing an input/output list to contain implied DO loop whose effect is very similar to that of a DO statement.

Suppose we want to read all the entries of one-dimensional array *A* with array length 10. The following DO loops will do the reading operation:

```
    DO 20 I = 1, 10
20  READ *, A (I)
```

The above statements can be replaced by the more concise statement which is called implied DO loop.

```
READ *, (A (I), I = 1, 10)
```

The implied DO loop can also be used for multidimensional arrays. Values can be read either columnwise or rowwise. For example, consider a two-dimensional array *A* (*I*, *J*), where *I* varies from 1 to 20 and *J* varies from 1 to 10. If we want to read all the elements of the array, then

```
    DO 20 I = I, 20
    DO 20 J = J, 10
20  READ *, A (I, J)
```

The above statement can be written as

```
READ *, ((A (I, J), J = 1, 10), I = 1, 20)
```

13.1.11. Format Specification

We have discussed the format-free input and output statements in order to write simple program. But often we require to input/output data result in a particular fashion. The FORMAT statement is used in conjunction with either READ or PRINT statement to specify the physical arrangements of data on an input/output medium. The format statement in a non-executable statement and its general form is

$$n \text{ FORMAT } (F_1, F_2, \ldots, F_m)$$

where n is the number of the FORMAT statements specified in the associated READ or WRITE statement and $F_1, F_2, \ldots, F_m$ are called format specifiers, format descriptors, field specifiers or field descriptors.

The field descriptors of the following types are discussed in detail:

(*a*) Field descriptors for numeric data
  (*i*) *I*
  (*ii*) *F*
  (*iii*) *E*
  (*iv*) *D*

(*b*) Field descriptors for alphanumeric data
  (*i*) *H*
  (*ii*) ' '
  (*iii*) *A*

(*c*) Field descriptor of data editing
  (*i*) *X*

13.1.11.1. Field Descriptors for Numeric Data

I Format : The *I* field descriptor is used to transfer integer data. Its general form is

r I w

where *r* represents an integer-repetition number indicating how many times the field descriptor is to be used (omitted when 1), *I* specifies integer type and *w* is an unsigned positive integer constant and specifies the width of the field in digits and includes the sign and any blanks within the field. The number is placed at the right, with blanks inserted at the left, if necessary, i.e., the number must be right adjusted. In case the *w* positions are not sufficient to accommodate the output value, the output field *w* is filled with asterisks.

Example :

```
      READ 10, L, M, N
   10 FORMAT (I 4, 2 I 6)
```

The formatted READ statement initiates the control of the FORMAT specification statement number 10 and the variable *L*, *M* and *N* get their values from 1–4, 5–10, 11–16 column positions. The input record is prepared as follows:

b – 9 bb 57 bb 9 + bbbb 3 .....

The values assigned to the integer variables *L*, *M* and *N* would be

$L = -90$, $M = 570009$, $N = +3$

Note :

(*i*) One position must be reserved for the minus sign if the integer is negative.
(*ii*) Any blanks within the integer field, leading, intermediate or trailing are interpreted as zero.

Example :

```
      PRINT 10, L, M, N
   10 FORMAT (I 6, I 10, I 4)
```

If the values of *L*, *M*, *N* in the internal storage are – 479, 3 and 67239, the output will appear as

*b b – 479 b b b b b b b b b 3 * * * **

F Format : The *F* field descriptor is used to transfer the real values (single precision or double precision). Its general form is

r F w . d

where *r* is the repeat count; *F* indicates that the data appearing within the field is to be treated as a real number; *w*, an unsigned integer constant indicating the total number of characters in the field including the sign, decimal point and any blanks; *d*, an unsigned integer constant and specifies the number of places after the decimal point.

Example :

```
READ 10, X, Y
10 FORMAT (F 4.2, F 8.3)
```

If the input record is prepared as

3.54 *bb* – 6.578

The values assigned to real variables X, Y would be

$X = 3.54$ and $Y = -6.578$

The statements

```
PRINT 10, A, B, C
10 FORMAT (F 10.3, F 9.6, F 6.2)
```

with the stored values of variables as

$A = 4.132643$

$B = 2.23$

$C = 25679.32$

will produce the output

4.133 2.230000****

E Format : *E* field descriptor is used to transfer real numbers in exponent form. Its form is

rEw.d

where *E* indicates that the data field is to be considered as a real number (mantissa) with an exponent, *w* indicates the total number of characters in the field including sign, decimal point, exponent symbol *E*, and any blanks occupy *w* columns, and *d* represents the number of digits after the decimal point in the mantissa.

Example : Assuming that the input data is:

213.6*E*21 ♯ 6.4

The statements

```
READ 10, A, B
10 FORMAT (E 8.1, F 4.1)
```

will assign 213.6*E*21 to *A* and 6.4 to *B*

on output *d* significant digits are printed to the right of the decimal point and if there are more than this in the number, then the fraction will be rounded. In an *E*-field descriptor $w \geq d + 7$ to make room for a sign, at least one digit to the left of the decimal point, the decimal point, the exponent symbol *E*, sign of the exponent, and the two digit exponent.

Example : The following are some real numbers, a sample *E*-field descriptors, and the way in which the number would be printed.

Specification	*Real constant*	*Output*
E 9.2	456712.23	♯ 0.45*E* ♯ 06
E 8.4	6789.231	* * * *
E 15.6	537.67182	♯ ♯ ♯ 0.537672 *E* + 03

D Format : The *D* field descriptor is generally used when dealing double precision variables. The *D* format is identical to the *E* format except for the use of *D* in place of *E*. The general form of a *D* format is

r D w. d

13.1.11.2. Field Descriptors for Alphanumeric Data

H Format : The *H* field descriptor is used for including alphabetic, numeric and other data types in the printed output. The general form is

nH string of characters

where *n*, which is an integer constant, indicates the number of characters (including blanks) in the character string.

Example : Assuming $X = 75.132$, the statements

```
      PRINT 10, X
   10 FORMAT (11H THE ␣ ROOT ␣ IS, F 10.3)
```

will print

```
THE ROOT IS 75.132
```

Apostrophe Form : The alternative method of having output of character information is enclosing the character string between two single quotes. The statements

```
      PRINT 10, X
   10 FORMAT ('THE ␣ ROOT ␣ IS')
```

will print the heading only.

A Format : This field descriptor is used when the input or output data is of character type. The general form is

Aw

The character *A* indicates that the field contains alphanumeric characters and *w* specifies the width of the field.

For Input

(*a*) If the field is too small, the leftmost characters are ignored,
(*b*) If the field is too large, blanks are filled on the right.

For Output

(*a*) If the field is too large, leading blanks appear,
(*b*) If the field is too small, only the first characters from the left are output.

Example : The following are some data items, a sample *A*-field descriptors, and the way in which the data item would be printed.

FORMAT	*Data Item*	*Output*
A 10	CALCUTTA	␣ ␣ CALCUTTA
A 6	MADRAS	MADRAS
A 4	BOMBAY	BOMB

13.1.11.3 Field Descriptor of Data Editing

X Format: The X descriptor is used to control positioning within an input or output statement. The general form is

nx

This causes *n* columns to be skipped on input or output from the current position.

Skip Specification (/) : If there are *n* slashes in input format, *n* lines (including the line being read currently) are skipped and $(n + 1)$th line is read. On output, *n* lines (including the current line) are skipped and the next output starts at the $(n + 1)$th line.

Example 13.15. *Write a program to convert a decimal number $n \leq 15$ into an equivalent binary number.* **(AMIE, W '94)**

Solution. When $n \leq 15$, the corresponding binary number will have maximum 4 bits.

```
      INTEGER B
      DIMENSION B (4)
      READ *, N
      IF (N.LT.00) GO TO 50
```

```
      DO 10 I = 1, 4
      B (I) = 0
10    CONTINUE
      DO 20 J = 4, 1, -1
      M = N/2
      B (J) = N - M * 2
      N = M
      IF (N.LT.1) GO TO 30
20    CONTINUE
30    PRINT *, N, B, L = 1, 4)
50    STOP
      END
```

Example 13.16. *Write a program to add even and odd numbers separately from a given series of integers.*

Solution.

```
      DIMENSION L (50)
      READ (*, 10) N
10    FORMAT (I 4)
      DO 20 I = 1, N
20    READ (*, 10) . L (I)
      SUM EV = 0
      SUM OD = 0
      DO 60 I = 1, N
      C = L (I)
      B = C/2
      K = B
      D = K
      If (B - D) 30, 40, 30
30    SUMOD = SUMOD + C
      GO TO 60
40    SUMEV = SUMEV + C
60    CONTINUE
      PRINT (*, 70) SUMEV, SUMOD
70    FORMAT (2X, 2I12)
      STOP
      END
```

Example 13.17. *Write a FORTRAN program to read the length of three sides of a triangle and to compute the area of the triangle.* **(AMIE, S '97)**

Solution.

```
C     CALCULATION OF AREA OF A TRIANGLE
      READ (*, 10) A, B, C
10    FORMAT (3 F 10.3)
      S = (A + B + C)/2.0
      AREA = SQRT (S * (S - A) * (S - B) * (S - C))
      WRITE (*, 20) A, B, C, AREA
20    FORMAT (1X, 3F 15.3, 5X, 'AREA =', F 15.3)
      STOP
      END
```

Example 13.18. *Write a FORTRAN program to find out the HCF of two positive non-zero integers A and B.* **(AMIE, S '97)**

Solution.

```
C     CALCULATION OF HCF OF TWO POSITIVE
C     NON-ZERO INTEGERS
      READ (*, 10) N1, N2
10    FORMAT (2 I 3)
      KOPY 1 = N1
```

```
      KOPY 2 = N2
   15 IR = KOPY 1 - KOPY 1/KOPY2 * KOPY 2
      IF (IR.EQ.0) GO TO 20
      KOPY 1 = KOPY 2
      KOPY 2 = IR
      GO TO 15
   20 KHCF = KOPY 2
      WRITE (*, 25) N1, N2, KHCF
   25 FORMAT (1X, 2I3, 5X, 'HCF =', KHCF)
      STOP
      END
```

Example 13.19. *Write a program of converting a temperature from centigrade to Fahrenheit.*

Solution. The formula for converting a temperature from centigrade to Fahrenheit is

$$F = \frac{9}{5} C + 32$$

```
C     PROGRAM FROM CENTIGADE TO FAHRENHEIT
      READ (*, 10) C
10    FORMAT (F 10.2)
      F = 9.0/5.0 * C + 32.0
      PRINT (*, 20) F
20    FORMAT (2X, 'FAHRENHEIT =', F10.2)
      STOP
      END
```

Example 13.20. *Write a program to find the largest of three numbers.* **(AMIE, W '95)**

Solution.

```
C     PROGRAM TO FIND OUT THE LARGEST
C     OF THREE NUMBERS
      READ (*, 10) A, B, C
10    FORMAT (3F 7.3)
      BIG = A
      IF (BIG .LT. B) BIG = B
      IF (BIG .LT. C) BIG = C
      PRINT (*, 20) BIG
20    FORMAT (5X, F 7.3)
      STOP
      END
```

Example 13.21. *Write a program to reverse a positive integer number.*

Solution.

```
C     PROGRAM TO REVERSE A NUMBER
      READ (*, 10) N
10    FORMAT (I 4)
      IMAGE = 0
      KOPY = N
15    IQ = KOPY/10
      IR = KOPY - IQ * 10
      IMAGE = IMAGE * 10 + IR
      IF (IQ. EQ.0) GO TO 20
      KOPY = IQ
      GO TO 15
20    PRINT (*, 30) IMAGE
30    FORMAT (5X, 'REVERSE NUMBER =', I4)
      STOP
      END
```

Example 13.22. *Write a computer program to convert a positive decimal number into an equivalent binary number.*

Solution. We assume that the equivalent binary corresponding to given decimal number will have maximum 16 bits.

```
      INTEGER A (16)
      READ *, N
      DO 10 I = 1, 16
      A(I) = 0
   10 CONTINUE
      DO 15 J = 16, 1, -1
      M = N/2
      A (J) = N – M * 2
      N = M
      IF (N .EQ. 0) GO TO 20
   15 CONTINUE
   20 PRINT *, (A (K), K = 1, 16)
      STOP
      END
```

Example 13.23. *Frame a FORTRAN program to calculate the roots of a quadratic equation* $AX^2 + BX + C = 0$ *with the coefficients A, B and C as inputs. Your program must handle all possible values of A, B and C, and generate appropriate output and/or messages.* **(AMIE, S '97)**

Solution.

```
C     PROGRAM TO SOLVE QUADRATIC EQUATION
      READ (*, 10) A, B, C
   10 FORMAT (3F 10.2)
      IF (A .EQ. 0) GO TO 50
      D = B * B – 4.0 * A * C
      IF (D) 20, 30, 40
   20 RP = – B /(2.0 * A)
      EMP = SQRT (–D)/(2.0 * A)
      WRITE (*, 15) RP, EMP
   15 FORMAT (5X, 'COMPLEX ROOTS'
                  5X, 'REAL PART =', F10.3, 5X, 'IMAGINARY PART=', F10.3)
      STOP
   30 X1 = – B/(2.0 * A)
      WRITE (*, 25) X1
   25 FORMAT (SX, 'EQUAL ROOT =', F 10.3)
      STOP
   40 X1 = (– B + SQRT (D))/(2.0 * A)
      X2 = (– B – SQRT (D))/(2.0 * A)
      WRITE (*, 35) X1, X2
   35 FORMAT (5X, 'TWO DISTINCT REAL ROOTS'
                  5X, F10.3, 5X, F10.3)
      STOP
   50 X = – C/B
      WRITE (*, 45) X
   45 FORMAT (5X, F 10.3)
      STOP
      END
```

Example 13.24. *Write a FORTRAN program to find if a given number is a prime number or not.* **(AMIE, W '96, W '97)**

Solution.

```
      READ (*, 10) N
   10 FORMAT (I 4)
      I = 2
   20 IF (I .GT. N/2) GO TO 40
      IR = N – N/I * I
      IF (IR .EQ. 0) THEN
      WRITE (*, 30)
   30 FORMAT (5X, 'GIVEN NUMBER IS NOT PRIME')
```

```
      ELSE
      I = I + 1
      GO TO 20
      ENDIF
40    WRITE (*, 50) N
50    FORMAT (5X, 'PRIME NUMBER =', I 4)
      STOP
      END
```

Example 13.25. Given two one-dimensional arrays named A and B of seven elements each, suppose that A array and B array are to be read separately with an F10.0 field specification. Write a program to read the arrays, then compute and print values of ANORM with E 20.8 field specification where

$$ANORM = \sqrt{\sum_{i=1}^{7} A_i B_i}$$ **(AMIE, S '94)**

Solution.

```
C     PROGRAM TO COMPUTE ANORM
      REAL A (7), B(7)
      READ (*, 10) (A(I), I = 1, 7)
      READ (*, 10) (B(I), I = 1, 7)
10    FORMAT (F 10.0)
      ANORM 1 = 0
      DO 20 I = 1, 7
      ANORM 1 = ANORM 1 + A (I) * B (I)
20    CONTINUE
      ANORM = SQRT (ANORM 1)
      PRINT (*, 30) ANORM
30    FORMAT (5X, E 20.8)
      STOP
      END
```

Example 13.26. Write a program to determine whether a given matrix is symmetric. **(AMIE, W '95)**

Solution.

```
C     PROGRAM CHECK SYMMETRY OF A GIVEN
C     MATRIX
      DIMENSION A (100, 100)
      READ *, N
      READ *, (A (I, J), J = 1, N) I = 1, N)
      DO 10 I = 1, N
      DO 10 J = 1, N
      IF (I .NE. J) THEN
      IF (A (I, J) .NE. A (J, I) GO TO 20
10    CONTINUE
      PRINT * 'THE MATRIX IS SYMMETRIC'
      STOP
20    PRINT *, 'THE MATRIX IS NOT SYMMETRIC'
      STOP
      END
```

Example 13.27. Write a program to find the mean and standard deviation of n numbers.

Solution. If $x_1, x_2, \ldots, x_n$ be n numbers. Then mean $\bar{x} = \frac{\Sigma x_i}{n}$ and standard deviation $= \sqrt{\frac{\Sigma x_i^2}{n} - (\bar{x})^2}$.

```
C   PROGRAM TO CALCULATE MEAN AND
C   STANDARD DEVIATION
    READ (* , 10) N
10  FORMAT (I 4)
    READ (*, 20) (X (I), I = 1, N)
20  FORMAT (F 10.3)
    SUM = 0
    SUM SQ = 0
    DO 30 I = 1, N
    SUM = SUM + X (I)
    SUM SQ = SUM SQ + X (I) ** 2
30  CONTINUE
    XN = N
    XMEAN = SUM/XN
    XSDV = SQRT (SUM SQ/XN – XMEAN ** 2)
    WRITE (*,40) XMEAN, XSDV
40  FORMAT (5X, 'MEAN =', F10.3, 5X,
                'STD DEV =', F 10.3)
    STOP
    END
```

Example 13.28. *Three positive integers a, b, c, where $a < b < c$, form a Pythagorian triplet if $a^2 + b^2 = c^2$. For example, 3, 4, 5 form a Pythagorian triplet.*

Write a FORTRAN program that generates all Pythagorian triplets a, b, c, where $a, b \leq 25$. **(AMIE, S '97)**

Solution.

```
C   PYTHAGORIAN TRIPLETS
    DO 10 L = 1, 23
    J = L + 1
    DO 10 M = J, 24
    I = M + 1
    DO 10 N = I, 25
    IF (L * L + M * M) .NE. (N * N)
    GO TO 10
    ELSE
    WRITE (*, 15) L, M, N
15  FORMAT (5X, 3 (5X, I2))
10  CONTINUE
    STOP
    END
```

Example 13.29. *Write a FORTRAN program to sort in descending order an array of 10 numbers.* **(AMIE, W '96)**

Solution.

```
C   PROGRAM FOR SORTING IN DESCENDING ORDER
    DIMENSION A (10)
    READ (*, 10) (A (I), I = 1, 10)
10  FORMAT (F 10, 3)
    DO 20 I = 1, 9
    DO 20 J = I + 1, 10
    IF (A (I) .GE. A (J)) GO TO 20
    R = A (J)
    A(J) = A(I)
    A (I) = R
    ENDIF
20  CONTINUE
    WRITE (*, 30) (A(I), I = 1, 10)
30  FORMAT (5X, 10F 10.3)
```

```
      STOP
      END
```

Example 13.30. *Write a program segment to evaluate the following integral by Trapezoidal rule.*

$$I = \int_a^b f(x)\,dx$$

Solution. Trapezoidal formula is

$$\int_a^b f(x)\,dx = \frac{h}{2}\,[f(a) + f(b) + 2\,(f(a+h) + f(a+2h) + \ldots + f(a+(n-1)h)]$$

```
C     PROGRAM FOR TRAPEZOIDAL RULE
      READ (*, 10) X0, XN, N
   10 FORMAT (2F 10.2, I3)
      SUM = 0.0
      H = (XN – X0)/N
      IF (N .GT. 1) GO TO 20
      AREA = (H/2.0) (F (X0) + F (XN))
      GO TO 35
   20 DO 30 I = 2, N –1
      SUM = SUM + F (X0 + IH)
   30 CONTINUE
      AREA = (H/2.0) * (F (X0) + 2.0 * SUM + F (XN))
   35 WRITE (*, 40) AREA
   40 FORMAT (5X, 'AREA =', F 10.5)
      STOP
      END
```

Example 13.31. *Write a program to evaluate the integral* $\int_a^b f(x)\,dx$ *by Simpson's one-third formula.* **(AMIE, W'97)**

Solution. Simpson's one-third formula is

$$\int_a^b f(x)\,dx = \frac{h}{3}\,[f(x_0) + f(x_n) + 2\,\{f(x_0+2h) + f(x_0+4h) + \ldots\} + 4\,\{f(x_0+h) + f(x_0+3h) + \ldots\}]$$

```
C     PROGRAM OF SIMPSON'S RULE
      READ (*, 10) X0, XN, N
   10 FORMAT (2F 10.3, I3)
      H = (XN – X0)/N
      SUM = F (X0) – F(XN)
      DO 20 I = 1, N–1, 2
   20 SUM = SUM + 4 * F (X0 + I * H) + 2 * F (X0 + (I + 1) * H)
      AREA = SUM * H/3
      WRITE (*, 30) AREA
   30 FORMAT (5X, 'AREA=', F 10.5)
      STOP
      END
```

Example 13.32. *Write a program to calculate and print the value of cos x correct up to four decimal places using the series*

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots.$$

Solution. Here $T_1 = 1$ and $T_{n+1} = -\dfrac{x^2}{2n(2n-1)}\,T_n$, $n = 1, 2, 3$ ....

```
C     PROGRAM FOR COS X SERIES
      READ (*, 10) X
```

```
10  FORMAT (F 10.5)
    COS X  = 1.0
    TERM = 1.0
    N = 0
20  N = N + 1
    TERM = – X * X * TERM / (2 * N * (2 * N – 1))
    IF (ABS (TERM) .LT. 0.00005) GO TO 30
    COS X = COS X + TERM
    GO TO 20
30  PRINT (*, 40) COS X
40  FORMAT (5X, 'X =' ; F 10.5, 5X, 'COS X =', F10.10)
    STOP
    END
```

Example 13.33. *Write a FORTRAN program to solve the following differential equation:*

$$\frac{dy}{dx} = \frac{y^2 + xy - 2x^2}{2xy - 15}$$

by fourth-order Runge-Kutta method with initial conditions $x_0 = 0$, $y_0 = 2$. *The program should print the values of y for* $x = 0.1, 0.2, \ldots, 2.0$.

Solution.

```
C   PROGRAM OF RUNGE-KUTTA METHOD
C   OF FOURTH ORDER
    REAL K1, K2, K3, K4
    F (X, Y) = (Y * Y + X * Y – 2.0 * X *  X) / (2.0 * X * Y – 15.0)
    X = 0.0
    Y = 1.0
    H = 0.1
    DO 10 I = 1, 20
    K1 = H * F (X, Y)
    K2 = H * F (X + H/2, Y + K1/2)
    K3 = H * F (X + H/2, Y + K2/2)
    K4 = H * F(X + H, Y + K3)
    Y = Y + (K1 + 2 * K2 + 2 *  K3 + K4) /6.0
    X = X + H
    PRINT *, X, Y
10  CONTINUE
    STOP
    END
```

13.1.12. Subprograms

Often at several points in a program identical type of computation with differing data are required. For example, areas of triangles may have to be worked out many times with different values of sides. It would save effort and time in writing and checking the program if such set of computations could be written only once and referred to whenever required. Some common mathematical functions like sine, cosine, log, etc., have been programmed and can be referred to whenever required and they are known as library functions. Library functions are examples of one type of subprogram. It is also possible for the users to define and write their own subprograms which may subsequently be used in the same way as library functions.

A subprogram is a completely independent program used by a main program or by other subprogram. Three forms of subprograms which can be written as:

(*a*) Statement function statement;

(*b*) FUNCTION subprogram; and

(*c*) SUBROUTINE subprogram.

A statement function statement is an integral part of a program whereas function subprogram and subroutine subprogram are complete programs in themselves and are compiled separately.

Purpose and Uses of Subprogram

A computer program can be decomposed into number of subprograms which have the following advantages in program construction:

1. The same subprogram can be used in a number of places in a program.
2. Each subprogram can be compiled and tested separately.
3. The debugging of a program is easier and faster since it is easier to trace through subroutines than it is through one large program.
4. The maintenance and documentation of a large program is much simplified by the maintenance and documentation of individual subprogram.
5. The modification of a program can be done by inserting, deleting and amending of individual subprogram.
6. Program becomes more understandable both by the developer and the user.
7. It allows different programmer to work on different subprograms concurrently.

The Statement Function

When a single arithmetic expression is evaluated at several points in the same program but with different variables on each occasion, a statement function is used.

The general form of the statement function is

$$\text{Name } (a_1, a_2, a_3, \ldots, a_p) = e$$

where Name is the statement function name, $a_1, a_2, a_3, \ldots, a_p$ are distinct non-subscripted variable names representing the dummy arguments, and e is a valid arithmetic or a logical expression that does not contain any subscripted variables.

A statement function definition which looks somewhat like an assignment statement may not be confused with it. The former is a non-executable statement whereas the latter is an executable statement. The statement function can be referenced in the expression like other library functions.

Examples of statement function definitions are

```
ROOT   (A, B, C) = (-B + SQRT (B * B - 4.0 * A * C)) / (A + A)
  ↑        ↑                          ↑
Function  Arguments          Arithmetic expression
Name
```

```
F (X, Y) = X**3 + X ** 2 * Y + X * Y ** 2
```

Example **13.34.** *Define the statement function*

$$DIST\ (x, y, z) = \sqrt{x^2 + y^2 + z^2}$$

and, using this definition, write arithmetic assignment statement to compute

$$\alpha = [(a + 6)^2 + (b + 7)^2]^{5/2}$$

$$\beta = \log [\sqrt{a^2 + b^2 + c^2}\ /(4 + \sqrt{p^2 + q^2 + r^2})].$$

Solution. The statements to compute α and β may be written as follows:

```
DIST (X, Y, Z) = SQRT (X * X + Y * Y + Z * Z)
ALPHA = DIST (A + 6.0, B + 7.0, 0) * * 5
BETA = ALOG (DIST (A, B, C)/(4.0 + DIST (P, Q, R)))
```

Explanation. In this example, the variables X, Y, and Z occurring in the statement function definition

```
DIST (X, Y, Z) = SQRT ( X * X + Y * Y + Z * Z)
```

are dummy arguments. At the execution stage of the statement

ALPHA = DIST (*A* + 6.0. *B* + 7.0, 0) * * 5

the dummy arguments X, Y, and Z will be replaced by the actual arguments *A* + 6.0, *B* + 7.0 and 0, respectively, the function DIST will be evaluated, and the result of the evaluation will be assigned to ALPHA. A similar explanation holds for the statement that evaluates BETA.

Rules. All statement function definition must appear before the first executable statement of the program.

The names of the function should be formed according to the same rules that apply to a variable name. The name of the function can be defined either with the rules for implicit type specification for variables or with explicit type specification. The name of the function must be different from the name of library function.

The actual arguments listed in the calling statement must agree in number, type and relative position with dummy arguments listed in a defining statement. The program will make the computation defined by the function using the values of the actual variables in the argument list and the result is returned to the calling statement.

The statement function is not a true subprogram since it is located in the main memory, but for simple calculation it serves the same purpose. The arithmetic statement function definition is limited to function that can be defined by one statement only. Another limitation is that it computes and returns only one value.

Function Subprogram

The FUNCTION subprogram consists of several statements instead of a single statement and is complied after the main program.

The general form is

FUNCTION Name (x_1, x_2, . . . , x_n)
Name = expression (at least one statement of this form in program)
RETURN (at each logical exits from program)
END

Name is a variable name consisting up to six characters, first letter determines type unless type is declared. If type is declared, the function statement is of the form

Type FUNCTION Name (x_1, x_2, . . . , x_n)

x_1, x_2, . . . x_n are dummy arguments which actually agree with the actual arguments in number, order and type. When the name of a FUNCTION subprogram is encountered in the calling program, control is transferred to the subprogram. The dummy arguments are replaced by the actual arguments in the body of the FUNCTION subprogram, and the execution of the subprogram is carried out. When this execution is over the RETURN statements return control to the particular referencing point in the calling program.

Important points

1. A FUNCTION statement must be the first statement of a subprogram.
2. FUNCTION subprograms are placed separately, following the END statement of the calling program.
3. A function subprogram may use other function subprograms or subroutine subprograms but must not in any case refer to itself.
4. A FUNCTION subprogram is called by referring to it by name in a single arithmetic statement in the calling program.

Example *13.35. Write a function subprogram that reads values for A and B, calls the subprogram to sum them, and prints the values for A, B and the sum.*

Solution.

Calling program	Function Subprogram
READ * A, B	FUNCTION ADD (X, Y)
SUM = ADD (A, B)	ADD = X + Y
PRINT * A, B, SUM	RETURN
STOP	END

Explanation: If the calling program reads a value of 3.0 for A and 2.5 from B, these values will replace the dummy arguments X, Y in FUNCTION ADD which sums them and returns the sum 5.5 to be stored in SUM by the calling program.

Example 13.36. *Evaluate a quadratic function given by* $g(x) = x^2 - 2x + 1$ *for* $x = 1, 2, 3, \ldots, 10$ *using a FUNCTION subprogram.* **(AMIE, S '96)**

Solution.

```
      PROGRAM MAIN
      REAL X, GX
      DO 100 X = 1.0, 10.0, 1.0
      GX = G (X)
      PRINT *, X, GX
  100 CONTINUE
      STOP
      END
      REAL FUNCTION G (X)
      G = X * X – 2 * X + 1
      RETURN
      END
```

Example 13.37. *(a) Write a FORTRAN function FACT to calculate the factorial of a given number; (b) Write another function NCR (which would be calling FACT), compute*

$$NCR = \frac{N!}{R\,!(N-R)!}$$

for any given N and R; and (c) Write a main program which will read in the values M, K and X, and print out the sum:

$$S = M_{C_0} + M_{C_1}X + M_{C_2}X^2 + \ldots + M_{C_R}X^K$$ **(AMIE, W '93)**

Solution.

(*a*)

```
      INTEGER FUNCTION FACT (N)
      INTEGER N, I
      FACT = 1
      DO 100 I = 2, N
      FACT = FACT * I
  100 CONTINUE
      RETURN
      END
```

(*b*)

```
      FUNCTION NCR (N, R)
      INTEGER N, R, NCR
      IF (R .EQ. 0) THEN
      NCR = 1
      RETURN
      ENDIF
      NCR = FACT (N)/((FACT (R) * FACT (N – R))
      RETURN
      END
```

(*c*)

```
      PROGRAM MAIN
      INTEGER S, M, K
      REAL X
      READ *, M, K, X
      S = NCR (M, 0)
```

```
      DO 100 K = 1, M
      S = S + NCR (M, K) * X ** K
  100 CONTINUE
      PRINT *, S
      STOP
      END
```

Example 13.38. *Write a FUNCTION subprogram ANGLE A (A, B, C) which would find the angle opposite to side A of a triangle whose sides are respectively A, B and C. Use this subprogram to find the sum of the base angles of the triangle having sides x_1, x_2 and x_3, where base is x_1.*

Solution. From the properties of a triangle,

$$\cos A = \frac{b^2 + c^2 - a^2}{2bc} \Rightarrow A = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2bc}\right)$$

where a, b and c are sides and A, the angle opposite to the side a.

```
      FUNCTION ANGLE A (A, B, C)
C     A, B, C ARE SIDES OF A TRIANGLE
C     FUNCTION COMPUTES ANGLE OPPOSITE TO A IN DEGREES
      REAL A, B, C, X
      X = (B * B + C * C - A * A) / (2 * B * C)
      ANGLE A = A COS (X) * 180/3.14
      RETURN
      END
      PROGRAM B SUM
      REAL X1, X2, X3, S1, S2, SUM
      READ *, X1, X2, X3
      S1 = ANGLE A (X1, X2, X3)
      S2 = ANGLE A (X2, X1, X3)
      SUM = S1 + S2
      PRINT *, SUM
      STOP
      END
```

Subroutine Subprogram

A subroutine subprogram, like statement function definition and function subprogram, is also a subprogram which avoids repeated writing a step or a series of steps.

The general form is

```
SUBROUTINE name (x1, x2, ..., xn)
    or
SUBROUTINE name
Program steps
RETURN (at each logical exits from program)
END
```

Name is same as variable name (up to six characters, etc.) except that the name of subroutine has no type significance since the result of calculation in a subroutine subprogram and transferred through the arguments and through its name.

Important points

1. A SUBROUTINE statement must be the first statement of a subroutine subprogram.
2. SUBROUTINE subprograms are placed separately, following the END of the calling program.
3. A SUBROUTINE subprogram may use STATEMENT FUNCTION definition,

FUNCTION subprograms or other SUBROUTINE subprograms but in no case should make reference to itself.

4. A SUBROUTINE subprogram is an independently compiled program and as such may have same variables and statement members as the main program without causing any confusion.
5. A SUBROUTINE subprogram may not transfer any value or may transfer any number of values to the calling program.
6. SUBROUTINE subprogram are called by a CALL statement in the calling program.

CALL name $(x_1, x_2, \ldots, x_n)$
for subroutine defined as having argument list.

CALL name
for subroutine defined as without argument list.

7. The dummy arguments in the defining statement must agree with the actual arguments in the calling statement in number, order and type.

Example 13.39. Write a subprogram to return the sum and product of two integers and call it from a main program. **(AMIE, W '95)**

Solution.

```
      PROGRAM MAIN
      READ *, M, N
      CALL SUPR (M, N, ISUM, IPRO)
      PRINT * , ISUM, IPRO
      STOP
      END
      SUBROUTINE SUPR (I, J, IS, IP)
      IS = I + J
      IP = I*J
      RETURN
      END
```

Example 13.40. Write a subroutine subprogram which transpose a square matrix A of order N.

Solution.

```
      SUBROUTINE TRANSP (A, N)
      DIMENSION A (N, N)
      DO 10 I = I, N
      DO 10 J = I + 1, N
      TEMP = A (I, J)
      A (I, J) = A (J, I)
      A (J, I) = TEMP
   10 CONTINUE
      RETURN
      END
```

Example 13.41. Write a subroutine subprogram to multiply a matrix A of order L × M with a matrix B or order M × N.

Solution. The output will be the product matrix C of order L × N_0, *where* $C = [c_{ij}] = \sum_{k=1}^{M} a_{ik} b_{kj}$.

The required subroutine may be written as follows:

```
      SUBROUTINE MATMUT (A, B, C, L, M, N)
      DIMENSION A (L, M), B (M, N), C (L, N)
      DO 55 I = 1, L
      DO 55 J = 1, N
      SUM = 0.0
```

```
      DO 44 K = 1, M
44    SUM = SUM + A ( I, K) * B (K, J)
55    C (I, J) = SUM
      RETURN
      END
```

A summary of FUNCTION subprogram and SUBROUTINE subprogram is given below:

	FUNCTION Subprogram	*SUBROUTINE Subprogram*
Name	Up to six letters and digits, the first character is a letter	Up to six letters and digits, the first character is a letter
Type	First letter in name or type declaration	Name has no type significance
Defining statement	FUNCTION name (list)	SUBROUTINE name (list)
Number of arguments	At least one argument	or more arguments
Method of referencing	By referring to it by name in a single arithmetic statement in the calling program	By CALL statement in the calling program
Number of results	One result	Any number of results

EQUIVALENCE Statement

Every variable used in a program are allocated memory locations for execution. But all the variables are never needed to be used at the same time. The EQUIVALENCE statement allows the programmer to assign different variables to the same storage location to avoid the wastage of memory. The variables should occur in the same program or in the same subprogram.

The general form of the statement

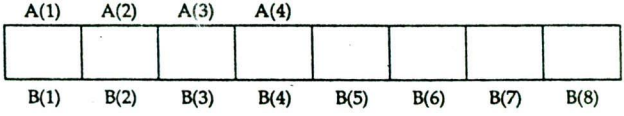
EQUIVALENCE (list 1), (list 2), ...

where each list represents super or subscripted variables separated by commas. The EQUIVALENCE being non-executable type is placed before all executable statements. For example, the statement EQUIVALENCE (A, B, C), (I, J, K) (P, Q) implies that A, B, C share the same memory locations; I, J, K share the next memory location and P, Q occupy the third memory location.

Arrays of different sizes may be equivalenced simply by declaring the first element of the arrays in the statement. For example, the statement

```
DIMENSION A(4), B(8)
EQUIVALENCE (A(1), B (1))
```

will automatically cause all the elements in the two arrays to have common storage in sequence, schematically shown as follows:



COMMON Statement

Common statement has to be used in case variables in main program and variables in

the subprogram can share the same storage locations. In such cases, there should be a COMMON statement one each in MAIN program and subprogram. A COMMON statement may appear in a main program as

COMMON X, Y, Z (10, 10)

If the main program call a subroutine where a common statement appears as

COMMON A, B, C (10, 10)

This means variables X, Y and 10 × 10 or 100 elements of the array Z are allotted storage location in a common area and the same area will also be shared by the variables A, B and 100 elements of the array C of the subroutine.

REVIEW QUESTIONS SET

1. Write a note on use of constants and variables in FORTRAN.
2. Explain why the following constants are invalid integer constants:
   (*i*) 23 + (*ii*) 4, 123 (*iii*) 7.2 (*iv*) 999999999999
3. Explain why the following constants are invalid real constants:
   (*i*) 1, 234 (*ii*) 0 (*iii*) E9 (*iv*) 0.1 E.
   (*v*) .5788E + 90 (vi) −1.23 E + 1.2
4. Write down the rules for variable names in FORTRAN.
5. Write the following algebraic expressions as FORTRAN arithmetic expression:
   (*i*) $\frac{A+BC}{C+D}$ (*ii*) $\frac{4}{3}\pi r^3$ (*iii*) $x^{3/2} - y^{5/3}z$ (*iv*) $|b^2 - 4ac|$
   (*v*) $\frac{\sin x}{x}$ (*vi*) $(C^A)^B$ (*vii*) $\frac{\alpha^2 + \beta^2}{(a+b)c}$ (*viii*) $\log_{10} b + 5 \log_{10} c$
   (*ix*) $(x_1 + x_2)^{n_1}$ (*x*) $\sin(\log_e(x^2 - 25))$ (*xi*) $\sin\theta + \cos\theta$
   (*xii*) $a \sin x + b \sin^2 x + c \cos^3 x$
   (*xiii*) $\sqrt{\frac{2a-5}{c-(d/e)}}$ (*xiv*) $1 + \frac{1}{1 + \frac{1}{1+\sqrt[3]{x^2}}}$
6. Write the mathematical expressions for the following FORTRAN expressions:
   (*i*) *A* + *B*/*A* − *B* (*ii*) *A*/*B* * *C* * *D*/*E*
   (*iii*) SQRT (A) * *B* * SIN (C) (*iv*) (1.0/*A* + 2.0/*B*) * * (C/4.0)
   (*v*) *V* * *T* * SIN (X) − *G* * *T* * *T*/2.0
7. Which of the following are identical to *A* * (*B* + *C*)/*D*? **(AMIE, W '94)**
   (*a*) (*B* + *C*) * *A*/*D* (*b*) (*B* + *C*)/*D* * *A*
   (*c*) (*A* * (*B* + *C*))/*D* (*d*) (*A*/*D*) * (*B* + *C*)
8. What will be the result of the following statements?
   (*i*) *K* = *I*/*J*, if *I* = 3, *J* = 2
   (*ii*) *N* = *I* + 2/*J*, if *I* = 3, *J* = 2
   (*iii*) *N* = *K* * * *L* + *I J*/*L* if *K* = 2, *L* = 3, *IJ* = 15
   (*iv*) *M* = 6 + 6 * (2 + 3 * (9/3 − 1) * * 2)
   (*v*) *A* = *J*/*K* + *I* − *J*/*I* * *K* if *I* = 3, *J* = 8, *K* = 4
   (*vi*) *D* = *B* + *J* * *K* * * *I*/*J*/*C* if *B* = 2.0, *C* = 5.0, *I* = 2, *J* = 3, *K* = 4
   (*vii*) (*a*) *R* /*I* * (*J*/*B*) (*b*) *R* = (*I* + 2)/*A*, *I* = 2.0, *A* = 5.0, *J* = 9.0, *B* = 3.0 **(AMIE, W '9 4)**
9. What will be the final value of *I*, *K* and *X* in the following program:

```
A = 3.0
B = 5.0
K = 16
J = 5
I = A/B
K = K * 2/B
X = K/J
```

10. Find the final value of *K* in the following (show the calculation procedure): **(AMIE, S '96)**

```
      K = 3
      M = 2
      DO 100 J = M, 7, M
      K = J + K + M
      IF (K .GT. 9) GO TO 200
  100 CONTINUE
  200 K = K/3
```

11. (*a*) What is the final value of *K*, (*b*) final value of *A* in the following programs: **(AMIE, S '94)**

(*a*)
```
K = 5
I = 3
IA = 252
M = I * 1000 + IA * 10
K = M/1000 + K
```
where *K*, *I*, *IA*, *M* are integers.

(*b*)
```
A = 2.56
A = (A + 0.05) * 10
I = A
A = I
A = I/10
```

12. Find the value of the following expressions:

(*i*) X + Y .LE. X − Y .OR. NOT. Y + Z .LT. Y − Z
where X = − 3.2, Y = 6.1, Z = 2.3

(*ii*) *A* .EQ. *B* .AND. TRUE.

(*iii*) .NOT. *A* .LT. *B* .AND. FALSE.
where *A* = 10.3, *B* = 2.5

(*iv*) 2 * ABS (*J*) .LE. 3, where *J* = − 4

(*v*) (*I* .LT. *K*) .AND. (*J* .LT. *K*) .AND. (C .GE. *B* + *A*), .AND. (*A* .EQ. *I*)
where *I* = 1, *J* = 2, *K* = 3, *A* = 1.0, *B* = 2.0 and *C* = 3.0

(*vi*) *A* * 2 .GT. *B*/*I* .AND.NOT.FALSE.OR.,TRUE.

13. Differentiate between executable and non-executable statements in FORTRAN.

14. What are various kinds of IF statements available in FORTRAN ? Illustrate with their general syntax.

15. Differentiate between the ASSIGNED GO TO and the COMPUTED GO TO statements in thieir format as well as application.

16. Write a FORTRAN program to compute the perimeter and area of a square, where the length of one side is given.

17. The resonant frequency of an L.C. circuit is given by $f = 1/2\pi \sqrt{LC}$.
Write a program to calculate for various values of *L* and C.

18. Write a program to determine the sum of the series. $S = 1 + x + x^2 + ... + x^n$, *x* being real.

19. Write a program to find the smallest of the three given numbers.

20. Write programs for summing up of the following series for $x = 0.5$, correct up to 5 decimal places:

(*i*) $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$

(*ii*) $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ...$ **(AMIE, W '97)**

21. Given a set of 50 values of *x* and *y*, write a FORTRAN program to compute Σx^2, Σxy, Σxy^2 and Σx^2y^2.

22. Write a program to find LCM of 3 integer numbers.

23. The Fibonacci sequence is given below:
1, 1, 2, 3, 5, 8, 13, ...
It is defined by assuming that the first two terms have the value 1, and each subsequent term is the sum of its preceding two terms. Write a FORTRAN program to generate the first *N* terms of the Fibonacci sequence.

24. Write a FORTRAN program to find out the second largest value from a given set of *N* values, where *N* is a positive integer determined by the user. **(AMIE, S '97)**

25. What is meant by nesting of IF statement? How is ELSEIF used in FORTRAN?

26. What is meant by an array? Describe its advantages.

27. Write a FORTRAN program to multiply two compatible matrices. **(AMIE, W '96)**

28. Write a FORTRAN program to evaluate the sum of the following series up to *n* terms:

$$x^2 - \frac{2^3 x^4}{4} + \frac{2^5 x^6}{6} - \ldots.$$

The values of n and x are to be read as inputs. **(AMIE, W '95)**

29. Write a program to calculate the sum of digits of a given integral number.
30. Write a program to calculate the trace of a given matrix, the elements of which are to be read.
31. Three two-dimensional arrays A, B and C have 10 rows and 10 columns each. Given the array elements of A and B, write a program to compute and print the quantities

$$\text{PROD} = \prod_{i=1}^{10} A_{ii}$$

$$C_{ij} = \sum_{k=1}^{10} A_{ik} B_{kj}, \quad i, j = 1, 2, \ldots, 10.$$

32. (*a*) Compute the scalar product of the two given vectors X and Y given by the formula

$$\text{SUM} = \sum_{i=1}^{20} x_i y_i$$

(*b*) Write a FORTRAN program that reads in two integer vectors X and Y of N elements each and computes the vector Z using only a single loop where Z is given by

$$Z = \frac{\sum_{i=1}^{N} (x_i y_i)}{\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i}$$

(AMIE, W '97)

33. Using the arithmetic IF statements, write statements to compute the function y (x) defined as

$$y = 3x + 4, \text{ if } x \leq 20$$
$$= 2x - 8, \text{ if } x > 20$$

34. Write a FORTRAN to solve the differential equation $\frac{dy}{dx} = -2xy^2$ by fourth-order Runge-Kutta method with initial condition $x_0 = 0$, $y_0 = 1$. The program should print the values of y for $x = 0.1, 0.2, 0.3, 0.4, 0.5$.
35. Write a FORTRAN program to solve the differential equation

$$\frac{dy}{dx} = x^2 + y^2$$

subject to the initial condition $x_0 = 0$, $y_0 = 1$, $h = 0.01$ using modified Euler's method. **(AMIE, S '94)**

36. Find the values of U, V, W and Z of the following program:

```
F (X, Y) = A * X + B * Y
X = 1.0
Y = 2.0
A = 3.0
B = 4.0
U = F (X, Y)
V = F (Y, X)
W = F (A, B)
Z = F (B, A)
```

37. What is the first statement in a FUNCTION subprogram? What is the purpose of the RETURN statement?
38. Write a FUNCTION subprogram ROUND that rounds a real number to the nearest integer value.
39. Write a FUNCTION subprogram to calculate the dot product of two vectors.
40. Write a FUNCTION subprogram to evaluate a second-order determinant, and use it to evaluate the third order determinant.

$$\begin{vmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 0.0 \\ 6.0 & 7.0 & 8.0 \end{vmatrix}$$

41. The length of a vector is defined as

$$\sqrt{A_1{}^2 + A_2{}^2 + ... + A_n{}^2}$$

A data record contains the N components of the vector as follows:

$$N \quad A_1 \quad A_2 \quad ... \; A_n$$

Write a main program to read records of this form (for N ≤ 10) and a function subprogram to compute the length of the corresponding vectors. **(AMIE, S '93)**

42. What is a SUBROUTINE subprogram? Differentiate between FUNCTION and SUBROUTINE subprogram.

43. Given the coordinates of two points P (x_1, y_1, z_1) and Q (x_2, y_2, z_2). Write SUBROUTINE subprogram to calculate the distance between two points.

44. Write a SUBROUTINE subprogram to transfer spherical coordinates into cartesian coordinates according to the formula

$$X = R \sin \theta \cos \phi, \; Y = R \sin \theta \sin \phi, \; Z = R \cos \theta$$

45. Write a FORTRAN subprogram to find the G.C.D. of any two integers.

46. Write a FORTRAN subroutine to arrange N numbers in ascending order. **(AMIE, W '97)**

47. What is the difference between COMMON and EQUIVALENCE statements?

48. Is the following EQUIVALENCE statement valid? Given reasons:

```
COMMON U, V
EQUIVALENCE (T, U) (V, W) (W, T)
```

(AMIE, S ' 94) **(AMIE, W '98)**

49. (a) Write the output values of A in the following program :

```
DO 20 J = 1, 3
IF (J . LE . 2) K = J * * 2
IF (J . GT . 2) K = J
A = J * * 2 + K * * 2
PRINT *, A
10 CONTINUE
```

(b) Find the output displayed by the program segment : **(AMIE, W '98)**

```
I = 0
K = 1
DO  18 J = 1, 17, 3
      K = K + J
      I = I + J + K
      PRINT *, I
18  CONTINUE
    STOP
    END
```

SOLUTIONS TO REVIEW QUESTIONS SET

2. (*i*) Sign must precede the first digit, (*ii*) commas are not allowed, (*iii*) decimal point is not allowed, (*iv*) too large number for most of the computers.

3. (*i*) Commas are not permitted, (*ii*) must contain a decimal point, (*iii*) exponent alone is not permitted, (*iv*) *E* must be followed by an integer, (*v*) too large a number, (*vi*) exponent must be an integer.

5. (*i*) (*A* + *B* * *C*)/(*C* + *D*)
(*ii*) 4.0/3 * 22/7 * *r* * * 3
(*iii*) *x* * * (3.0/2.0) – *Y* ** (5.0/3.0) * Z
(*iv*) ABS (*B* ** 2 – 4.0 * *A* * *C*)
(*v*) SIN (X)/*X*
(*vi*) (C * * *A*) * * *B*
(*vii*) (ALPHA * * 2 + BETA * * 2)/((*A* + *B*) * *C*)
(*viii*) ALOG 10 (*B*) + 5 * ALOG 10 (C)
(*ix*) (X1 + X2) * * *NI*
(*x*) SIN (LOG (*X* * *X* – 36.0)
(*xi*) SIN (THETA) + COS (THETA)
(*xii*) *A* * SIN (X) + *B* * SIN (X) *.* 2 + C * COS (X) * * 3
(*xiii*) SQRT ((2 * *A* – 5)/(C – (*D*/*E*)))
(*xiv*) 1 + 1/(1 + 1 (1 + X * * (2.0/3.0)))

6. (*i*) $A + \frac{B}{A} - B$ (*ii*) $\frac{ACD}{BE}$ (*iii*) $\sqrt{A}B \sin C$
   (*iv*) $\left(\frac{1}{A} + \frac{2}{B}\right)^{C/4}$ (*v*) $H = VT \sin x - \frac{GT^2}{2}$
7. All four are identical to $A * (B + C) / D$
8. (*i*) $K = 1$ (*ii*) $N = 3$ (*iii*) $N = 13$ (*iv*) $M = 90$
   (*v*) -3.0 (*vi*) $D = 5.2$ (*vii*) (*a*) $R = 6.0$ (*b*) $R = 0.8$
9. $I = 0$, $K = 6$, $X = 3.2$
10. $K = 4$
11. (*a*) $K = 10$, (*b*) 2.6
12. (*i*) .TRUE. (*ii*) .FALSE. (*iii*) .FALSE. (*iv*) .FALSE.
    (*v*) .TRUE. (*vi*) .TRUE.
36. $U = 11.0$, $V = 10.0$, $W = 25.0$, $Z = 24.0$
47. Yes, with shared memory U and V all variables will be assigned same value.
49. (*a*) 2, 20, 18 (*b*) 30, 13, 33, 66, 115, 183

13.2 COBOL

COBOL (COmmon Business Oriented Language) was originally developed in 1959 by a group called CODASYL committee to meet the growing demand of a high level programming language suitable for business data processing. The fact that COBOL uses a highly structured format and English-like expressions means that it is, to a large extent, self-explanatory. Since its creation, the COBOL language has undergone extensive refinements in an effort to make it more standardized by the American National Standards Institute (ANSI). All versions of ANSI COBOL are very similar, although there are some variations.

Character Set

The characters permitted in a COBOL program are collectively referred to as the COBOL character set which is as under:

0-9	(10 numerals)
A-Z	(26 English alphabets — only capital letters)
	(Space or blank — sometimes denoted by ∅)
-	(Minus sign or hypen)
+	(Plus sign)
*	(Asterisk or multiply sign)
/	(Slash or division sign)
=	(Equal sign)
$ or £	(Currency sign)
,	(Comma)
;	(Semi colon)
.	(Period or decimal point)
"	(Quotation mark)
(	(Left parenthesis)
)	(Right parenthesis)
>	(Greater than symbol)
<	(Less than symbol)

The above characters are often grouped for convenience of reference. The characters 0-9 are called numeric characters or digits. The characters A-Z are called letters and the remaining characters are called special characters. The space or blank character in certain cases is treated as a letter.

COBOL Words

A sequence of characters can form a word. There are two types of words in COBOL.

(*i*) **Reserved Words :** These words are defined by COBOL and cannot be used for any other purpose than defined by the language. Examples are ACCEPT, ADD, COPY, DELETE, etc.

(*ii*) **User-defined Words :** The user defined words are formed from the letters, digits and hypen. Rules for forming user-defined words are

1. A word is limited to 1 to 30 character.
2. Letters, digits and hypens are only used.
3. First and last character cannot be hypen.
4. No embedded blanks are permitted.
5. Must contain atleast one alphabetic character.
6. Not a COBOL reserved word.

13.2.1. Data Names (or Identifier)

Data names are the most common type of user-defined words. A data name in COBOL is analogous to a variable in algebra. A data name gives reference to the storage area in the memory where the actual value is stored. A single data name or a data name qualified, indexed or subscripted is normally referred to be the general term Identifier. Examples of Data names are

TOTAL - HOURS
1337

13.2.2. Literals

Quite often it is not necessary to assign data names to those quantities that do not undergo changes during the execution of a program. These quantities are used directly as and when required. Such constant quantifiers are known as Literals. There are three types of literals: numeric, non-numeric and figurative constants.

Numeric Literals: A numeric literal can be formed with the help of digits only and is used primarily for arithmetic operations. The following rules must be observed while writing a numeric literal:

1. It is limited to 1 to 18 digits.
2. A + or – sign may be present at the left-most character. The absence of this sign shows that the literal is positive.
3. A decimal point may also be present in a literal but not as the right-most character.
4. No space is allowed in a numeric literal.

Examples of valid numeric literals are

5, –2123, –52.34

Some examples for invalid numeric literals are:

Literals	*Reasons*
2,00	Commas are not permitted
26.	Decimal point at the right-most is not permitted
15.45	Plus or minus, if used, must appear at the left-most character
$ 200	Dollar sign is not permitted.

Non-numeric Literals: It can be any alphanumeric value enclosed in quotation marks and is normally used to output messages or headings. Some systems permit single quotation marks while others permit the use of double quotation marks. The maximum number of characters within quotation marks are usually 120. The following are examples of non-numeric literals :

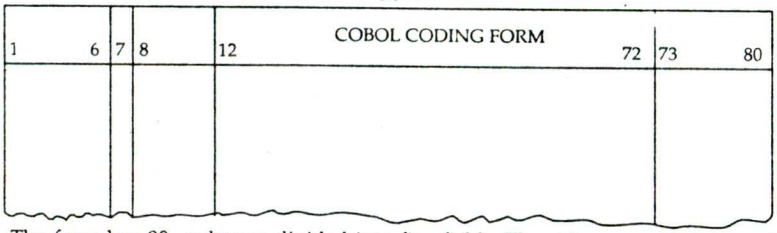
"SEVEN"
"SALES REPORT FOR THE YEAR 1996"

Figurative Constants : COBOL programs use some constants so frequently that these constants have been assigned fixed names. The constants that are referred to by fixed names are called figurative constants and they have special significance to the compiler. When compiler recognises these words it sets up the corresponding values in the object program. The following is the set of figurative constants and their meanings:

Figurative Constants	*Meaning*
ZERO ZEROS ZEROES	Represents the value 0, or one or more of the character 0, depending on the context.
SPACE SPACES	Represents one or more spaces or blanks.
HIGH-VALUE HIGH-VALUES	Represents the highest values in the collating sequence.
LOW-VALUE LOW-VALUES	Represents the lowest value in the collating sequence.
QUOTE QUOTES	Represents one or more of the single character ("). The word QUOTE may not be used to bound a non-numerical literal.
ALL literal	Represents one or more of the string characters comprising the literal. Example : Move ALL " * " TO LONG-FIELD. the receiving field in this MOVE Statement (LONG-FIELD) will be filled with asterisks.

13.2.3. CODING of a COBOL Program

COBOL programs are generally written on coding or program sheets before they are keyed into a computer. The coding sheet appears as shown below.



The form has 80 - columns divided into five fields. The column numbers associated with each field are:

1–6	Sequence number field
7	Continuation field
8–11	Name field
12–72	Text field
73–80	Identification field

Use of Columns

Columns 1–6 are used to ensure that the line are sequenced properly. Sequence numbers or page and line numbers are written for easy identification. The sequence numbers are six digit numbers. The smallest the largest numbers are 000001 and 999999, respectively.

Column 7 is used for continuation, comment or starting a new page. A continuation mark or hyphen is put when one sentence requires more than one line. An asterisk in column 7 is a comment line which is used for providing documentary information on how a program will process data. Column 7 can also be used for causing the printer to skip to the next page when the source listing is being printed by putting a slash (/) in column 7.

Columns 8–11 (Area *A*) are reserved for beginning of division headings, section names, paragraph names, level indicators and certain level numbers. The column 8 is labelled Margin *A*. If an entry is to be coded in Area *A*, it may begin in position 8, 9, 10 or 11. Most often, area *A* entry begins in position 8.

Columns 12–72 (Area *B*) most COBOL entries, particularly those in the PROCEDURE DIVISION are coded here. Column 12 is called Margin *B*. If an entry is to be coded in Area *B*, it may begin anywhere after position 11.

Columns 73–80 are used for program identification. The identification position 73–80 and the page and serial number position 1–6 are optional entries in a COBOL program.

13.2.4. Structure of a COBOL Program

A COBOL program can be defined as a set of declarative and procedural statements arranged in a prescribed order to achieve the target of data processing. The program consists of four divisions. These divisions are subdivided into sections and sections into paragraphs. Paragraphs are broken into statements and statements into words. The hierarchy of the breakdown is as follows:

Division	(A division may contain several sections.)
Section	(A section may contain several paragraphs.)
Paragraph	(A paragraph may contain several statements.)
Statement	(A statement may contain several words.)

A COBOL program consists of up to four divisions in the order presented below, and with the following names:

- IDENTIFICATION DIVISION
- ENVIRONMENT DIVISION
- DATA DIVISION
- PROCEDURE DIVISION

Before discussing the divisions in detail, the purpose of each division is given below:

Name	*Purpose*
IDENTIFICATION DIVISION	Identifies the program to the computer. It also provides some documentation about the program.
ENVIRONMENT DIVISION	Describes the specific computer equipment that will be used by the program.
DATA DIVISION	Describes the input and output formats to be used by the program. It also defines any constants and work areas necessary for the processing of data.
PROCEDURE DIVISION	Contains the instructions necessary for reading input, processing it, and creating output.

13.2.4.1. Identification Division

This division is the first division in every COBOL program. It consists of a number of paragraphs but no section. Except the PROGRAM-ID paragraph, all other paragraphs are optional. However, it is required as a means of identifying the job to the computer. The structure of the identification division is:

```
IDENTIFICATION DIVISION
PROGRAM-ID. Program name.
AUTHOR. Name of author.
INSTALLATION. Name of the computer centre.
DATE-WRITTEN. Date.
DATE-COMPILED. Date.
SECURITY. Security information.
REMARKS. Remarks, if any.
```

The following is an example of the IDENTIFICATION DIVISION:

```
IDENTIFICATION DIVISION
```

```
PROGRAM-ID. PAY LIST.
AUTHOR. A.K. GUPTA.
INSTALLATION. I.E. COMPUTER CENTRE.
DATE-WRITTEN. 24.3.97.
DATE-COMPILED. 24.3.97.
SECURITY. AUTHORIZED PERSONNEL ONLY.
REMARK. SECA RESULT.
```

13.2.4.2. Environment Division

The ENVIRONMENT DIVISION consists of two SECTIONS, the CONFIGURATION SECTION and INPUT-OUPUT SECTION. The CONFIGURATION SECTION provides information about the computer system to be used to compile, and the computer system to be used to execute the program, in most cases they are both in the same machine. The INPUT-OUTPUT SECTION describes the various input and output devices used in the program. The format of the division is

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. Name of computer.
OBJECT-COMPUTER. Name of the computer.
INPUT-OUTPUT SECTION.
FILE-CONTROL
        SELECT input file-name ASSIGN TO input device.
        SELECT output file-name ASSIGN TO output device.
```

The following illustration will describe the method of writing this division.

Illustration : Write an ENVIRONMENT DIVISION for processing a pay-roll file. The input file is PAYROLL fed through the disk and the output file is PRINT-FILE. The device used for PRINT-FILE is printer. The computer used for program compilation and execution is IBM 4043.

Solution.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM 4043.
OBJECT-COMPUTER. IBM 4043.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT PAYROLL ASSIGN TO DISK.
    SELECT PRINT-FILE ASSIGN TO PRINTER.
```

13.2.4.3. Data Division

The DATA DIVISION is that part of a COBOL program that defines and describes fields, records, and files in storage. Following two main sections of the DATA DIVISION in the text are focused:

1. FILE SECTION describes the data items that should be read or written onto some external devices (in the form of files).
2. WORKING-STORAGE SECTION describes those data names that will be used to hold the results of the intermediate computations. Basically, these are used to reserve memory locations so that the value may be stored and manipulated.

The File Section

The file section of the data division considers that all data defined are grouped in files. Files are considered to be divided into records which in turn are subdivided into items. Items are often split into sub-items which can in turn be sub-divided. Items which are not further subdivided are called elementary items. The FILE SECTION must have a file description followed by the record description.

The FD Statement

The file description must begin with FD followed by the file name. FD stands for File Description. This file name must be identical with the file name used in the SELECT clause of the ENVIRONMENT DIVISION. The format for file description is

DATA DIVISION
FILE SECTION
FD file name

LABEL RECORD(S) [IS / ARE] {OMITTED / STANDARD}
RECORD CONTAINS (Integer) CHARACTERS
BLOCK CONTAINS (Integer) RECORDS
DATA RECORD(s) [IS / ARE] (record name(s))

Three entries DATA DIVISION, FILE SECTION and FD are coded in Area *A*. All other entries are coded in Area *B*. For LABEL RECORD, STANDARD is specified if header and trailer labels are used on disk or tape; OMITTED is used for unit record files. The RECORD CONTAINS clause indicates the number of characters in the record. The BLOCK CONTAINS clause indicates whether the file is blocked or unblocked and specifies the blocking factor. DATA RECORD clause indicates the name of each record in the file. After all clauses within FD are coded, a period is to be placed at the end. The order of the clause within the FD is not significant.

Level Structure

To indicate the relationship between a record and various data fields, the concept of level numbers is used in a structured form known as level structure. The level numbers are numeric varying from 01 to 49. A record name is coded on the 01 level and all fields within the record are coded on any level between 02 and 49. Level 01 is coded in area *A* and all others are coded in Area *B* for COBOL 74, although area *A* can be used for COBOL 85. Any field of data with a higher level is subordinate to its lower level. Fields of the same status have the same level numbers. Level numbers 66, 77 and 88 are used for special purposes.

Picture (PIC) Clause

Elementary items or fields not further subdivided must be described with a PICTURE (PIC) clause. The functions of the picture clause are:

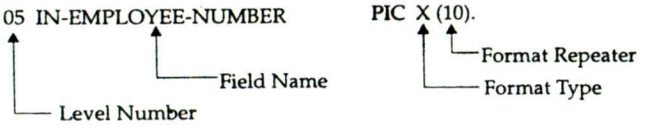
1. To specify the type of data contained with an elementary item.
2. To indicate the size of the field.

The characters used in the PIC clause are

A for Alphabetic
X for Alphanumeric
I for Numeric
V for implied decimal position (used only with numeric fields)

The format repeater specifies the length of the variable being defined. This length may

be stated in two ways. First, by placing a number in parentheses, is shown below and denote a length of ten characters. Second, to specify the variable length by repeating the format description. PIC********** would also denote an alphanumeric string of length ten.



Record Description

It describes the details of a record and indicates the following facts:

(*a*) The data items that appear in a record;

(*b*) The order of data items in which they appear in a record; and

(*c*) The relationship, if any, between these data items and the record.

The following is an example of students record written in Record Description entry:

A	B	
01	STUD-REC.	
02	NAME	
	03 FIRST NAME	PIC A(10).
	03 MIDDLE-NAME	PIC A(10).
	03 SURNAME	PIC A(10).
03	DATE-OF-BIRTH	
	03 DAY	PIC 9(2).
	03 MONTH	PIC 9(2).
	03 YEAR	PIC 9(2).
02	SEX	PIC A(1).

In the above example, record name STUD-REC is defined at 1 level. Group item NAME has no PIC clause. Elementary data items to construct the group are FIRST NAME, MIDDLE NAME and SURNAME. Each of these data items is alphabetic and maximum characters specified are 10. DATE-OF-BIRTH is again a group item and the elementary data items are DAY, MONTH, YEAR. Each of these data items is numeric and maximum digit specified is 2. SEX is the last data item with alphabetic type and size one.

Working-storage Section

This section contains all fields not part of input or output but are used to describe areas wherein intermediate results are stored temporarily and assigned a special level number 77. The working storage section is often used to provide headings for a report, i.e., report headings can be designed and moved to an output file before write operation.

All group items follow the 77 level elementary items and are coded in the margin A. The independent items in the working storage section are generally initialised, i.e., they are given initial contents by a VALUE clause.

VALUE Clause

The VALUE clause specifies the initial of an item at the start of execution of the object program. The literal can be a numeric or a string of characters included within quotes or any figurative constant.

```
WORKING-STORAGE SECTION
77 DATA-N          PIC 9(3) VALUE ZERO.
01 DATE.
```

```
05  DAY        PIC 99.
05  FILLER     PIC X VALUE "/"
05  MONTH      PIC 99
05  FILLER     PIC X VALUE "/"
05  YEAR       PIC 99.
```

Filler

It is a COBOL reserved word used to designate a field that will not be accessed by the program. To understand its application, let a file exists in which the record consists of 8 fields. It is required to process the file, taking 4th and 6th field of the record, the record description entry could be coded as

```
01  EMPLOYEE-REC-IN
    05  FILLER     PIC X(25).
    05  NAME       PIC X(25).
    05  FILLER     PIC X(30).
    05  B-PAY      PIC 9(4).
```

RENAMES Clause

The general form is

66 <Data Name> RENAMES <Data Name 2> [THRU <Date Name 3>]

This helps in re-grouping of elementary data-items in a record so that they can belong to the original as well as to the new group. The following rules are to be observed while using this clause:

1. The RENAMES clause must be used only with the special level number 66 which begins in Margin *A*. There must be at least one blank between the level number and the data-name - 1 and the latter must begin from margin *B*.
2. All RENAMES clause should be placed after the last record description entry.

Example

```
01  SALARY-REC.
    03 FIXED-PART.
       05  BASIC PAY      PIC 9(4) V99.
       05  D–A            PIC 9(4) V99.
    03 ADDITIONAL-PART.
       05  HOUSE-RENT     PIC 9(3) V99.
       05  OTHERS         PIC 9(3) V99.
66 PAY-EXCLUDING-BASIC RENAMES D - A THRU OTHERS.
```

Here PAY-EXCLUDING-BASIC becomes a new group consisting of DA, HOUSE-RENT and OTHERS.

REDEFINES Clause

The general form is

<level number> <data-name-1> REDEFINES <data-name-2>

This clause is used for the purpose of sharing of storage space possibly in the working storage section. Two data-names whose records may not have any meaning resemblance can be used to share the same storage area when they are not used in the program simultaneously. The following rules govern the use of this clause:

1. The level numbers of the data-names must be identical.
2. Except for the 01 level, the size of the data-name-1 must be same as that of data-name-2. The size of data-name-1 must not exceed that of data-name-2.

Example :

```
77 CMS PICTURE 9999
77 METERS REDEFINES CMS PICTURE 99V99
```

The data names CMS and METERS both refer to the same 4 character location in core storage, the difference being that METERS has an assumed decimal point.

OCCURS Clause

An OCCURS clause is defined in the DATA DIVISION to indicate the repeated occurrence of items with the same format within an array. Using traditional methods, coding the input record with 24 independent hourly fields indicating the temperature, we have

```
01 TEMP-REC
   05 ONE-AM      PIC 9(3)
   05 TWO-AM      PIC 9(3)
   05 MIDNIGHT    PIC 9(3)
```

With an OCCURS clause we can write

```
01 TEM-REC
   05 TEMPERATURE OCCURS 24 TIMES PIC 9(3).
```

This defines 24 three position numeric fields. Thus, TEMPERATURE is an array that refers 24 three-byte fields.

When describing an area of storage, more than one level OCCURS may be used. Both double and triple-level OCCURS are permitted with all COBOL compilers.

Suppose we wish to establish in storage an array of hourly temperature reading for Calcutta during a given week. Using a double level occurs in the WORKING-STORAGE SECTION OF DATA DIVISION, we can write

```
01 TEMPERATURE ARRAY
   05 DAY-OF-WEEK OCCURS 7 TIMES
      10 HOUR OCCURS 24 TIMES PIC 9(3).
```

The array consists of 7 (rows) × 24 (columns) temperature readings, that is, there are 24 hourly temperature readings for each of 7 days. To access any of these areas, the lowest level data-name must be used with two subscripts. The first subscript refers to the major (higher) level OCCURS clause and the second (lower) refers to the minor level item. Thus, HOUR (1,6) refers to the temperature for SUNDAY (the first row) at 6 a.m. (the sixth column in the array). The first subscript can vary from 1 to 7 and the second varies from 1 to 24.

When three subscripts are used to access an item, the first refers to the major level, the second to the intermediate level and the third to the minor level.

Consider the following DATA DIVISION entry:

```
01 TEMP-ARRAY
   02 MAJOR CITY OCCURS 12 TIMES
      03 DAY-OF-WEEK OCCURS 7 TIMES
         04 HOUR OCCURS 24 TIMES PIC 9(3).
```

The number of storage locations reserved for the TEMP-ARRAY is 12 × 7 × 24 = 2018. The data-name that may be accessed in the PROCEDURE DIVISION is HOUR. The first subscript varies from 1 to 12, second from 1 to 7 and third from 1 to 24.

13.2.4.4. Procedure Division

This division contains the actual instructions for computations and manipulation of data to be executed by the computer. It is always headed by the words

PROCEDURE DIVISION.

and consists of one or more paragraphs. A paragraph starts with a paragraph name followed after a period by one or more sentences. A COBOL sentence usually consists of one or more operational statements ended by periods. All statements in the PROCEDURE DIVISION are coded in Margin B. Only paragraph names are written in Margin A.

The two most commonly used types of PROCEDURE DIVISION statements are:

(*i*) **Imperative statements** : These statements consist of a verb that indicates action, plus appropriate operands involved in the action. They include:
  (*a*) Arithmetic verb. ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE
  (*b*) Data Transfer verb — MOVE
  (*c*) Control verb — PERFORM, STOP
  (*d*) Input-Output verb — OPEN, READ, WRITE, CLOSE.

(*ii*) **Conditional statements** : These include the statements
  (*a*) If
  (*b*) At End.

Imperative Statement

Arithmetic verb : ADD Statement

Arithmetic operations are performed by the following arithmetic statements. It is important to note that all the literals and data-names appearing in an arithmetic statement must be numeric.

ADD Statement

The format of ADD Statement is

Format 1 ADD {Literal 1 / Data–name 1} {Literal 2 / Data–name 2} To Data–name 3...

Format 2 ADD {Literal 1 / Data–name 1} {Literal 2 / Data–name 2} GIVING Data–name 3.

The result of the ADD operation is always placed in the last field specified after the word TO or GIVING

ADD *A*, *B* TO *C*

The values of *A*, *B* and C are added and the total value will be stored in *C*.

ADD *A*, *B*, *C* GIVING *D*

The values of *A*, *B* and *C* are added and the total value will be stored in *D*.

Format 1 may be used when there is no need to keep the original contents of an operand after the addition.

Format 2 may be used when the original contents of the operands are to be retained.

Arithmetic Verb : SUBTRACT Statement

The subtract operation has the following instruction formats:

Format 1 SUBTRACT {data–name–1 / literal–1} FROM data name – 2

Format 2 SUBTRACT {data–name–1 / literal–1} FROM {data–name –2 / literal–2} GIVING data–name – 3 .... .

In format 1, two or more operands may be subtracts from data-name-2. In addition, operands may be subtracted from data-name-3, data-name-4 and so on. In format 2, any number of data-name can follow the word GIVING, but after the word FROM only one data-name or literal is permitted.

Arithmetic Verb: MULTIPLY Statement.

The MULTIPLY statement has the following instruction formats :

Format 1 MULTIPLY {data–name – 1 / literal–1} BY data-name–2 ...

Format 2 MULTIPLY {data–name – 1 / literal–1} BY {data–name – 2 / literal – 2} GIVING data–name–3....

Arithmetic Verb: DIVIDE Statement.

The divide statement has the following instruction formats:

Format 1 DIVIDE {data–name – 1 / literal – 1} INTO data–name – 2 ....

Format 2 DIVIDE {data–name – 1 / literal – 1} INTO {data–name–2 / literal–2} GIVING data–name–3... .

Format 3 DIVIDE {data–name – 1 / literal – 1} BY {data–name – 2 / literal – 2} GIVING data–name–3 ... .

These arithmetic operations are summarised as follows:

Arithmetic Statement	*Value after Execution of the Statement*			
	A	*B*	*C*	*D*
1. ADD *A* TO *B*	*A*	*A* + *B*		
2. ADD *A B C* TO *D*	*A*	*B*	*C*	*A* + *B* + *C* + *D*
3. ADD *A B C* GIVING *D*	*A*	*B*	*C*	*A* + *B* + *C*
4. ADD *A* TO *B C*	*A*	*A* + *B*	*A* + *C*	
5. SUBTRACT *A* FROM *B*	*A*	*B* – *A*		
6. SUBTRACT *A B* FROM *C*	*A*	*B*	*C* – (*A* + *B*)	
7. SUBTRACT *A B* FROM C GIVING *D*	*A*	*B*	*C*	*C* – (*A* + *B*)
8. MULTIPLY *A* BY *B*	*A*	*A* × *B*		
9. MULTIPLY *A* BY *B* GIVING *C*	*A*	*B*	*A* × *B*	
10. DIVIDE *A* INTO *B*	*A*	*B*/*A*		
11. DIVIDE *A* INTO *A* GIVING C	*A*	*B*	*B*/*A*	
12. DIVIDE *A* BY *B* GIVING *C*	*A*	*B*	*A*/*B*	

Use of REMAINDER Clause in DIVIDE Statement

It is sometime useful to store the remainder of a division operation. The REMAINDER clause with the DIVIDE statement can serve this purpose.

Before DIVISION	*Statement*	*After DIVISION*
A = 32	DIVIDE *A* BY *B* GIVING *C*	*A* = 32, *B* = 5
B = 5	REMAINDER *D*	*C* = 6, *D* = 2

COMPUTE Statement

Most business applications operate on large volumes of input and output and require comparatively few numeric calculations. For this type of processing, four arithmetic verbs just discussed may be adequate. If complex or extensive arithmetic operations are required in a program, however, the use of four arithmetic verbs may prove cumbersome. The

COMPUTE verb provides another method of performing arithmetic. The statement has the form

COMPUTE <data-name> {EQUALS} <arithmetic expression>

COMPUTE $A = B + C - D$

COMPUTE $C = A ** 2 + B ** 2$

Options Available with Arithmetic Verbs

A. ROUNDED Option

ROUNDED is optional with all arithmetic operations and is used to obtain rounded result.

```
ADD  A  TO  B  GIVING   C  ROUNDED
A PIC        99V999      10.857  }
B PIC        99V999      12.142  }  Values
C PIC        99V99       50.67   }
```

After ADD the content of C would be C = 10.857 + 12.142

= 22.999 = 23.00 (rounded)

B. ON SIZE ERROR Option

If after an arithmetic operation, the result exceeds the largest value that can be accommodated in the result field, the error is called a size error. Consider an example,

```
ADD    A    TO B
PIC    A    99
PIC    B    999
```

If $A = 49$ and $B = 990$, after ADD $B = 49 + 990 = 1039$ causes a size error. This is because the result field B is not large enough to hold the result of addition. COBOL has a built in solution, use an ON SIZE ERROR clause with any arithmetic operation as follows:

Arithmetic Statement (ON SIZE ERROR imperative-statement)

ADD A TO B ON SIZE ERROR GO TO ERROR-PARA

The computer performs the arithmetic and ignores the SIZE ERROR clause if there is no size error condition. Otherwise the statement will cause the control to be transferred to ERROR-PARA.

If the ON SIZE ERROR option is employed along with the ROUNDED option, the word ROUNDED always precedes on SIZE ERROR.

DIVIDE A BY B GIVING C ROUNDED ON SIZE ERROR GO TO ERROR-PARA.

If $A = 4.9$ and $B = 0.44$ and A, B and C are described by PIC 9V9, there will be size error and C will be undefined.

DIVIDE A INTO B GIVING C ROUNDED REMAINDER D ON SIZE ERROR TO TO ERROR-PARA

and A, B, C and D are described by PIC 99 and $A = 20$, $B = 50$, $C = 50/20 = 2.5 = 03$ (Rounded) and $D = B - C \times A = 50 - 2 \times 20 = 10$.

Whenever size error occurs in calculating the quotient, the remainder is not stored.

The formats of Arithmetic Verbs with ROUNDED and ON SIZE ERROR are given.

```
ADD      {identifier–1} [{identifier–2}]   GIVING identifier–3
         {literal–1   }  {literal–2   }
         [ROUNDED]  [ON SIZE ERROR imperative statement]

ADD      {identifier–1} [{identifier–2}]   TO identifier–3
         {literal–1   }  {literal–2   }
         [ROUNDED]  [ON SIZE ERROR imperative statement]
```

SUBTRACT {identifier–1 / literal–1} {identifier–2 / literal–2} FROM identifier–3 [ROUNDED]
[ON SIZE ERROR] imperative statement

SUBTRACT {identifier–1 / literal–1} {identifier–2 / literal–2} FROM identifier–*n*
GIVING identifier-*m* [ROUNDED]
[ON SIZE ERROR imperative statement]

MULTIPLY {identifier–1 / literal–1} BY identifier–2 [ROUNDED]
[ON SIZE ERROR imperative statement]

MULTIPLY identifier–1 BY identifier–2 GIVING identifier–3
[ROUNDED] [ON SIZE ERROR imperative statement]

DIVIDE {identifier–1 / literal–1} INTO identifier–2 [ROUNDED]
[ON SIZE ERROR imperative statement]

DIVIDE {identifier–1 / literal–1} INTO {identifier–2 / literal–2} GIVING identifier–3
[ROUNDED] [ON SIZE ERROR imperative statement]

DIVIDE identifier–1 BY identifier–2 GIVING identifier–3
[ROUNDED] [ON SIZE ERROR imperative statement]

Data Transfer Verb: MOVE Statement

The MOVE instruction is given to copy data from a sending field to a receiving field. The sending field is unchanged by the execution of a MOVE statement. The receiving field(s) have their original contents replaced with the contents of the sending field. Thus, the instruction MOVE *A* to *B* designates that *B* should contain a copy of the contents of *A*. In the case of numeric data being moved to an edited numeric field, the data in the receiving field is edited to conform with the editing specified in the PIC clause of the receiving field.

The syntax of MOVE statement is

MOVE {data name – 1 / literal} TO <data – name–2>, <data – name–3> .... .

If lengths of the sending and receiving fields are different, MOVE may result in truncation at the receiving end.

Given

```
01 SAMPLE–MOVE–ITEMS.
   05 A   PIC X(5).
   05 B   PIC X(5).
   05 C   PIC X(3).
   05 D   PIC X(7).
```

Suppose that field *A* contains "MONTH". Then

1. MOVE *A* TO *B*
   copies the contents of data item *A* into data item *B*. Since *A* and *B* have the same PICTURE, they both contain "MONTH" after MOVE. The original contents of item *B* are lost.
2. MOVE *A* TO *C*
   copies the contents of item *A* into item C. Since C is PIC X(3), the value "MONTH" is truncated on the right after the MOVE, *A* contains "MONTH" and C contains "MON".
3. MOVE *A* TO *D*

copies the contents of item *A* into item *D*. Since *D* is PIC X(7), the value "MONTH" is padded on the right with spaces : after the MOVE, *A* contains "MONTH" and *D* contains "␣␣MONTH␣␣".

Control Verb : PERFORM Statement

The PERFORM statement is used to depart from the normal sequence of statement in order to execute a statement or series of statements, a given number of times; or until a specified condition exists. After the statements are executed, control is returned to the statment following the PERFORM.

A simple PERFORM statement is used whenever a series of instructions at a particular paragraph are to be executed from different points in the program. Its format is

PERFORM <Paragraph-Name>

PERFORM CALCULATE-TAX

This will instruct the computer to execute all instructions in the named paragraph and transfer control to the next sequential step following the PERFORM statement. PERFORM statements are permitted within PERFORM statements.

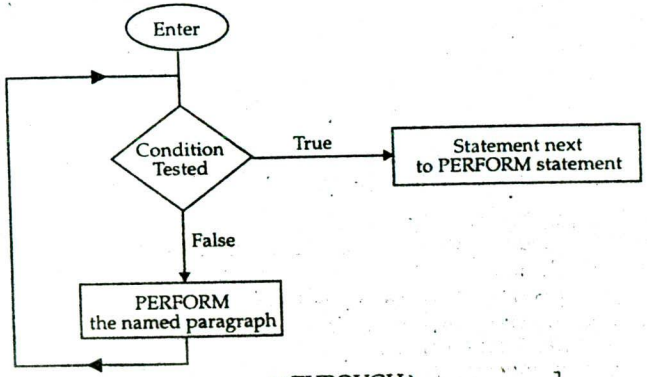
When several paragraphs or routines are to be executed from different points in a program, we can use the statement

PERFORM <paragraph-1> [{THROUGH / THRU} paragraph–2]

The PERFORM executes all statements beginning at paragraph-1 until the end of paragraph-2 is reached. It may be noted that other paragraphs may also be be included in the range. GO TO statements may be included within the named paragraphs using THRU option as long as its control is not transferred outside the range of the indicated paragraphs.

PERFORM with UNTIL Option

The format of the PERFORM statement with the UNTIL option has the following syntax:



PERFORM <paragraph name-1> [{THROUGH / THRU} paragraph–2]

UNTIL <condition>

The condition is tested at the beginning of each repeated execution of the range, and if the condition is found to be true, the PERFORM statement is terminated and the control goes to the next statement following the PERFORM.

```
PERFORM 200 - VALIDITY - CHECK UNTIL X > 7
PERFORM 700 - PRINT-RTN
UNTIL-A = B OR A = C.
```

PERFORM with TIME Option

The PERFORM with TIME option has the following format:

PERFORM <paragraph-Name - 1 > [{ THROUGH / THRU } paragraph-2]

{ integer-1 / Data-name -1 } TIMES

The range is repetitively executed for the number of times specified through the integer or data name and after that the control goes to the next statement following the PERFORM statement. The word preceding TIMES can also be an arithmetic expression with COBOL 85.

Example 13.42. *Write a program segment to add first 100 natural numbers using PERFORM statement with TIME option.*

Solution.

```
MOVE ZERO TO SUM, NUM
PERFORM ADD PARA 100 TIMES
ADD-PARA
        ADD 1 to NUM
        ADD NUM TO SUM
```

Example 13.43. *Write a program segment to sum the odd numbers from 1 through 99.*

Solution.

```
MOVE 1 TO NUM
MOVE 0 TO TOTAL
PERFORM ADD-PARA 50 TIMES
WRITE TOTAL-REC
ADD-PARA
    ADD NUM TO TOTAL
    ADD 2 TO NUM
```

PERFORM with VARYING Option

This option of the PERFORM statement has the following format:

PERFORM <paragraph-name-1> [{ THROUGH / THRU } paragraph-2]

VARYING <identifier-1> FROM < { identifier-2 / integer-1 } >

BY { identifier-3 / integer-2 } UNTIL <condition>

This option is similar to the UNTIL option in the sense that the specified range is executed repetitively until the condition becomes true. The identifier-1 is set to some initial value through identifier-2. The initial value is incremented by which the value of the identifier-1 is increased and is specified through identifier-3.

Example. 13.44. *Write program segment to find the sum of all odd-numbered integers from 1 through 101 using PERFORM VARYING format.*

Solution.

```
MOVE ZERO TO ODD- CTR
PERFORM ADD-PARA VARYING XI FROM 1
BY 2 UNTIL XI 101
ADD-PARA
    ADD XI TO ODD-CTR.
```

Control Verb : STOP Statement

The general from is: STOP {RUN / literal}

The literal may be numeric or non-numeric. When the form STOP RUN is used, the machine is instructed to permanently terminate the execution (run) and the program cannot be executed by pressing the START key.

When the form STOP literal is used, the machine pauses its execution and prints the message stop literal on the console typewriter. The program execution may be re-started by pressing the START key.

Input-Output Verb : READ Statement

After an input file has been opened, it may be read. A READ statement transmits data from the input device, assigned in the ENVIRONMENT DIVISION to the input storage area, defined in the FILE SECTION on the DATA DIVISION.

The general form of the statement is

READ <file-name> RECORD [INTO <data-name>]
AT END <imperative statement>

The INTO option must be used only if the file contains only one record (level 01 data-name). If the file contains more than one type of records, the programmer should test which record has been read. When there are no more records in the file, the program obeys the statement after END.

READ CARD-FILE AT END GO TO EOJ.

This instructs to read a card from the specified input device and the next sequential instruction in the program will be executed unless there are no more cards. AT END clause must be specified for every READ statement.

Input-Output Verb: WRITE Statement

WRITE statement is an output verb and is similar to the input verb READ with the only difference that in WRITE reference is made to a record-name rather than a file name as in READ statement. WRITE statement may be used only for a file which has been opened for OUTPUT, I - O or EXTEND.

The WRITE statement, as used with the printer, has the following format:

WRITE <record-name> [FROM <identifier-1>]

[. {BEFORE / AFTER} ADVANCING < {identifier–2 / integer / mnemonic–name / PAGE} > [LINE / LINES] .]

The FROM option is the counterpart of the INTO option for the READ verb. For example, we can write

MOVE HEADER TO OUTPUT-RECORD
WRITE OUTPUT-RECORD . . .

or we can use the form:

WRITE OUTPUT-RECORD FROM HEADER ...

The ADVANCING clause is used only with files assigned to a printer, and it is used to control the vertical spacing on the printer. We can write with WRITE ... AFTER ADVANCING 2 LINES to achieve double spacing, for instance.

BEFORE ADVANCING states that the information should not be written until the specified number of lines.

Input-Output Verb: OPEN Statement

Before an input file is read or result is written in an output file, we must first OPEN the file. Thus to access a file we use an OPEN statement. An OPEN statement in COBOL has the format.

OPEN INPUT <file-name(s)> OUTPUT <file-name(s)>

An OPEN statement also designates files as input or output. In addition to distinguishing input files from output files and accessing specified devices, an OPEN statement performs certain checking functions. If label records for an input files are indicated as STANDARD, for example, an OPEN statement checks the header label to determine if the correct tape is mounted. If label records for an output tape file are designated as STANDARD, then the OPEN statement will create the header label.

Several OPEN statements may also be written. Thus, the statement,

OPEN INPUT TAPE-1, TAPE-2, OUTPUT TAPE-3 PRINT-FILE.

may also be written in four distinct statements as follows:

```
OPEN    INPUT     TAPE-1.
OPEN    INPUT     TAPE-2.
OPEN    OUTPUT    TAPE-3.
OPEN    OUTPUT    PRINT-FILE.
```

The order in which files are opened is not significant. Since the OPEN statement allows the accession of files, it is generally the first instruction issued to the computer in the PROCEDURE DIVISION.

Input-output Verb : CLOSE Statement

The form is

CLOSE <file-name-1> [NO REWIND] [<file name-2>, . . .]

The tape is rewind if the NO REWIND option is not used.

This statement is used when the processing of a file is completed. This statement is necessary to release the files after processing. All files that have been opened must be closed at the end of a job. Note that a CLOSE statement, unlike an OPEN statement, does not denote which files are input or which files are output.

Conditions and Conditional Statements

A condition is an entity that at any point of time one can have only one of the two values – TRUE or FALSE. In COBOL, a condition can be any of one of the following:

(*i*) Relational condition
(*ii*) Sign condition
(*iii*) Class condition
(*iv*) Condition — name condition.

Relational Condition

A relational condition causes a comparison between two operands and has the following general format:

First	Relational-operator	Second
Operand		Operator

Each operand can be identifier, literal or arithmetic expression. The first operand is called the subject of the condition and the second operand is called the object of the condition.

One of the following can be used as relational operator:

IS [NOT] GREATER THAN

IS	[NOT]	>	
IS	[NOT]	LESS	THAN
IS	[NOT]	<	
IS	[NOT]	EQUAL	TO
IS	[NOT]	=	
IS	UNEQUAL		TO
EQUALS			
EXCEEDS			

Sign Condition

The sign condition is used to test whether the current value of an arithmetic expression is positive, negative or zero. Its general form is

$$\text{Arithmetic expression is [NOT]} \left\{ \begin{matrix} \text{POSITIVE} \\ \text{NEGATIVE} \\ \text{ZERO} \end{matrix} \right\}$$

Class Condition

The class condition determines whether or not the value of an operand is numeric or alphabetic. An operand is numeric if it contains only the digits 0 to 9. An operand is alphabetic if it contains only the letters A to Z and space.

$$\text{Data-name is [NOT]} \left\{ \begin{matrix} \text{NUMERIC} \\ \text{ALPHABETIC} \end{matrix} \right\}$$

Condition—Name Condition

It is a user defined word established in the DATA DIVISION that gives a name to a specific value that an identifier can assume. It is an entity which itself is a condition and thus it can have either a true or false value. However, a condition name cannot be defined independently. It is associated with a data name called conditional variable.

A condition – name is always coded on the 88 label and has only a VALUE clause associated with it. Since a condition – name is not the name of a field, it will not contain a PICTURE clause.

Condition—names refer only to elementary items in the DATA DIVISION. The identifier to which the condition-name refers must contain a PICTURE clause.

The format is

$$\text{88 <condition-name>} \left\{ \begin{matrix} \text{VALUE IS} \\ \text{VALUES ARE} \end{matrix} \right\} \text{<literal–1> ....}$$

The following illustrates the use of condition names:

77	MARITAL	STATUS	PIC 9.
88	SINGLE	VALUE IS	ZERO
88	MARRIED	VALUE IS 1	
88	WIDOWED	VALUE IS 2	
88	DIVORCED	VALUE IS 3.	
88	ONCE-MARRIED	ARE 1, 2, 3.	

There are five condition names and all of them are associated with the condition variable MARITAL-STATUS. If at some state the MARRIED STATUS get a value of 2 then the condition names, WIDOW, ONCE-MARRIED will become true and others will become false.

Basic Conditional Statement

A conditional statement is one that performs a certain operation depending upon

the occurrence of some condition. The basic instruction format for IF statement is as follows:

```
IF      <condition-1>
        Imperative statement-1      . . . .
ELSE    i
        Imperative statement-2      . . . .
```

For example,

```
IF      A > B
        ADD   A TO C
ELSE
        MULTIPLY B BY C
```

Explanation. If *A* is greater than *B*, ADD operation is performed. The second part beginning with ELSE clause is ignored. If *A* is not greater than *B*, then ADD operation is not executed. Only the ELSE portion of the statement, the MULTIPLY operation is executed.

IF Statement without ELSE Clause

If some operation is required only if a condition exists and nothing different need to be done. If the condition does not exists, the entire ELSE clause may be omitted.

```
IF AMOUNT – IN IS EQUAL TO ZEROS
MOVE 'NO TRANSACTIONS' TO PRINT-AREA.
```

IF Statement using NEXT SENTENCE Clause

```
IF <Condition>
    NEXT SENTENCE
ELSE
    Imperative statement ....
```

In this case if the condition is true nothing is done and control passes to the statement that follow the IF statement. IF it is false the statement after ELSE is executed. Thus NEXT SENTENCE clause does not give any extra power to the IF statement. Example,

```
IF ROLL NO. IS NOT EQUAL ROLL-NO-OUT.
        NEXT SENTENCE
        ELSE ADD 1 TO STD - COUNT
        MOVE MARKS - IN TO MARKS - OUT.
```

Nested Conditional (Nested IF)

The nested IF sentence is examined in a left to right manner to encounter each ELSE in the order of its appearance. As soon as an (ELSE) IF is encountered, it must be paired with the immediately preceding IF that has not been paired with an ELSE.

```
IF DEPT = 1
   (1)
IF DAYS – WORKED IS 175 MOVE 1 TO BONUS – CODE
   (2)
ELSE NEXT SENTENCE
   (2)
ELSE IF DAYS – WORKED  205
   (1)      (3)
        MOVE 2 TO BONUS - CODE
        ELSE NEXT SENTENCE
        (3)
```

IF-ELSE associations have been indicated by marking each pair with identical numbers within circles. **(AMIE, W '97)**

13.2.2.5. Distinction between COBOL and FORTRAN Languages

COBOL	*FORTRAN*
1. Business-oriented language.	1. Scientific computation oriented language.
2. Extensive non-numerical processing is feasible.	2. Mainly numerical processing is feasible.
3. Run data structure.	3. Few data types.
4. Enforces documentation.	4. Documentation at the user's hands.

REVIEW QUESTIONS SET

1. Name the four COBOL divisions.
2. Name the sections comprising the Data Division.
3. What is the level number of record descriptions?
4. What are the special level numbers and their uses?
5. List the figurative constants.
6. In which DIVISION of a COBOL program do the following appear:
   (*a*) PICTURE (*b*) MOVE (*c*) FILESECTION
   (*d*) PROGRAM-10 (*e*) OBJECT-COMPUTER
7. Write the single statement to calculate
   $A + (B + C)/D \times E$
8. If NUM-1 contained 126 V 36 and NUM-2 contained OV63, what could be the result of the statements.
   (*a*) ADD NUM-1 TO NUM-2 GIVING SALES.
   (*b*) ADD NUM-1 TO NUM-2 GIVING SALES ROUNDED.
9. Write statements in COBOL to carry out the following operations:
   (*a*) Divide TOTAL by 30 determine AVERAGE
   (*b*) To name the program as RESULT
   (*c*) Store the value in FIELD-A
   (*d*) Add the value of SAL, HRA and DA and place the sum in GROSS.
10. Write down the classes of data and the PICTURE string characters to represent their occurrence.
11. Write a single statement to execute 500–PARA-5 if *A* is between 3 and 13, exclusive of the end points.
12. Write a single statement to execute 300-PARA-3 if the following conditions are met, otherwise perform 200-PARA-2 (*a*) $A = B$, (*b*) $C = D$, (*c*) $E = F$.
13. What is the purpose of the PICTURE clause?
14. What is the use of FILLER?
15. Given

```
01   SAMPLE-ITEM
     05   A    PIC 9(4) V99
     05   B    REDEFINES A
          10   C    PIC 9(2) V9
          10   D    PIC 9(3).
```

    What are the contents of C and of D after the statement MOVE 74.86 TO A has been executed?
16. What is the function of the OPEN and CLOSE statements?
17. Define conditional statement and imperative statement.
18. What is the purpose of the PERFORM statement?
19. Are the following statements

```
COMPUTER RATE = 125
MOVE 125 TO RATE
```

    equivalent?

ANSWERS TO REVIEW QUESTIONS SET

1. IDENTIFICATION Division, ENVIRONMENT Division, DATA Division, PROCEDURE Division
2. File Section, Working-storage section, Linkage section.
3. 01
4. 77 non-contiguous, 88-conditions
6. (*a*) Data Division (*b*) Procedure Division
   (*c*) Environment Division (*d*) Identification Division
   (*e*) Enviroment Division
7. COMPUTE $A = (B + C)/D \times E$
8. (*a*) 126 (*b*) 127
9. (*a*) DIVIDE TOTAL by 30 GIVING AVERAGE
   (*b*) PROGRAM-ID. RESULT
   (*c*) MOVE 50 TO FIELD-*A*
   (*d*) ADD SAL, HRA, DA GIVING, GROSS
10. Numeric - 9
    Alphabetic - *A*
    Alphanumeric - *X*
11. IF $A > 3$ AND $A < 13$
    PERFORM 500-PARA-5
12. IF $A = B$ and $C = D$ AND $E = F$
    PERFORM 200-PARA-3
    ELSE
    PERFORM 200-PARA-3
13. A contains 0012.34 because of the REDEFINES, however, C and *D* refer to exactly the same computer memory location as does *A*. C contains left half of *A* which is 00.1 and *D* contains 234. *B* is a group item.
19. Yes.

13.3. ALGOL

ALGOL (ALGOrithmic Language) is one of the oldest procedure - oriented programming languages. It was first formulated in 1958 and became known as ALGOL 58. It was later modified in 1960 known as ALGOL 60 which was a substantial improvement over ALGOL 58. In 1968, the version ALGOL 68 was developed with remedies of shortcomings in ALGOL 60. Though the language was designed to write programs which are mostly based on algorithms that are required in science and engineering but it could not compete with more structured and modular languages like Pascal, FORTRAN, etc.

13.3.1. Character Set

The character set of ALGOL consists of

(*i*) 26 upper case letters (A to Z)
(*ii*) 26 lower case letters (a to z)
(*iii*) 10 digits (0 to 9)
(*iv*) Arithmetic operators : $+ - * / + \uparrow$
(*v*) Relational operators : $<, \leq, >, \geq, =, \neq$
(*vi*) Boolean operators : $\equiv \vee \wedge \neg \supset$
(*vii*) Assignment symbols : =
(*viii*) Delimiters : ; ,
(*ix*) Brackets [ ] ( )

Characters are grouped together to form words using standard rules and there are reserved words which have well defined meaning and significance. The following is the list of common keywords in ALGOL, GO TO, IF THEN ELSE, FOR UNTIL, WHILE DO, BEGIN, END, TRUE, FALSE, REAL, INTEGER, ARRAY, STRING, BOOLEAN, SWITCH, VALUE, LABEL, OWN, PROCEDURE, COMMENT.

13.3.2. Constants and Variables

ALGOL can process integer and real numbers, Boolean data, arrays and strings.

Type	Examples
Integer	9, –2087
Real	.0015, 2.0 e + 3, –3.0 e^{-4}
Boolean	TRUE FALSE
String	"$ 18.75 EACH"

ALGOL variables are quantities referred to by name. Variable identifiers are the symbols used for variables; they are composed of strings of small or capital letters and digits, the first of which is a letter. There is no limit on the size of the symbol. Examples are:

LIST, Start, A567, *B*

Subscripted variables are indicated by placing the subscripts in square brackets. The elements of a two-dimensional array might be written as follows:

a [1, 1], *a*[1, 2], ..., *a*[3, 3]

13.3.3. Data Types

ALGOL permits the following three data-type specifications:

REAL I, J MAX, LIST;

This declaration specifies that values associated with the identifiers must be chosen from a set of real numbers.

INTEGER C1, C2, C3

This declaration specifies that values associated with the identifiers must be chosen from a set of integer numbers.

BOOLEAN P, Q, R

This declaration specifies that *P*, *Q* and *R* are two-valued variables which have one of the two values — TRUE or FALSE.

A variable may be assigned a value at the same time as it is declared by writing

integer x : = 7;
real b : = 625.3;

Undeclared variables are assumed to be real.

13.3.4. Assignment Statement

The general form of assignment statement is

variable : = expression;

where the left hand side variable and the right hand side expressions are compatible in their types. The statement

x: = y + z;

assigns value of $y + z$ to the identifier x.

This statement

x: = y: = s;

assigns the value s to the variable x and y. This is known as multiple assignment statement.

Label

A statement may be assigned a label for reference purposes; a label is similar to a symbolic address and may consist of an integer or a symbol. If a label *L* is assigned to a statement *S*, it is separated from the statement by a colon:

$L : S$

Switch Declaration

In ALGOL, there are facilities for defining arrays of labels and selecting an element of a label-valued array by means of array subscripts. A label-valued array is declared by means of a switch declaration

Switch S: = $L1$, $L2$, $L3$, $L4$;

This declaration declares S to be a label-valued array with $S(1) = L1$, $S(2) = L2$ and so on.

13.3.5. Array Declaration

An array declaration indicates how many numbers are present in an array in each dimension. It is written in the form

array subs v [l_1; u_1, l_2 : u_2, ...]

where subs v is a subscripted variable, and l_i and u_i are the lower and upper bounds of the subscripts of the subs v for several dimensions. For example, the 3×3 array would be written as

array a [1 : 3, 1 : 3]

Several variables may be listed within one array declaration

array x [−3 : 5, 4 : 9], y [4 : 5, 6 : 15]

The array declaration indicates the x-array is 9×6 in size and that the b-array is 2×10 or 20 in size. The word array means real array for other types, the type must be specified, as in integer array.

Unconditional Jump

To indicate an unconditional transfer of control, the following statement is used:

go to L

where L is a label.

Conditional Jump

To indicate transfer of control if statement is used

if R then S

where R is a relational expression. The statement S can be of any type except another if statement. Frequently S is a go to statement

if R then go to P

Here if R is true, control will go to P; otherwise, control will go in sequence after the go to statement.

Example 13.45. Write a program segment that sums 100 numbers.

Solution.

```
      Sum : = 0;
        i : = 1;
M : Sum : = sum + a [i];
        i : = i + 1;
if i ≤ 100 then go to M
```

Another version of the **if** statement is used if two specific, different actions are to be taken for the cases where the expression is true and false. It is of the form

if R **then** $S1$ else $S2$

If R is true, statement $S1$ is executed and statement $S2$ is skipped; if R is false, $S1$ is skipped and $S2$ is executed. In either case, control then passes to the next statement in sequence after $S2$, unless $S1$ or $S2$ causes a transfer of control. This statement $S2$ can be of any type, including another **if** statement.

13.3.6 Loop

Loops are most easily written using a **for** statement. There are three basic forms to the **for** statement.

1. One form, which might be called the "step-until" form, permits stepping of the control variable over a finite range in equal increments. The statement is written as

   for $v := a$ **step** c **until** b **do** S

   where a, b, and c are arithmetic expressions. The statement S is executed repeatedly with the variable v equal successively to a, $a + c$, $a + 2c$, $a + 3c$, until v equals the largest value not exceeding b. Most commonly, a, b, and c are positive or negative integers.

   The program, for example 13.3.1, can be written using step-until form as

   ```
   sum : = 0;
   for i : = 1 step 1 until 100 do sum: = sum + a [i]
   ```

2. A second form uses a list of arithmetic expressions, each entry of which in turn is to be assigned to the control variable for the execution of S. The statement is written as

   for $v := E1, E2, \ldots$ **do** S

   where $E1, E2, \ldots$ are arithmetic expressions.

3. The third form permits repeated execution of a statement as long as a conditional relation is true. The form is

   for $v := E$ **while** B **do** S

   where E is an arithmetic expression or a list of arithmetic expressions and B is a Boolean relation. Sometimes E is the initial value of v, and subsequent values are computed by the statement S.

13.3.7. Block Structure

Compound statements : Several statements may be combined to form a compound statement, which delimited by begin and end.

For example,

```
begin
  x : = 3
  y : = 7
end
```

This compound statement assigns values to both the identifiers x and y.

Blocks : The begin-end statement parenethesis are also used for the purpose of determining the range of statements over which the identifiers retain their meaning. The basic module for specifying this range is called a *block*.

Example 13.46. *Evaluate the formula*

$$f = Ax^2 + Bx + C$$

For each of the values of $x = 1, 2, 3, 4$, *while* $A = 7$, $B = 8$ *and* $C = 9$.

Solution.

```
for x = 1, 2, 3, 4 do
begin
    A : = 7; B: = 8; C = 9;
    f: = A * x ↑ 2 + B * x + c;
end
```

13.3.8. Subroutines

The term procedure in ALGOL is used for subroutine.

procedure <name of subroutine> (P_1, P_2, ...)

The subroutine is later called as follows:

<name of subroutine> (V_1, V_2, ...)

with particular values of the parameters given as V_1, V_2, ... We can write a procedure that returns the area of triangle through a parameter as shown below:

```
procedure area of triangle (a, b, c, area);
    real a, b, c;
    begin
        real S: = (A + B + C)/2;
        area : = sqrt (S * (S – a) * (S – b) * (S – c));
    end;
```

To call on the subroutine to compute the functions for particular values of the parameters, calling the result A, we write

area of triangle (4, 3, 5, A)

for which the computed answer would be

$A = 6$

Functions are considered to be a special case of procedures, where the result is a single value. The function is used in a statement by calling it in the manner of a procedure, but the name of the function is considered to be a variable. The word procedure is preceded by a type word (real, integer or Boolean).

Example 13.47. *Given a set of n numbers, count the number of negative numbers present.*

Solution.

```
j: = 0;
for i : = 1 step 1 until n do
    if a[i] < 0 then j : = j + 1 else j : = j
```

Example 13.48. *Given two matrices, A with m rows and n columns and B with n rows and p columns, determine the matrix product*

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk} \qquad 1 \le i \le m,\ 1 \le k \le p$$

Where a_{ij} and b_{jk} are the elements of A and B, respectively.

Solution.

```
for i : = 1 step 1 until m do
begin for k : = 1 step 1 until p do
  begin c [i, k] : = 0; for j : = 1 step 1 until n do
    c [i, k] : = c [i, k] + a [i, j] × b [j, k];
  end;
end;
```

REVIEW QUESTIONS SET

1. Write the list of the character set in ALGOL.
2. Give the list of relational and Boolean operators in ALGOL.
3. What are various loop constructs in ALGOL? Give examples of each.
4. Write an ALGOL program to find the sum and average of n numbers.
5. Write an ALGOL program to find the transpose of a square matrix.
6. Write a function which computes the value of f, given a, b and c. The function is real.
   $f = a^2 + b^2 + c^2$
7. Write a program to find the sum of the tenth power of eighty numbers.
8. Write a progoram to find the sum of the squares of first 100 natural numbers.

13.4. PL/1 (PROGRAMMING LANGUAGE/1)

The programming language PL/1 was developed during 1963-66 by a committee of users of large-scale IBM computers, specially IBM 360 computer system. This is a language used in science and engineering and also in business. It is called a multipurpose language. The design of PL/1 draws heavily on the earlier languages like FORTRAN, ALGOL and COBOL and also some new features were added.

13.4.1. Character Set

Character set of PL/1 consists of the following characters:

Extended Alphabet (29 characters) :
A, B, ..., Z, $, @, #
Decimal Digits (10 characters) : 0, 1, ..., 9.
Special Characters (21 characters) :

Plus sign	+
Minus sign	–
Multiplication sign	*
Slash or Divide sign	/
Equal or Assignment sign	=
NOT symbol	¬
AND symbol	&
OR symbol	|
Greater than symbol	>
Less than symbol	<
Break character	-
Question mark	?
Left parenthesis	(
Right parenthesis	)
Single quotation mark	'
Per cent symbol	%
Comma	,
Semicolon	;
Point or Period	.
Blank	
Colon	:

Special characters may be combined to generate other symbols such as ¬, = (used for 'not equal to'), ** (to denote exponentiation), etc.

13.4.2. Numeric Constant

A number in PL/1 is characterised by following four attributes:

(*i*) Mode specifies whether it is real or complex,
(*ii*) Scale specifies whether it is in fixed point or floating point representation,
(*iii*) Base specifies whether it is a binary or a decimal number,
(*iv*) Precision specifies the number of digits both total and after the decimal point (or binary point).

The attributes are used with following syntax:

mode scale base precision

The following are the examples of real data defined with attributes:

REAL FIXED DECIMAL (6, 2) e.g. 3425.67, – 234.21
REAL FIXED BINARY (8, 3) 1001.111*B*, – 110.001*B*

The precision for FLOAT attribute contains only total number of digits. For example,

REAL FLOAT DECIMAL (5) e.g. 44.102 E 5

Non-numeric Constant

A character string constant is represented as one or more characters enclosed within quotes; for example,

'COUNT' '$ 3.45' 'AbBbIS'

A bit string is represented by string of symbols 0 and 1 only within quotes followed by a letter *B* after the closing quote. For example,

'00110' *B* '11'*B*

13.4.3. Identifiers

The PL/1 distinguished two types of identifiers. An external identifier consists of from one to seven/six alphanumeric characters, the first of which must be alphabetic. External identifiers are used both by the operating system and within the PL/1 program.

An internal identifier is a name which is used and known within a PL/1 program. The internal identifiers should abide by the following rules:

(*a*) Maximum 31 characters are allowed,

(*b*) First character should be either any one from A to Z or $, #, @, followed by a letter or a digit,

(*c*) An underscore — can be used in-between to increase readability.

Some valid identifiers are

$ NAME-OF-STUDENT
# NUMBER-CODE
Z 3W2

Variable names in PL/1 form part of internal identifiers.

13.4.4. Variable Declarations

A variable in PL/1 can explicitly be declared optionally with one or more attributes using DECLARE statement. If there is no DECLARE statement, then the type of variable is as per initial-letter naming convention, i.e., integer type if it begins with one of the letters I, J, K, L, M, N, otherwise it is treated as of real type. The general form of DECLARE statement is

DECLARE name attributes [, name attributes] .... [, name attributes];

where name is a variable identifier and attributes describe its characteristics stating the mode and precision. For example,

DECLARE MAX CHARACTER (100);

declares the variable MAX to be character in form with a maximum length of one hundred characters.

We can have

DECLARE TAX FIXED DECIMAL (7, 2);
DECLARE B FLOAT DECIMAL (10);

The declarative statement can be abbreviated as

DCL	for DECLARE
CHAR	for CHARACTER
DEC	for DECIMAL
BIN	for BINARY
CPLX	for COMPLEX

When several variables are required to be declared of same type, the attributes are factored and the general form of the statement is

DCL (name-1, name-2, ...) attributes;

More than one variables of same type attributes can appear as follows:

```
DCL   (A, B, C)   FIXED DEC (8, 2);
```

The form of arithmetic data items can also be declared by the use of PICTURE attributes, which gives a symbolic representation of the contents of the variable. For example,

```
DECLARE SUM   PICTURE  '99999';
DECLARE TOTAL PICTURE  '999V99';
```

where 9 specifying a decimal digit for the data.
V specifying the position of assumed decimal point.

Assignment Statement

Assignment is used to signify that the value of an expression is to be copied to one or more receiving fields. For example,

```
X1 = ( - B + SQRT (B ** 2 - 4 * A * C))/(2 * A);
C1, C2, C3 = - 6.67;
TITLE = 'SAMPLE HEADING';
```

13.4.5. Control Statements

GO TO and IF Statement

GO TO has a format of

```
GO TO < label >;
```

where <label> is a label of the statement to which control is to be transferred.

IF statement has the form

```
IF <condition> THEN <statement>; [ELSE <statement 2 >;]
```

ELSE enclosed in a bracket is optional. Following are examples of IF statements.

IF $A - B > 0$ THEN $C = D$; ELSE $C = -D$;

IF $T_{\neg} = X$ THEN $Y = 1$;

Do Statement

It has two forms :

(*i*) DO < bar > = < e_1 > TO < e_2 > BY < e_3 >;
< Statements in the loop >;
END;
where < bar > is the control variable, < e_1 > is the initial value, < e_2 > is the final value and < e_3 > is the increment.

(*ii*) DO WHILE <logical Exp.>;
<statements in the loop>;
END;
An example of form (*ii*) is

```
DO WHILE (P > Q);
    X = P + Q;
    END;
```

13.4.6. Arrays

Arrays may be defined in the same manner as the FORTRAN's method of giving dimensions with type declaration such as

```
DCL A (6, 5, 4) FIXED DEL (9, 2);
```

The statement declares a three-dimensional ($6 \times 5 \times 4$) array, each element having 9 decimal digits of which two are after the decimal point.

13.4.7. Input-Output Statement

The input and output in PL/1 are carried out through files, similar to COBOL. There are two modes of data input/output. The first one is known as stream input/output and is known as Record input/output.

Stream I/O, a mode of data transmission, treats both input and output as continuous streams of data without distinguishing physical breaks like end of block, end of record, etc. A record I/O mode considers the data to be prearranged in certain forms and groupings, with separation from records corresponding to actual physical record like end of magnetic tape.

The stream input/output can be grouped into four major groups:

(*a*) DATA directed

(*b*) LIST directed

(*c*) EDIT directed

(*d*) STRING directed.

(*a*) **DATA directed transmission:** The data directed stream transmission is used when proper labelling of output data is required. This transmission can be effected by two commands, namely, GET DATA and PUT DATA. The general form of GET DATA statement is

GET DATA (v_1, v_2, ..., v_n);

each v denotes a variable for which a value is to be read. PL/1 converts each value to its declared base, scale, mode and precision. If an undeclared numerical variable is read in, it will be assigned default attributes.

The basic form of data directed output is given by

PUT DATA (v_1, v_2, ..., v_n)

(*b*) **LIST directed transmission :** In the LIST directed mode of transmission, the input file contains values of data, which are assigned to identifiers in the list, the reverse being the case for output. The general format of GET LIST and PUT LIST statements are

GET LIST (v_1, v_2, ..., v_n);

and PUT LIST (v_1, v_2, ..., v_n);

(*c*) **Edit directed transmission :** Edit directed transmission has been designed so that the programmer can specify the form and organisation of this data. In output, the user can format the output in his desired way. Edit directed transmissions can be carried out by GET EDIT and PUT EDIT statements. The general form of the statements are

GET EDIT (v_1, v_2, ..., v_n) (d_1, d_2, ..., d_n);

where d_1, d_2, ..., d_n is a corresponding list of descriptions of the variables. For example,

GET EDIT (P, Q) (F (3), F (6.3));

and PUT EDIT (v_1, v_2, ..., v_n) (d_1, d_2, ..., d_n);

(*d*) **String directed transmission :** This method uses a character string stored in the memory as input or output. The basic form of the statement for reading the string is

GET STRING (string name) EDIT (v_1, v_2, ..., v_n) (d_1, d_2, ..., d_n);

where string name is the name of the area in store from which the information is to be transmitted.

The data can be output using the statement whose general form is

PUT STRING (string name) EDIT (v_1, v_2, ..., v_n) (d_1, d_2, ..., d_n);

13.4.8. Procedures

In PL/1, a procedure is a set of statements. Simple program consists of a single procedure which is given a title followed by a colon and has PROCEDURE OPTIONS (MAIN), written after it. For example, the following simple program would print squares and cubes of numbers betweeen 1 and 100.

```
POWERS : PROCEDURE OPTIONS (MAIN);
DO I = 1, 100;
J = I * I, K = J * I;
PUT LIST (I, J, K); END;
STOP, END POWERS;
```

13.4.9. Subroutines

The PROCEDURE statement of a subroutine usually contains parameters in which are placed values taken from the calling procedure. Consider the following subroutine which converts millimetres to centimetres and metres.

```
METRIC : PROCEDURE (PMM, LM, SCM, DMN);
    DECLARE (PMM, LM, SCM, DMM) FIXED;
    LM = (PMM/1000);
    SCM = PMM – (1000 * LM);
    IF SUM = 0 THEN GO TO OUT;
    SCM = (SCM/100);
OUT : DMM = PMM – (1000 * LM + 100 * SCM);
    RETURN; END METRIC;
```

Subroutines are invoked by the CALL statement (in the calling program). The CALL statement for the METRIC subroutine procedure could be

```
CALL METRIC (A, B, C, D);
```

The arguments should have one - one correspondence with the parameters and they should have identical attributes.

REVIEW QUESTIONS SET

1. Give the list of PL/1 character set.
2. Discuss about the data attributes of PL/1 data types.
3. Discuss GET and PUT statements with examples.
4. Explain the 'construct' of an IF statement in PL/1.
5. Explain the different mode of transmissions with respect to I/O statements.

14

COMPUTER GRAPHICS

14.0. INTRODUCTION

Computer graphics remains one of the most exciting and rapidly growing computer fields. The study of computer graphics calls for an understanding of computer and display technologies and a solid grounding in two - and three - dimensional analytic geometry. This is to be combined with a knowledge of matrix operations, data structures and algorithms. It is the aim of this unit to provide the necessary technical and mathematical background required for the study of computer graphics.

14.1. WHAT IS GRAPHICS?

Graphics is the study and use of graphs. These graphs may be used to represent numerical values in pictorial form. A picture is a powerful means of communication. Drawing plays an important role in communicating the engineering designs. Some limitations of manual graphical representation are:

(*a*) Modifications are difficult;
(*b*) Any improvement in the quality of picture drawn is difficult; and
(*c*) The quality depends on the experience and the skill of the person preparing them.

14.2. COMPUTER GRAPHICS

Computer graphics refers to the drawings generated by computer. It is the use of computer to define, store, manipulate and present pictorial output. Some of the commonly used areas of application of computer graphics are:

(*a*) Computer Aided Design
(*b*) Computer Aided Drafting
(*c*) Computer Aided Manufacturing
(*d*) Computer Animation
(*e*) Modelling
(*f*) Video Games
(*g*) Image Processing
(*h*) Real Time Simulation and Training.

Computer graphics can be divided into two parts;

(*a*) Non-interactive or Passive Computer Graphics; and
(*b*) Interactive Computer Graphics.

Non-interactive Computer Graphics: In this, the observer has no direct control over the picture. Based on the instructions programmed, the computer prepares and presents stored information to an observer in the form of picture. Such pictures may be generated with the help of computer-controlled plotter on paper or film, the titles shown on TV, etc.

Interactive Computer Graphics: In this, the observer has control over the images. He can change and modify the images by interacting with the system, using a graphic input device. For example:

(*a*) Simulators can be simulated and a CRT displays realistic scenes which change in response to action of learner,

(*b*) Real-time animations using wireframe displays on a video monitor are useful for testing performance of a vehicle or system. The calculations of each segment of the animation can be performed quickly to produce a smooth real-time motion on the screen. Also, wireframe displays allow the designer to see into the interior of the vehicle and to watch the behaviour of inner components during motion.

14.3. INTERACTIVE COMPUTER SYSTEM

A typical interactive computer system consists of four major subsystems: computer display, processing unit, display device, and input device. Figure 14.1 shows the block diagram of a computer graphic system.

The display processing unit (display processor) is a special purpose processor acting as an interface between the host and the terminal. It receives the request from the terminal. Depending upon the type of request it executes a sequence of display instructions to draw pictures on the display device. It activates appropriate signal in the display generator.

The display file memory is a form of main memory which contains display program statements. Each word of the display list describes a basic picture elements such as a point or a line which can be accepted by the display device.

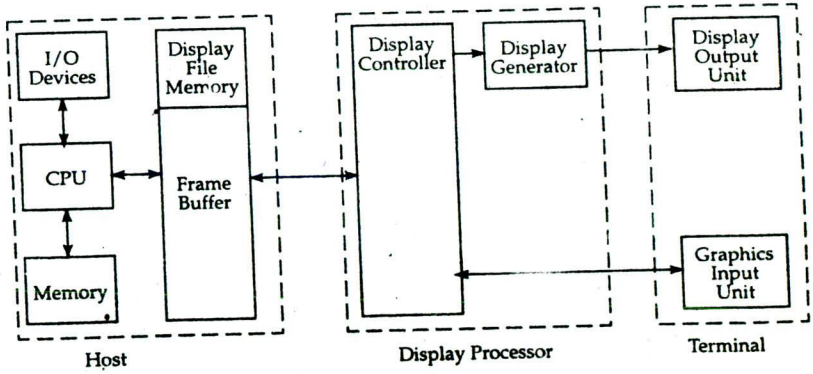


Fig. 14.1: A computer graphics system

14.4. DISPLAY DEVICES

The primary display device in a graphics system is a video monitor. The operation of most video monitors is based on the standard Cathode Ray Tube (CRT) design, but several other technologies exist.

Two types of CRTs are (*a*) Refresh CRT, and (*b*) Storage CRT.

Refresh Cathode Ray Tubes

Figure 14.2 illustrates a simplified diagram of a CRT showing its major components. It consists of an evacuated glass tube enclosing a source of electrons emitted by the source, a focusing system to constrain the accelerated electrons in a narrow beam, a horizontal

and vertical deflection system to deflect the beam, and a phosphor-coated screen which glows when an electron beam strikes it.

Basic Operation

Figure 14.2 illustrates the basic operation of a CRT. A beam of electrons (cathode rays), emitted by an electron gun, passes through focusing and deflection systems that direct the beam toward specified positions on the phosphor-coated screen. The phosphor then emits a small spot of light at each position contacted by the electron beam. Because the light emitted by the phosphor fades very rapidly, some method is needed for maintaining the screen picture. One way to keep the phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back over the same points. This type of display is called a refresh CRT.

Primary Components

1. Electron gun: The assembly of electron source, accelerator and focusing system is known as an electron gun. Electrons are generated by a cathode heated by an electric filament. The electrons are then accelerated by application of an electric field.

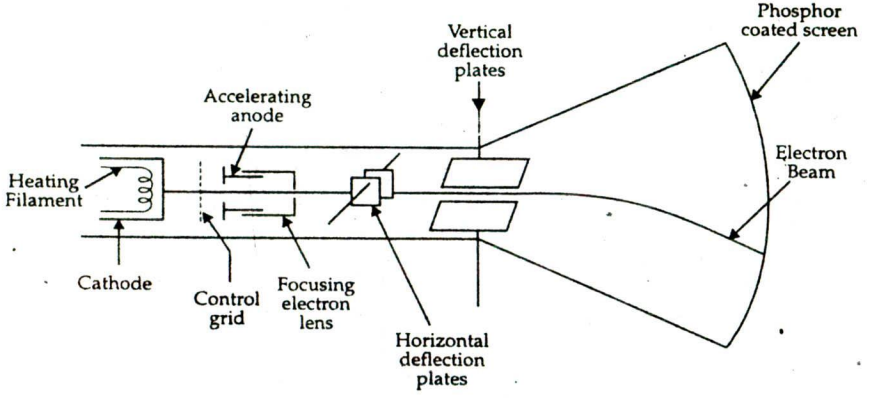


Fig. 14.2: Simplified diagram of a CRT display

2. Control grid: Intensity of the electron beam is controlled by setting voltage levels on the control grid, which is a metal cylinder that fits over the cathode. A high negative voltage applied to the control grid will shut off the beam by repelling electrons and stopping them from passing through the small hole at the end of the control grid structure. A smaller negative voltage on the control grid simply decreases the number of electrons passing through. Since the amount of light emitted by the phosphor coating depends on the number of electrons striking the screen, the brightness of a display is controlled by varying the voltage on the control grid.

3. Deflection system: A set of yoke sealed inside the neck of the tube forms part of the deflection system responsible for addressing in the Cathode Ray Tube. In this system, two pairs of coils or yokes are used, one to the control vertical deflection and other for the horizontal deflection. The beam from the electron gun passes between a set of vertical deflection plates. By applying a positive voltage to the top plate the beam can be moved up. It can be moved down by applying a positive voltage to the bottom plate. The beam

can thus be made to trace a line along the vertical or Y-direction by controlling the voltage applied to the vertical deflection plates.

Following the vertical deflection plates is a pair of horizontal deflection plates. If a positive voltage is applied to the left plate, the beam will move left along the horizontal direction. It moves right if a positive voltage is applied to the right plate. Thus, the beam can be made to trace a line along the horizontal or X-direction by controlling the voltage applied to the horizontal deflection plates. The deflection system deflects rapidly, since speed of deflection determines how much information can be displayed without flicker. The large-amplitude currents have been used in the yoke to achieve fast deflection. An important part of the deflection system is, therefore, the set of amplifiers that convert the small voltages received from the display controller into currents of the appropriate magnitude.

4. Phosphorus-coated screen: The inside front surface of every CRT is coated with special crystals called phosphorus which glow when they are hit by a high-energy electron beam. They continue to glow for a distinct period of time (the exact time and colour is unique to each type of phosphor) after CRT beam is removed. The time taken for the spot to disappear, when the beam is blanked, is known as the persistence time. Lower persistence phosphors require higher refresh rate (the number of times a picture is drawn in a second) to maintain a picture on the screen without flicker. A phosphor with low persistence is useful for animation; a high-persistence phosphor is useful for displaying highly complex static pictures.

Raster Scan and Random Scan

Scanning in refresh type video displays can be done in two different ways. These are called raster and random scans.

The most common type of graphics monitor employing a CRT is the raster-scan display, based on television technology. In a raster scan, the viewing screen is divided into a large number of tiny phosphor picture element, often referred to as pixels. They are arranged into a matrix, called a raster, that represents the display area of the screen. A raster scan display typically consists of a raster of 256 by 256 pixels, but it can go up to 1024 by 1204 pixels, providing a total over 1 million pixels.

During operation, the electron beam follows a fixed path. The beam starts at the upper left corner of the screen and moves horizontally to the right. This defines a scan line. The electron beam energises a small phosphor element as it passes a pixel along the line during a sweep. During the scan, the intensity of the beam is modulated according to the pattern of the desired image along the line. At the right edge of the screen, the beam is blanked and moved to the left edge, positioned one unit below the previous starting point. This is shown as a dotted line. This pattern of scanning is continued until

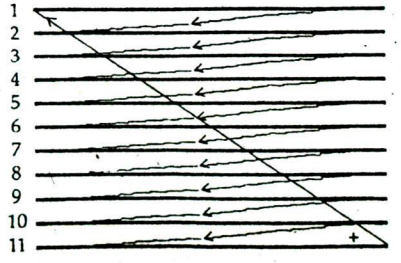


Fig. 14.3: The raster pattern on a non-interlaced CRT

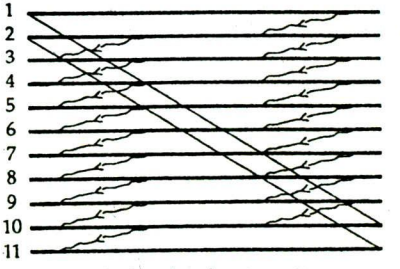


Fig. 14.4: An interlaced raster pattern on a CRT

the bottom corner on the right hand is reached. At this point, one scan is completed, and the beam is then repositioned at the upper left corner of the screen for starting another.

Refreshing on raster scan displays is carried out at the rate of 60 to 80 frames per second, although some systems are designed for higher refresh rates. Generally there are two commonly used methods to refresh a raster display: (*i*) sequential or non-interlaced scanning and (*ii*) interlaced scanning.

In sequential scanning, the sweeping of each horizontal line starts from the top to the bottom of the screen sequentially. The disadvantage of this type of scanning is that if the refresh rate is not matched with the persistence of the phosphor, the top part of the display appears to fade while the bottom part is still being scanned.

Often an interlacing is used to break the raster lines into two passes. In the first pass, the beam sweeps out the remaining scan lines. Although interlacing has been a popular method on standard television sets, it presents many problems to the graphics programmer.

Random scan: When operated as a random scan display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is drawn. Random scan monitors draw a picture one line at a time and for this reason are also referred to as vector writing or caligraphy display. With random scan, creating line diagram is easier. It is, therefore, more suitable for engineering, scientific drawing while raster graphics can be used to provide animation. Refresh rate on a random scan system depends on the number of lines to be displayed. The random scan requires a vector generator. Picture definition is stored as a set of line-drawing commands. A pen plotter is an example of a random scan, hard copy device.

Measures of CRT Quality

The following are important to measure the quality of CRT:

Brightness/luminance/life
Contrast
Resolution
Square pixels
Interlace
Refresh Rate
Bandwidth
Phosphor Persistence
Colour
Colour Purity
Convergence
Linearity
Size
Maintenance

A high quality CRT is one which provides an image that is stable and undistorted. A high quality CRT meets its specifications for resolution, and shows no perceptible flicker. A high quality CRT continues to be a high quality CRT even after its continuous use over a period of two to four years.

Storage CRT (Direct View Storage Tubes)

CRTs can be used in another way instead of refreshing the screen, the picture is stored inside the CRT. CRTs of this type are called Direct View Storage Tube (DVST). The DVST is similar in construction to the conventional electrostatically controlled CRT, but has an additional grid electrode and a second electron gun, called a flood gun, which generates

continuous stream of low speed electrons which are attached by the positively charged areas of storage grid and pass through it to the screen without affecting the positive charge pattern on the storage grid.

As opposed to a raster image, it is stable and unchanging and the main advantage of DVST is its low cost, but whenever any part of an image had to be changed or deleted, the storage image would have to be erased completely and the whole image redrawn together with the modification. Such CRTs are not suitable for interactive graphics display.

14.5. POINT PLOTTING TECHNIQUE

A point is basically represented as a dot using one pixel and this is the most visible elementary output for a graphics program.

Point plotting techniques are based on the cartesian coordinate system. Points are represented in two dimensions by an ordered pair or coordinates (x, y). The computer screen can be thought to have two axes, namely, x axis (abscissa) and y axis (ordinate). It depends upon the implementation of the graphics system software to establish the origin at a corner of the screen and of placing the origin at the lower left corner of the screen is fairly common.

Points are plotted in response to digital signals from the computer. For example, if x and y are passed to the display, each as a 10-bit binary number, there can be only 2^{10} (=1024) distinct x-coordinate values and only 10^{24} for y. Then, the screen offers a 1024 × 1204 array of positions at any one of which a point may be displayed. A display command for a point can be

Plot (x, y)

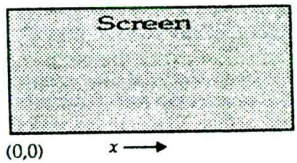


Fig. 14.5: Screen coordinator

Point plotting techniques can be used to draw complex figures by selecting several points and connecting successive points by straight lines.

14.6. LINE GENERATION TECHNIQUES

A line segment is a basic entity in creating computer graphics. A line segment may be specified by its two end points. From these two end points one can determine the equation of the line. The equation of the line is used to determine the x, y coordinates of all the points that lie between these two end points.

The problem is to select pixels on the computer screen which lie near to the line segment. We might try to turn on every pixel through which the line segment passes but it would not be easy to find all such pixels and the apparent thickness of the line would change with slope and position. To generate intermediate points on a line segment, the following criteria should be met:

Straightness: The selection should be such that the line drawn on the display screen should appear straight.

Density: The number of pixels used to generate a line segment should be equally spaced. The density of a line is defined by the ratio of the number of pixels displayed to the length of the line. The line density should not vary with the slope and the length of the line.

Speed: The amount of computation should be minimum so that line segments be drawn quickly for interactive applications.

Incremental techniques are very effective in generating coordinates of the points between the two end points of the line segments by selection of increments in x or y or both directions. The various techniques of line drawing are:

Symmetrical DDA: DDA stands for Digital Differential Analyser. It is a technique to determine the pixels which should be turned on computer screen. The name comes from the fact that we use the same technique as a numerical method for solving differential equations.

The equation for a straight line is in the form $y = mx + c$ with $m = \frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x}$, the slope of the line and (x_1, y_1) and (x_2, y_2) are the given points and b is the y intercept. We can determine the intermediate points starting with one end point, say, (x_1, y_1). The new y coordinates can be written as

$$y_1' = y_1 + dy$$
$$= y_1 + \frac{y_2 - y_1}{x_2 - x_1} dx$$

DDA simultaneously increments x and y by an amount proportional to the first derivatives of x and y. In case of straight lines, the derivatives are constant. Thus, the increments of x and y become proportional to Δx and Δy and can be taken as $\alpha \Delta x$ and $\alpha \Delta y$ where $\alpha = 2^{-n}$ and $2^{n-1} \leq \max(|\Delta x|, |y|) < 2^n$.

Thus, α is the reciprocal of the DDA's line-length estimate (2^n). The desired result is achieved by using of arithmetic overflow. x and y are kept in registers that have two parts, integer and fractional. The increments $\alpha \Delta x$ and $\alpha \Delta y$ are repeatedly added to the fractional parts. If the addition of fractional parts and increments overflow, *i.e.*, exceed the magnitudes, the corresponding integer part is incremented. The new integer parts of x and y registers are used in plotting the line.

The symmetrical DDA algorithm can be written as follows:

Step 1	Read	(x_1, y_1, x_2, y_2)
Step 2		$\Delta x = x_2 - x_1$
		$\Delta y = y_2 - y_1$
		$L = \text{sqrt}(\Delta x * \Delta x + \Delta y * \Delta y)$
		$\alpha = 1/L$
Step 3	Plot	(Int (x_1), Int (y_1))
Step 4	If	$((x_1) > (x_2))$, stop
Step 5.		$x_1 = x_1 + \alpha \Delta x$
		$y_1 = y_1 + \alpha \Delta y$
Step 6.	Plot	(Int (x_1), Int (y_1)),
	GO TO Step 4	
	End - of - Algorithm.	

Example 14.1. *If a line is drawn from (0, 0) to (10, 5) with a symmetric DDA, (i) how many iterations are performed, and (ii) how many distinct points are generated?* **(AMIE, S '93)**

Solution. Here $x_1 = 0$, $y_1 = 0$, $x_2 = 10$ and $y_2 = 5$

Hence,
$$\Delta x = x_2 - x_1 = 10 - 0 = 10$$
$$\Delta y = y_2 - y_1 = 5 - 0 = 5$$
$$L = \sqrt{\Delta x \times \Delta x + \Delta y \times \Delta y} = \sqrt{10 \times 10 + 5 \times 5}$$
$$= \sqrt{125} = 11.18$$
$$\alpha = 1/L = 1/11.18 = 0.089$$
$$\alpha \Delta x = 0.089 \times 10 = 0.89$$

and
$$\alpha \Delta y = 0.089 \times 5 = 0.445$$

The points generated by symmetric DDA are shown below:

No. of Iterations	$x_1 = x_1 + \alpha \Delta x$	$y_1 = y_1 + \alpha \Delta y$	*Coordinates Generated*
1.	0 + 0.089 × 10 = 0.89	0 + 0.089 × 5 = 0.445	(0, 0)
2.	0.89 + 0.089 × 10 = 1.78	0.445 + 0.089 × 5 = 0.990	(1, 0)
3.	1.78 + 0.89 = 2.67	0.990 + 0.445 = 1.435	(2, 1)
4.	2.67 + 0.89 = 3.56	1.435 + 0.445 = 1.880	(3, 1)
5.	3.56 + 0.89 = 4.47	1.880 + 0.445 = 2.325	(4, 2)
6.	4.47 + 0.89 = 5.36	2.325 + 0.445 = 2.770	(5, 2)
7.	5.36 + 0.89 = 6.25	2.770 + 0.445 = 3.215	(6, 3)
8.	6.25 + 0.89 = 7.14	3.215 + 0.445 = 3.660	(7, 3)
9.	7.14 + 0.89 = 8.03	3.660 + 0.445 = 4.105	(8, 4)
10.	8.03 + 0.89 = 8.92	4.105 + 0.445 = 4.550	(8, 4)
11.	8.92 + 0.89 = 9.81	4.550 + 0.445 = 4.995	(9, 4)
12.	9.81 + 0.89 = 10.70	4.995 + 0.445 = 5.440	(10, 5)

Hence, the number of iterations are 12 and the distinct points generated are 11.

Simple DDA: In order to speed up the computation, the increment of α is chosen so that the larger of $\alpha \Delta x$ and $\alpha \Delta y$ becomes the increment step. Mathematically, this is achieved by rounding the real values of new coordinates x and y. The first point is obtained by adding 0.5 to both x and y and taking only their integral parts.

The simple DDA algorithm can be written as follows:

```
Read (x1, y1, x2, y2)
Lx = abs (x2 − x1)
Ly = abs (y2 − y1)
L = max (Lx, Ly)
Incr x = (x2 − x1)/L
Incr y = (y2 − y1)/L
x = x1 + 0.5
y = y2 + 0.5
For i = 1 to L do
Plot (Int (x), Int (y))
        x = x + Incr x
        y = y + Incr y
End {For i}
End-of-Algorithm.
```

Example 14.2. *If a line is drawn from (0, 0) to (5, 4) with a simple DDA, (i) how many iterations are performed, and (ii) how many distinct points are generated?*

Solution. Here $x_1 = 0$, $y_1 = 0$, $x_2 = 5$ and $y_2 = 4$

$$Lx = |5 - 0| = 5$$
$$Ly = |4 - 0| = 4$$
$$L = \max(5, 4) = 5$$
$$\text{Incr } x = (5 - 0)/5 = 1.0$$

Incr $y = (4 - 0)/5 = 0.8$

First point, $x = 0 + 0.5 = 0.5 = 0$ (integer value),

$y = 0 + 0.5 = 0.5 = 0$ (integer value)

The other points can be found by iteration as shown below:

No. of Iterations	$x = x + Incr\ (x)$	$y = y + Incr\ (y)$	*Coordinates Generated*
1.	0.5	0.5	(0, 0)
2.	1.0 + 0.5 = 1.5	0.5 + 0.8 = 1.3	(1, 1)
3.	1.5 + 1.0 = 2.5	1.3 + 0.8 = 2.1	(2, 2)
4.	2.5 + 1.0 = 3.5	2.1 + 0.8 = 2.9	(3, 2)
5.	3.5 + 1.0 = 4.5	2.9 + 0.8 = 3.7	(4, 3)
	4.5 + 1.0 = 5.5	3.7 + 0.8 = 4.5	(5, 4)

Number of iterations are 5 and the points generated are 5.

Bresenham's Line Algorithm

This method finds the closest integer coordinates to the actual line path by using integer arithmetic which makes it more efficient. The algorithm to select the optimum raster locations to represent a straight line. To accomplish this, the algorithm always increments by one unit in either x or y depending on the slope of the line. The increment in the other variable, either zero or one, is determined by examining the distance between the actual line location and the nearest grid locations. For example, if (x_i, y_i) is the last pixel position which has been plotted, the next pixel position is decided between $(x_i + 1, y_i)$ and $(x_i + 1, y_i + 1)$.

Let the distance of those pixels lying immediately above the true line be P_i and the distance to those directly below of it be S_i and $e_i = S_i - P_i$. When $e_i < 0$, the closest pixel in the raster will be the pixel below the true value and when $e_i \geq 0$, the pixel immediately above the true line is closest.

Initially set $e_i = 2\Delta y - \Delta x$, where $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$. Thereafter, if $e_i \geq 0$, then $x_{i+1} = x_i + 1$, $y_{i+1} = y_i + 1$ and $e_{i+1} = e_i + 2\ (\Delta y - \Delta x)$. If $e_i < 0$, then $x_{i+1} = x_i + 1$ and $e_{i+1} = e_i + 2\Delta y$.

For straight lines whose slope of the angle is greater than 45° and less than 90°, the algorithm must be inverted so that just y or both x and y are incremented at each step.

The Bresenham's algorithm for the line whose slope angle lies between 0 and 45° is given below:

Read (x_1, y_1, x_2, y_2)

$x = x_1$

$y = y_1$

$\Delta x = x_2 - x_1$

$\Delta y = y_2 - y_1$

$e = 2\Delta y - \Delta x$

For $i = 1$ to Δx do

Plot (x, y)

If $e > 0$, then

$y = y + 1$

$e = e + 2\Delta y - 2\Delta x$

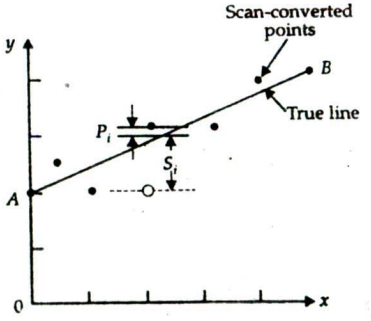


Fig. 14.6: Scan-converted line

Else

$e = e + 2\Delta y$

End if

$x = x + 1$

Next i

End-of-Algorithm.

Example 14.3. *If a line is drawn from (1, 1) to (7, 5), how many distinct points are generated?*

Solution. Here $x_1 = 1,\ y_1 = 1,\ x_2 = 7,\ y_2 = 5$

$$\Delta x = x_2 - x_1 = 7 - 1 = 6$$
$$\Delta y = y_2 - y_1 = 5 - 1 = 4$$
$$e = 2\Delta y - \Delta x = 2 \times 4 - 6 = 2$$

The following table indicates the points generated by the algorithm:

No. of Iterations	*Coordinates Generated*	$e = e - 4,\ e > 0$ $= e + 8,\ e < 0$
1	(1, 1)	2
2	(2, 2)	–2
	(3, 2)	6
3	(4, 3)	2
4	(5, 4)	–2
5	(6, 4)	6
	(7, 5)	

Hence, the number of iterations required is 5 and 5 distinct points are generated.

14.7 CIRCLE GENERATING ALGORITHMS

Circle generating algorithms can be developed based on the line drawing algorithms. Here it is discussed that how the curved shapes, with the help of line segment and point plotting techniques, are employed to generate circles and ellipse algorithms in special reference. The equation of a circle in Pythagorean theorem (Fig. 14.7) is given by

(*a*) *Cartesian co-ordinate system*

$$(X - XC)^2 + (Y - YC)^2 = R^2$$

or $$Y = YC \pm SQRT\ (R^2 - (X - XC)^2) \quad \ldots (1)$$

(*b*) *Polar co-ordinate system*

$$X = XC + R \times \cos\theta \quad \ldots(2)$$
$$Y = YC + R \times \sin\theta$$

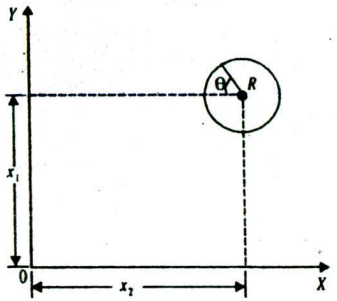


Fig. 14.7: Circle representing its centre at (X_C, Y_C) and radius R

To evaluate pixel positions along the circumference of the circle using Eq. (1) or (2), too many calculations are required. These calculations can be minimized in Bresenham's circle generating algorithm.

14.7.1. Bresenham's Circle Generating Algorithm

Alike line generating algorithm, the choice for next pixel position may be such that out of two pixel positions, algorithm must choose such position which is nearer to the specific circular path at each forward step. For simplicity, it is assumed that the centre of the

circle is at origin of co-ordinate system, i.e., $XC = 0$, $YC = 0$. The unit increment of x-co-ordinate will start from $X = 0$ to $X = R$. Thus, initial co-ordinate is $(0, R)$ and final co-ordinate is $(R, 0)$. Now, consider that at initial position, the pixel is at (X_i, Y_i) and closer to the path of the circle (Fig. 14.8). The next pixel position may either be $(X_i + 1, Y_i)$ or $(X_i + 1, Y_i - 1)$. Using Eq. (1), the actual value of y obtained on the circle path is

$$y = \text{SQRT}\ (R*R - (x_i + 1)^2)$$

or
$$y^2 = (R^2 - (x_i + 1)^2)$$

Suppose d_1 and d_2 be the difference parameters with respect to the actual position of y-co-ordinate on the circle path as shown in Fig. 14.9, then

$$d_1 = y_{i2} - y^2 = y_{i2} - R^2 + (x_i + 1)^2 \quad (3)$$

and
$$d_2 = y^2 - (y_i - 1)^2 = R^2 - (x_i + 1)^2 - (y_i - 1)^2 \quad (4)$$

and
$$P_i = d_1 - d_2 = 2\,(x_i + 1)^2 + y_{i2} + (y_i - 1)^2 - 2R^2 \quad (5)$$

If the value of P_i is less than 0, select the pixel position of y_i. Otherwise, select the pixel positon at $(y_i - 1)$.

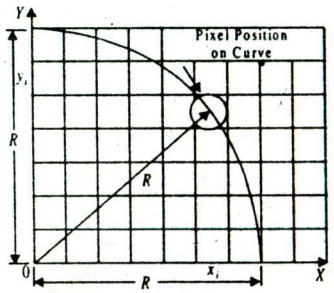


Fig. 14.8: Initial pixel position in an arbitrary location at (x_i, y_i)

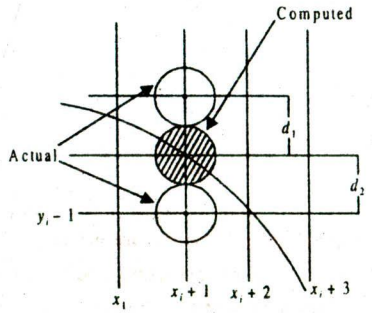


Fig. 14.9: Actual and computed y positions

Similarly, for next pixel position,

$$P_{i+1} = 2\,((x_i + 1) + 1)^2 + (y_{i+1})^2 + (y_{i+1} - 1)^2 - 2R^2$$

or
$$P_{i+1} = P_i + 4x_i + 6 + 2\,((y_{i+1})^2 - y_{i2}) - 2\,(y_{i+1} - y_i)$$

To evaluate P_i at the starting point, $x_1 = 0$, $y_1 = R$ and substituting these values in Eq. (5),

$$P_1 = 3 - 2R$$

On the circle path, the actual value of y position at $(y_i + 1)$ is either y_i or $(y_i - 1)$, depending upon the value of P_i. There exist two conditions:

Condition 1

If P_i is less than zero, the next pixel position will be either $(x_1 + 1, y_1)$ or $(x_1 + 1, y_1 - 1)$ and all successive co-ordinates will be evaluated by the same analogy till $x = R$ and $P_{i+1} = P_i + 4x_i + 6$.

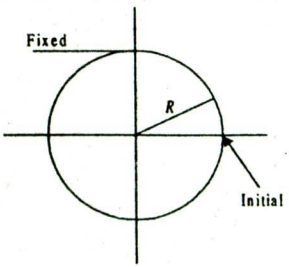


Fig. 14.10: Initial and final position of co-ordinate system

Condition 2

If P_i is greater than or equal to 0, $P_i + 1 = P_i + 4(x_i - y_i) + 10$. Again, if $P_{i+1} < 0$, the next pixel position will be selected as $(X_i + 2,\ Y_i + 1)$ and if $P_{i+1} \geq 0$, the next pixel position will be selected as $(x_i + 2,\ y_{i+1} - 1)$.

14.8. TWO-DIMENSIONAL GRAPHICS TRANSFORMATIONS

A graphic system allows the programmer or user to move and manipulate the picture on the computer screen. The manipulation of pictures can be performed through transformation because the graphic images are coded as numbers and stored in the memory. The numbers can be modified by mathematical operations, called transformations. The major basic transformations are:

(*a*) Translation
(*b*) Rotation
(*c*) Scaling
(*d*) Reflection.

These basic transformations can be combined or concatenated to perform a compound transformations. There are two basic ways of describing object movement:

(*i*) **Geometric transformations:** When the coordinate system is stationary and each point of the object is moved or transformed.
(*ii*) **Coordinate transformations:** When the object remains stationary while the coordinate system is moved relative to the object.

An example involves the motion of a car against a scenic background. This can be simulated by (*i*) moving the car while keeping the background fixed (a geometric transformation), and (*ii*) keeping the car fixed while moving the background scenery (a co-ordinate transformation).

Geometric Transformation

An object can be thought of as a set of points, every point having specific position in a coordinate system. When an object is translated or rotated, every point on the object is translated by same amount and every point on the object is rotated through the same angle. The major basic geometric transformations are discussed here.

Translation: A translation is a displacement of an object in a given direction through a given magnitude. We translate a point from coordinate $P(x, y)$ to new $P'(x', y')$ by adding translation distances t_x and t_y to the original coordinates, i.e., $x' = x + t_x$, $y' = y + t_y$, and the translation vector is given by

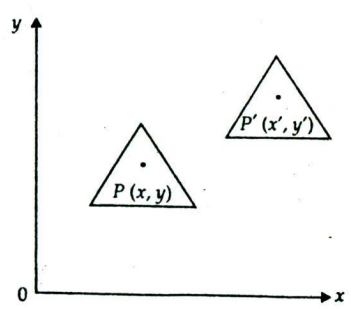


Fig. 14.11: The translation of a triangle

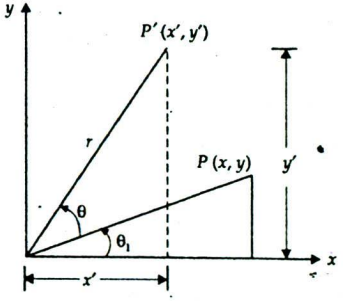


Fig. 14.12: Rotation of a position vector

$$\vec{v} = t_x \hat{i} + t_y \hat{j}$$

Since a solid object has definite shape, therefore, the new coordinates of all points can be obtained by applying the same translation vectors.

Rotation: Transformation of object path along circular path is called rotation. In Fig. 14.12, the position vector $P(x, y)$ is rotated through an angle θ about the origin. The convention is that the direction of rotation is counterclockwise if θ is a positive angle and clockwise if θ is a negative angle. Let $P'(x', y')$ be the new position of P.

Since $P'(x', y')$ is at an angle $(\theta_1 + \theta)$ from x-axis, therefore, x' and y' now become

$$x' = r \cos(\theta_1 + \theta)$$

$$x' = r \cos\theta_1 \cos\theta - r \sin\theta_1 \sin\theta$$

and

$$y' = r \sin(\theta_1 + \theta)$$

$$= r \sin\theta_1 \cos\theta + r \cos\theta_1 \sin\theta$$

or

$$x' = x \cos\theta - y \sin\theta$$

$$y' = y \cos\theta + x \sin\theta = x \sin\theta + y \cos\theta$$

as

$$x = r \cos\theta_1 \quad \text{and} \quad y = r \sin\theta_1$$

In matrix form, $[x'\ y'] = [x\ y] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

Thus, the transformation for a general rotation about the origin by an arbitrary angle θ is

$$[T] = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

For example, if we use a 3×2 matrix containing the x and y coordinates of the triangle's vertices $A(5, -1)$, $B(6, 1)$ and $C(4, 1)$ and the triangle is rotated through $90°$ about the origin in a counterclockwise sense, then

$$\begin{bmatrix} 5 & -1 \\ 6 & 1 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} \cos 90° & \sin 90° \\ -\sin 90° & \cos 90° \end{bmatrix} = \begin{bmatrix} 5 & -1 \\ 6 & 1 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ -1 & 6 \\ -1 & 4 \end{bmatrix}$$

which produces the triangle $A'B'C'$ (Fig. 14.13).

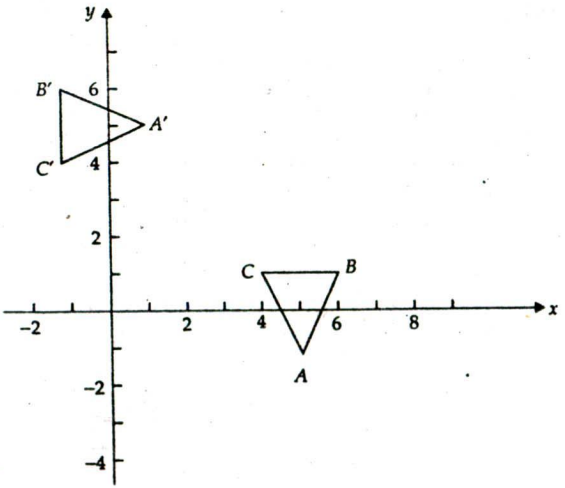


Fig. 14.13: Rotation

Scaling: Scaling is the process of expanding or compressing the dimensions of an object. Positive scaling constants S_x and S_y are used to describe changes in x and y directions, respectively. A scaling constant greater than 1 means expansion of length, called magnification, while a scaling constant less than 1 compresses the length and is called reduction. The scaling transformation is given by

$$x' = S_x \cdot x$$
$$y' = S_y \cdot y$$

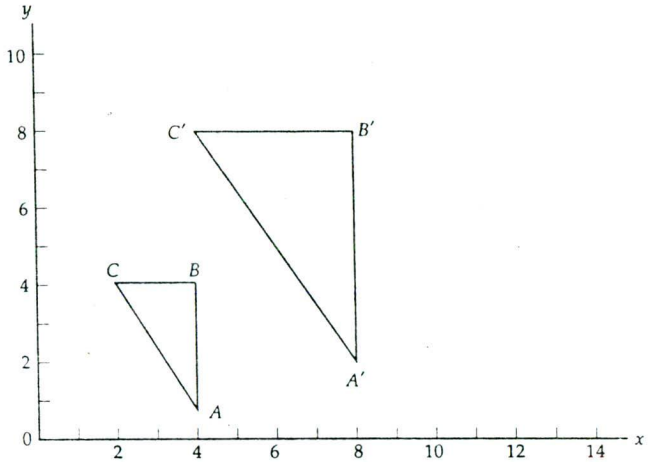


Fig. 14.14: Uniform scaling

In matrix form, if we select $[T] = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$, then the new point (x', y') can be computed by the following equation:

$$[x' \quad y'] = [x \quad y]\,[T] = [xS_x \quad yS_y]$$

If both scaling constants have the same value, then scaling is said to be uniform. If the magnitudes are unequal, a distortion occurs, and the scaling is said to be non-uniform. For example, if the matrix $[T] = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ is used as an operator on the vertices A (4, 1), B (4, 4) and C(2, 4) of the triangle ABC, these effects are shown in Fig. 14.14. Triangle ABC is transformed to yield $A'B'C'$ where a uniform scaling occurs.

Reflection: If either the x-axis or the y-axis is treated as a mirror, the object has a mirror image or reflection. If we select

$$T = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

then the new point (x', y') will be reflection about the y or x axis. Reflection about the y axis is defined by

$$[x' \quad y'] = [x \quad y] \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = [-x \quad y]$$

Similarly, reflection about the x axis is given by

$$[x' \quad y'] = [x \quad y] \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = [x \quad -y]$$

A reflection about the line $y = x$ occurs for

$$[T] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Similarly, a reflection about the line $y = -x$ is given by

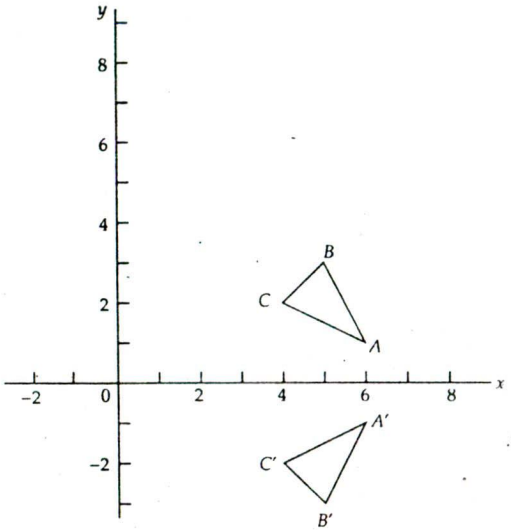


Fig. 14.15: Reflection about the x-axis

$$[T] = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

Two reflections of the triangle ABC with vertices $A(6, 1)$, $B(5, 3)$ and $C(4, 2)$ about the x-axis is shown in Fig. 14.15. The transformed new vertices $A'B'C'$ are given by

$$\begin{bmatrix} 6 & 1 \\ 5 & 3 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 6 & -1 \\ 5 & -3 \\ 4 & -2 \end{bmatrix}$$

Combined Transformations

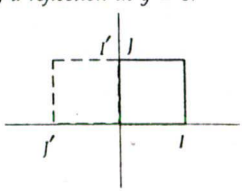
By performing matrix operations on the position vectors which define the vertices, the shape and position of the surface can be controlled. However, a desired orientation may require more than one transformation. Since matrix multiplication is non-commutative, the order of application of transformations is important.

Example 14.4. *Use matrices to find the single transformation which is equivalent to a rotation of 90° about the origin, followed by a reflection in $y = 0$.*

Solution.

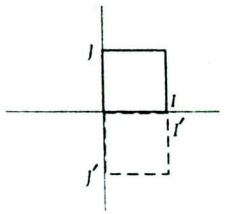
Rotation of 90° about (0, 0):

Matrix $\mathbf{A} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$



Reflection in $y = 0$:

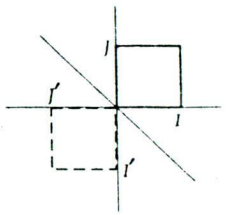
Matrix $\mathbf{B} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$



Combined matrix = $\mathbf{BA} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$

and the matrix $\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$

defines a reflection in the line $x + y = 0$.



Thus, a rotation of 90° about the origin followed by a reflection in $y = 0$ is equivalent to a reflection in $x + y = 0$.

Matrix Representations and Homogeneous Coordinates

Transformation equations can be written in a consistent matrix form by first expressing points as homogeneous coordinates. This means that a two-dimensional coordinate position (x, y) is represented as the triple $[x_h \;\; y_h \;\; w]$, where

$$x_h = x \cdot w, \; y_h = y \cdot w$$

The parameter w is assigned a non-zero value in accordance with the class of transformations to be represented. For the two-dimensional transformations discussed in the last section, one can set $w = 1$. Each two-dimensional coordinate position then has the homogeneous coordinate form $[x \;\; y \;\; 1]$, other values for w are useful with certain three-dimensional viewing transformations. With coordinate positions expressed in homogeneous form, the basic transformation equations can be represented as matrix multiplications employing 3×3 transformation matrices as follows:

Translation : $$[x' \;\; y' \;\; 1] = [x \;\; y \;\; 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \quad ...(6)$$

Rotation: $$[x' \;\; y' \;\; 1] = [x \;\; y \;\; 1]\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ...(7)$$

Scaling: $$[x' \;\; y' \;\; 1] = [x \;\; y \;\; 1]\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ...(8)$$

By solving equations (6), (7) and (8),

$$x' = x + t_x$$
$$y' = y + t_y$$
$$x' = x \cos\theta + y \sin\theta$$
$$y' = -x \sin\theta + y \cos\theta$$

$$x' = x\, S_x$$
$$y' = y\, S_y$$

Inverse Coordinate Transformations

Each coordinate transformation has an inverse which can be found by applying the opposite transformation:

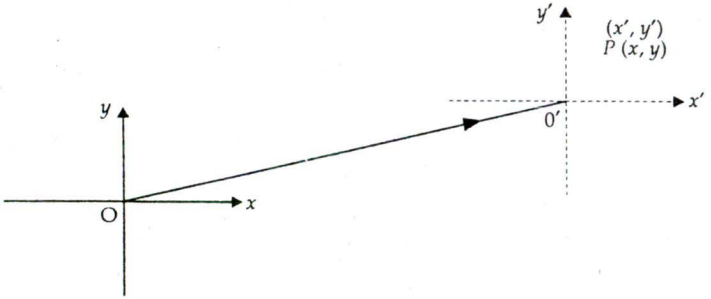


Fig. 14.16

Translation: $T_v^{-1} = T_{-v}$, translation in the opposite direction

Rotation: $R_\theta^{-1} = R_{-\theta}$, rotation in the opposite direction

Scaling: $S_{s_x, s_y}^{-1} = S_{1/s_x, 1/s_y}$

Mirror reflection: $M_x = M_x$ and $M_y = M_y$.

Coordinate Transformations

Let us consider two coordinate systems in the plane. Let 0 be the origin of the first system with coordinate axes xy and 0′ be the origin of the second system with $x'y'$ as coordinate axes. The second system can be obtained from the first system by transformation applied to the first system, known as coordinate transformation. Now, each point on the plane has two coordinates (x, y) or (x', y') (Fig. 14.16).

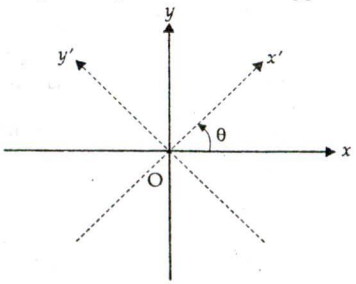


Fig. 14.17

Translation: If the xy coordinate system is displaced in a given direction through a given magnitude to form $x'y'$ coordinate system, then

$$x' = x - t_x, \qquad y' = y - t_y$$

where t_x and t_y are the translation distances.

Rotation: If the xy coordinate system is rotated through an angle about the origin, then $x' = x \cos \theta + y \sin \theta$ and $y' = x \sin \theta + y \cos \theta$ (Fig. 14.17).

Scaling: A new coordinate system is formed by leaving the origin and coordinate axes unchanged, but different units of measurement along the x and y axes. If the new units are obtained from the old units by a scaling of S_x units along the x axis and S_y units along the y axis, the coordinates in the new system are related to coordinates in the old system through the scaling transformation $x' = xS_x$, $y' = yS_y$. Figure 14.18 shows coordinate scaling transformation using scaling factors $S_x = 2$ and $S_y = 1/2$.

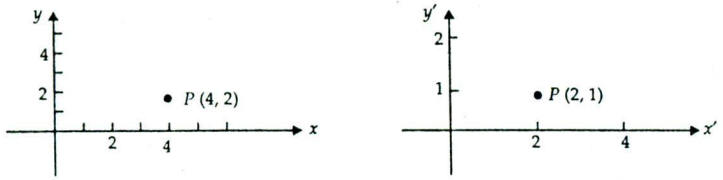


Fig. 14.14. Coordinate scaling transformation

14.9. THREE-DIMENSIONAL GRAPHICS TRANSFORMATIONS AND MODELLING

The three-dimensional transformations are an extension of the two-dimensional techniques. They are used in modelling to express locations of objects relative to others.

Translation: The transformation which translates a point (x, y, z) to a new point (x', y', z') is presented as follows:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

where t_x, t_y and t_z are the components of the translation x, y and z direction, respectively. By solving,

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

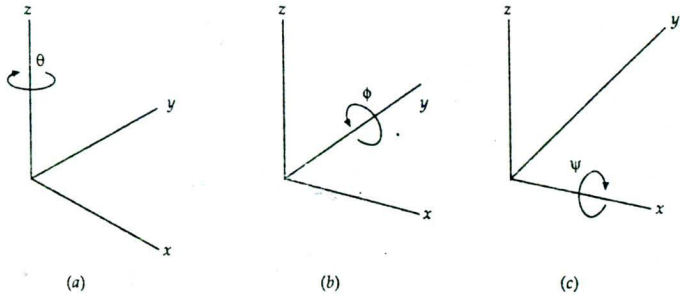


Fig. 14.19: Three-dimensional rotations

Rotation: Three-dimensional rotation transformations have more complexity than two-dimensional rotation transformations. In three dimensions, it is helpful to devise transformations for rotation about each of the three coordinate axes as shown in Fig. 14.19. Rotation about the z coordinate axis, through an angle θ, is given by

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation angle θ is measured clockwise about the origin when looking at the origin from a point on the +z axis.

Rotation about the y coordinate axis, through an angle θ, is given by

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about the x coordinate axis, through an angle ψ, is given by

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling: A scaling transformation could be used to scale dimensions in each coordinate direction separately:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projections

The representation on a plane of any three-dimensional object placed in front of this plane is called the projection of this object on the given plane. Thus, the projection is a view of an object thrown on to a plane by means of straight lines or rays, which are usually drawn through significant points of the object up to where they pierce the plane.

The lines or rays drawn from the object to the plane are called **projections.**

The plane on which the projections are drawn is known as **plane of projection.**

There are two basic methods for projecting three-dimensional objects on to a two-dimensional viewing surface. All points of the object can be projected to the surface along parallel lines, or the points can be projected along lines that converge to a position called the **centre of projection.** The two methods, called **parallel projection** and **perspective projection,** respectively, are illustrated in Fig. 14.20.

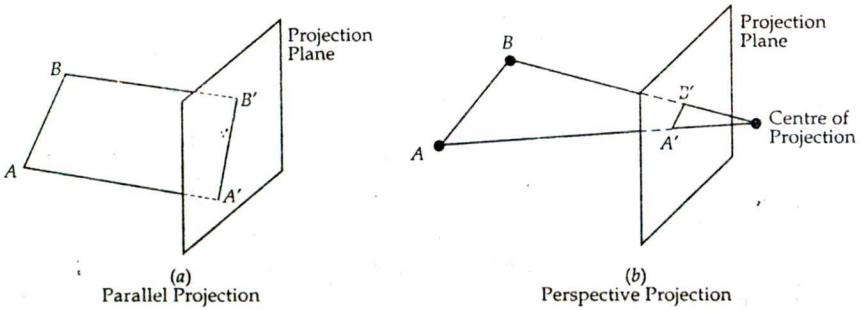


Fig. 14.20

Parallel Projections: A parallel projection is formed by extending parallel lines from each vertex on the object until they intersect the plane of the screen. The point of intersection is the projection of the vertex. The so called vertices are connected by line segments which correspond to connections on the original object. Views formed with parallel projections can be characterised according to the angle that the direction of projection makes with the projection plane.

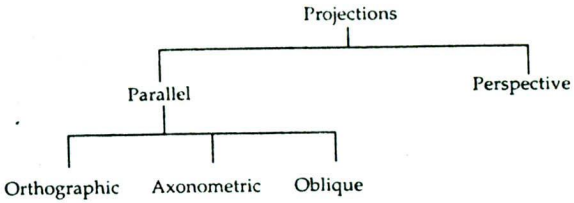


Fig. 14.21

Orthographic projections are characterised by the fact that the direction of projection is perpendicular to the view plane. When the direction of projection is parallel to any of the principal axes, this produces the front, top and side views of mechanical drawings.

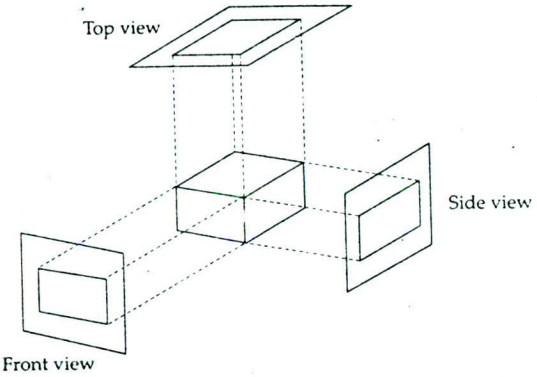


Fig. 14.22

Axonometric Projections: There are orthographic projections in which the direction of projection is not parallel to any of the three principal axes. An axonometric projection is constructed by manipulating the object, using rotations and translations, such that at least three adjacent faces are shown. When the direction of projection is not perpendicular to the view plane, it is called **oblique projection** (Fig. 14.23).

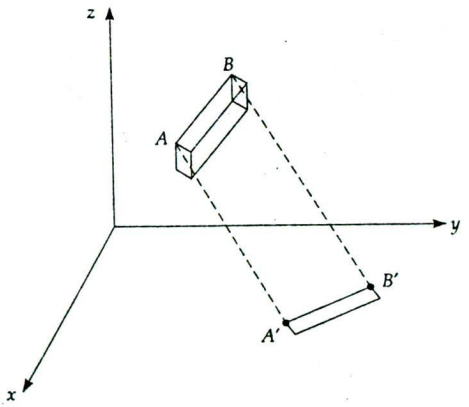


Fig. 14.23: Oblique projection

Perspective Projection: In a perspective projection, the lines of projection are not parallel. Instead they all converge at a single point, called the centre of projection (Fig. 14.24). It is the intersections of three converging lines with the plane of the screen that determine the projected image. It more closely approximates the view obtained by the human beings. The illusion that objects lengths look smaller as their distance from the centre of projection, called depth, increases is called **perspective foreshortening.** The illusion of meeting of certain sets of parallel lines at a point is called **vanishing point.**

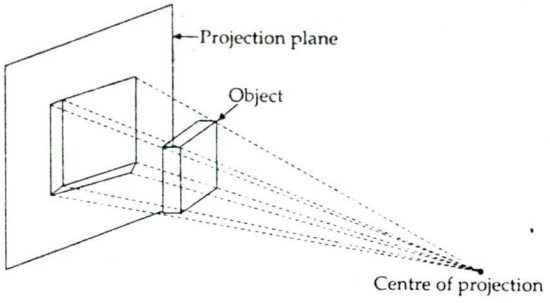


Fig. 14.24 : Perspective projection

Interactive Devices

Once the picture has been presented, interaction with or modification of the picture is required. To meet this requirement, a number of interactive devices like alphanumeric keyboard, joystick, track ball, digitizer, etc. are available.

Alphanumeric Keyboard

Several forms of keyboard terminals are available as CAD input devices. The most common type is the alphanumeric terminal which is available with nearly all interactive graphics system. The alpha refers to the key used for input letters and numeric refers to the key used for input number. It is used to enter commands, functions and supplemental data to the CAD system. A keyboard contains all letters, numbers, symbols and a regular typewriter, plus other keys which may include

(*i*) a numeric key pad which functions like a calculator,
(*ii*) function keys whose operation can be determined by the user of pre-programmed by the software being used,
(*iii*) cursor control keys which are used to select coordinate position by positioning the screen cursor on the computer screen.

Special purpose keyboards contain only the set of buttons, dials or switches to facilitate the selection of customized graphics operations required for a particular operation.

Joystick

A joystick consists of a small, vertical lever (called the stick) mounted on a base. The distance that the stick is moved in any direction from its centre position corresponds to screen cursor movement in that direction. By moving the lever forward and backward, the setting of one of the potentiometer changes. Similarly, moving the lever left or right, the setting of the other potentiometer changes. Thus, with the help of the same lever, it is possible to change or alter the position of the point in x and y directions simultaneously. Sometimes joysticks are constructed with one or more buttons that

can be programmed as input switches to signal some type of action once a screen position has been selected.

Track ball

The track ball is similar to the joystick. Here a spherical ball is mounted in a base with only a portion projecting above the surface. The ball is free to rotate in any direction. The cursor movement is controlled by the rotation of the ball. Two valuators, either potentiometers or shaft encoders, mounted in the base sense the rotation of the ball and provide results proportional to its relative position.

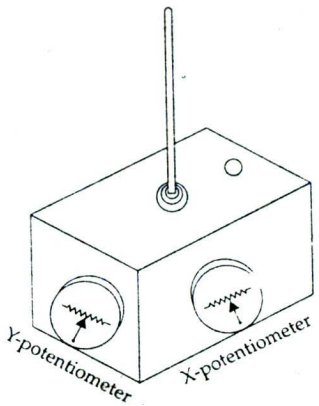


Fig. 14.25 : A joystick

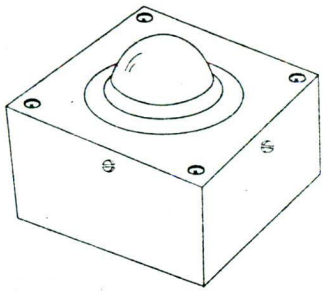


Fig. 14.26 : A track ball

Digitizers

The digitizer consists of a large, smooth board and an electronic tracking device which can be moved over the surface to follow existing lines. The electronic tracking device contains a switch for the user to record the desired x and y coordinate positions. The coordinates can be entered into the computer memory or stored on an off-line storage medium such as magnetic tape. High-resolution digitizers, typically with a large board (e.g., 42 inch by 60 inch) can provide resolution and accuracy of the order of 0.001 inch.

The tablet is a small low-resolution digitizer. The tablet serves as a surface corresponding to the CRT but remote to it. The user writes on the surface of the tablet, and the position of the pen is transmitted to the computer. The screen which then displays a cursor is an echo of the position of the pen on the tablet.

Modelling 3-D Objects

Three-dimensional models can be represented geometrically in three different ways:

(*i*) as a wireframe models,
(*ii*) as a surface model,
(*iii*) as a solid model.

Wireframe Model

In the construction of wireframe models, the edges of the object are displayed as lines (wires). For the objects in which there are curved surfaces, contour lines can be added to indicate the contour. The image assumes the appearance of a frame constructed out of wire, hence the name wireframe model.

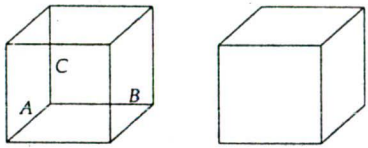


Fig. 14.27 : (*a*) Wireframe model of a cube, (*b*) A cube with hidden lines removed

But these models prove to be inadequate for 3-D model representation. For example, there might be ambiguity in the case of a definition, as to which side of the surface is solid. This type of limitation prevents the system from achieving comprehensive and unambiguous definition of object. To avoid these limitations, wireframe models are supported with Hidden Line Elimination.

Hidden Line Elimination

Removing the hidden edges of the model from display, is called Hidden Line Elimination (HLE). HLE improves the understanding of the 3-D wireframes displayed on a 2-D screen, with less ambiguities.

Many systems support the display of hidden edges, in a different colour or in a different line style. This helps in identifying the hidden parts of the object visually. Hidden Line Eliminated model display can also be plotted.

Surface Model

In surface modelling systems, models are developed by representing the shape of its outside surfaces. The surface model of an object represents the shell of the object, but not the internal details or mass properties. Several surface modelling techniques are used in the surface modelling systems as given.

Ruled surfaces: Surfaces are generated by moving a straight line through space with its end points tracing curves.

Surface of revolution: Surfaces are created by rotating a curve or line through a circular arc.

Sweep surfaces: Surfaces are generated by moving one curve through another.

Sculptured surfaces: Complex curved surfaces can be created by connecting surface shapes (patches) into a quilt like pattern to approximate the surface.

Surface models may be constructed using a large variety of surface features often provided by many CAD/CAM systems. As surface models do not actually represent the solid nature of parts because they contain no information describing what lies within the part interior, they cannot be used as a basis for engineering analysis programs such as finite element and model analysis for stress and strain predictions. These programs often require such properties as weight, volume and moments of inertia which cannot be derived from a surface model alone. Interpretation of a surface model can still be ambiguous.

Hidden Surface Removal

Removing the hidden surfaces of an object from its display is called Hidden Surface Removal (HSR). HSR is provided with 3-D modelling systems as an advanced feature and is supported by either part of the software or making use of the hardware facilities available on the system.

Shading

Shading is a tool for producing realistic object images on the computer graphics screen. Shading of objects requires a good number of colour shades (typically in thousands).

Many of the low and CAD systems may not have such hardware support. These facilities are generally available with high end CAD workstations.

Specularity

Specular reflection is a metallic feature. Specular reflection causes bright spots, reflecting the light source on metallic objects. Specularity gives more realism to the generated picture.

Light Sources

Generating a realistic looking display, requires a light source. Modelling packages allow the user to interactively position a light source. These light sources are taken as a pointed light source. Advanced software allow multiple light sources.

Depth-Cue

When 3-D wireframe models are displayed onto a 2-D screen, the depth information is lost. Depth-Cue models provide the depth-information along with the wireframe models, by highlighting the part of the object nearer to the eye point compared to the parts far away. The intensity of the light used for wireframe display is controlled, as an inverse function of the distance of the object points from the eye point.

Apart from the above basic visualisation tools, many advanced modelling packages support even the effects of reflection, refraction, etc., on the models displayed on the screen, giving it more realism.

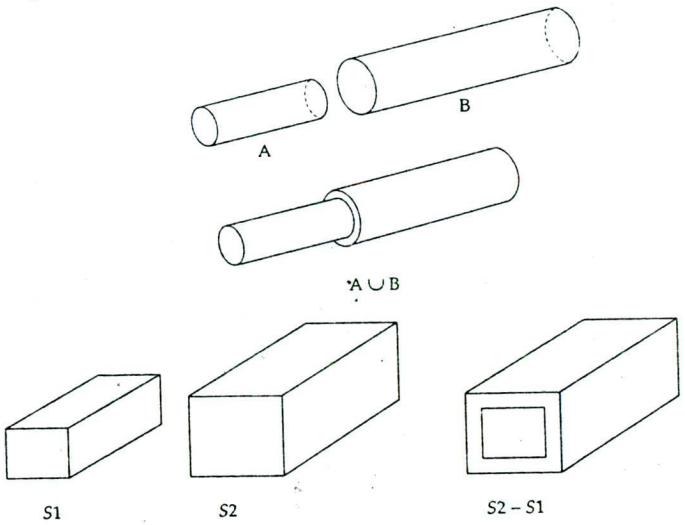


Fig. 14.28

Solid Model

All solid modelling systems provide facilities for creating, modifying and inspecting models of three-dimensional solid objects. The representation schemes may be divided into three general classes as follows:

(*i*) Constructive solid geometry (CSG);

(*ii*) Boundary representation (B-rep); and
(*iii*) Sweeps.

Constructive Solid Geometry (CSG)

This is basically a volume-oriented representation scheme, using a "building-block" approach to create 3-D geometric models out of simple solid shapes or primitives. The primitive solids can be rectangular blocks, spheres, cones, cylinders and other "primitive" volumes. Primitives can be combined to construct a solid model by Boolean or logical operations such as union, intersection or difference. For example, two primitives can be added together at some point with a union operation to form a part. A hole in a part can be created by intersecting the part with a negative cylinder. Figure 14.28 illustrates how a simple model can be constructed from primitives by logical operations. It may be proved that primitives for sphere, cone, cylinder and plane is sufficient to generate solid model for all objects. Other primitives such as cube or torus may be included for convenience.

Primitive modelling is certainly a powerful technique to model solid parts, but not all parts can be conveniently or best modelled with primitives. Complex shapes can be modelled much more easily and efficiently by boundary modelling.

Boundary Representation (B-Rep)

Boundary representation is a face-oriented representation scheme, in which objects are presented by the surfaces which bound the object. However, unlike surface models, the data structure in a boundary representation model contains enough information to determine whether a point lies on the inside or on the surface of the object. In this representation, a solid is represented by the data structure contains the elements which describes its boundary. These elements are divided in two categories: topological and geometric. The topology of an object is the property that describes how its vertices, edges and faces are connected. The geometry of an object defines the dimensions and positions of its vertices, edges and faces and, therefore, fixes it in space.

Wireframe *vs.* Solid Modelling

The two approaches have their relative advantages and disadvantages. The CSG systems usually have a significant procedural advantage in the initial formulation of the model. It is relatively easy to construct a precise solid model out of regular solid primitives by adding, subtracting, and intersecting the components.

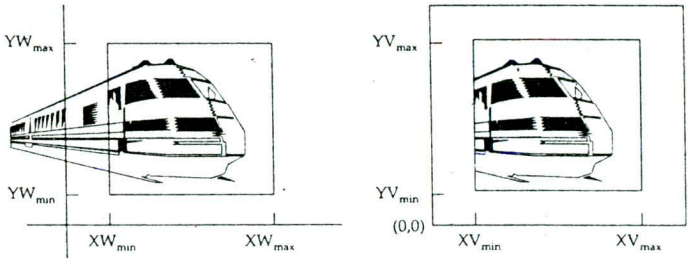
On the other hand, B-rep systems have their relative advantages. One of them becomes evident when unusual shapes are encountered that would not be included within the available repertoire of the CSG systems.

Another point of comparison between the two approaches is the difference in the way the model is stored in the data base for the two systems. The CSG approach stores the model by a combination of data and logical procedures (the Boolean model). This generally requires less storage but more computation to reproduce the model and its image. In contrast, the B-rep system stores and explicit definition of the model boundaries. This requires more storage space but does not necessitate nearly the same computation effort to reconstruct the image. A related benefit of the B-rep system is that it is relatively simple to convert back and forth between a boundary representation and a corresponding wireframe model.

Windowing and Clipping

An user describes the picture that is to be displayed in the World Coordinate System (WCS), which is a right-handed coordinate system. It may sometimes be desirable to select a part of the object or a drawing for display, in order to obtain sufficient detail of

the object on the display device. For displaying one has to convert the world coordinate system into the Physical Device Coordinate System (PDCS) that corresponds to the device or workstation where the image of the picture be displayed. This transformation is called viewing transformation. Different devices have different physical coordinate systems, so the same application program may have to be modified before it can be seen on different devices. To remove this inconvenience, the Normalized Device Coordinate System (NDCS) is introduced. This coordinate system is a right-handed coordinate system in which the display area of the virtual display device corresponds to the unit (1 × 1) square whose lower left hand corner is at the origin of the coordinate system. A rectangular area which covers a finite region of the picture in WCS is called a **window**. This window is then mapped onto a specific location on the screen of the device. This subregion of the screen is called a **viewport**. The window in Fig. 14.29 (*a*) is mapped onto a viewport shown in Fig. 14.29 (*b*). If we move the viewport, the viewport that displayed picture



(*a*) Window (*b*) Viewport

Fig. 14.29

moves with it. Increasing the size of the viewport, a bigger section of the object can be displayed. The method for selecting and enlarging portion of a drawing is called windowing.

Commands to set the window and viewport areas from an application program can be defined as

set_window (xw_min, xw_max, yw_min, yw_max)
set_viewport (xv_min, xv_max, yv_min, yv_max)

Parameters in each function are used to define the coordinate limits of the rectangular areas. The limits of the window are specified in world coordinates. Normalized device coordinates are most often used for the viewport specification, although device coordinates may be used if there is only one output device in the system.

When we map a window area onto a viewport, only a portion of the picture is displayed. Clipping is the procedure of eliminating all the parts of a defined picture lying outside the window. Clipping involves determining which lines or portions of lines in the picture lies outside the window. These lines or portions of lines are not passed on to the display device.

REVIEW QUESTIONS SET

1. What is graphics? What is interactive computer graphics? What are its advantages?
2. Describe the outline of a computer graphics system using block diagram.
3. Explain the following methods of image creation on CRT:
   (*a*) Raster generation method,
   (*b*) Vector generation method.

4. What is meant by transformation? What are different types of transformation?
5. Explain the two-dimensional transformation. Also, give an example of Translation, Rotation, Scaling.
6. Write short notes on the following:
   (*a*) Parallel projections
   (*b*) Perspective projections
   (*c*) Depth-Cues.
7. What is modelling? Discuss wireframe, geometric and solid modelling.
8. What is meant by solid modelling?
9. Write shost notes on the following:
   (*a*) Clipping
   (*b*) Windowing
   (*c*) Frame buffer.
10. Find the images of the points A (2, 1), B (5, 1) and C (4, 5) under the lines transformation with matrix

$$\begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix}$$

[Ans. A' (3, 1), B' (9, –2), C' (5, 5)]

11. Explain DDA. Discuss symmetrical DDA.
12. Explain Bresenham's line drawing algorithm. Use the algorithm to find the distinct coordinates generated of the straight line between the end points (0, 0) and (4, 3).
    [Ans. (0, 0), (1, 1), (4, 3)]
13. Find the distinct coordinates generated of the straight line between the end points (0, 0) and (5, 5) using simple DDA algorithm. **[Ans. (0, 0), (1, 1), (2, 2,) (3, 3)]**
14. Write a FORTRAN program to output the coordinates of all points required to draw a straight line between two given points $A(x_1, y_1)$ and B (x_2, y_2). **(AMIE, W '97)**
15. What is circle generating algorithms? Explain the same in terms of cartesian and polar coordinate systems with neat diagram.
16. Explain mathematically the Bresenham's circle generating algorithm with illustrations.

15

VIRTUAL OFFICE

15.0. INTRODUCTION

Automation began in the factory and spread to the office in the form of office automation (OA). OA was intended to assist secretarial and clerical workers, but its ability to facilitate both formal and informal communications with people both inside and outside the organisation attracted managers and professionals as users. All these office workers use OA to increase their productivity.

Improvements in data communication, in recent past, have made it possible for some workers to perform some or all of their work anywhere. More and more employees are working at home and communicating with their offices by means of such electronic communications as electronic mail (E-mail), voice mail, facsimile transmission (FAX), multimedia, computer conferencing, and video conferencing. When an organisation performs its office work in this manner, the workplace is known as the **virtual office**.

The technology that makes the virtual office possible is the office automation. There are eleven OA applications. Some require the use of the computer, some do not. However, all can provide problem-solving information to employees at all levels, including managers at all levels. It is possible to tailor OA, through systems theory, to the communication needs of each manager. A particular manager's OA mix is influenced by the type of organisation, the manager's personal preference, and the available OA resources. The electronic office of today looks dramatically different from the mechanised office of just a few years ago, when typewriters, mechanical devices, and the postal system were the primary means of communication.

15.1. OFFICE AUTOMATION

The origin of office automation can be traced back to the early 1960s, when IBM coined the term *word processing* to express the concept that office activity is centered on the processing of *words*. The first tangible evidence of this new concept came in 1964, when IBM placed on the market a machine called the MT/ST. The letters stood for Magnetic Tape/Selectric Typewriter. As a typist typed a letter, the information was stored on tape. The letter could then be typed over and over from the tape. The typist had only to type the name and address of the person to receive each letter. The letters looked like originals.

Since the debut of word processing, other technologies have been applied to office work, and together they are known as *office automation*. **Office automation** includes all formal and informal electronic systems primarily concerned with the communication of information to and from persons both inside and outside the organisation.

Formal and informal electronic systems. Some OA systems are formal in that they are planned and perhaps are documented by a written procedure. These formal systems are implemented on a organisation-wide basis to meet its requirements in much the same manner as an MIS. Most OA systems, however, are neither planned nor described in writing. These informal OA systems are implemented as they are needed by individuals to meet their own needs.

Communication of information. The keyword that characterises OA is *communication*. OA is intended to facilitate all types of communication – both oral and written.

Persons both inside and outside the organisation. Today's OA systems facilitate communications not only among people inside the organisation but also between those people and others involved in the organisation's environment.

15.1.1. OA Users

OA is used by all the people who work in offices. Essentially there are four types of users – managers, professionals, personal secretaries and clerical employees. Managers and professionals are collectively known as *knowledge workers*, since their main contribution to their activities is their knowledge. Personal secretaries and clerical employees support the knowledge workers.

15.2. VIRTUAL OFFICE

The capability of OA to link people electronically has opened up new vistas in the performance of office work. The greatest challenge perhaps will be the cultural shock and the lack of control that managers may feel when they face quiet corridors and invisible employees.

Virtual office is a group of employers and employees who are not bound by a physical work environment, but instead use technology to come together temporarily and communicate to achieve a shared business goal. Thus, businesses need not invest in office space, furniture, and other amenities.

Impact of the virtual office. The virtual office began to emerge during the 1970s as low-priced microcomputers and data communications equipment made it possible for individuals to work at home. At the time, the term *teleprocessing* was used to describe data communications, and later the term *telecommuting* was introduced because it described how employees could electronically 'commute' to work. Now, mobile computing technologies are in commonplace. Mobile computing allows workers at remote locations to dial up the office, retrieve messages and urgent documents via touch-tone phones or voice recognition systems, and give directions to their co-workers in the organisation as easily as if they were in the same building. Clearly, the virtual office will create a different kind of knowledge worker and pose new challenges for management.

15.2.1. Advantages of the Virtual Office

The virtual office overcomes the physical constraints of the workplace with electronics and thereby makes possible following real advantages:

Reduced facility cost. The organisation need not to have a large office capacity, as some employees are working elsewhere, thus making possible reduced cost on office expenses.

Reduced equipment cost. Instead of providing office equipment to each employee, the telecommuters can share equipment in much the same way that participants in a LAN share its resources.

Reduced work stoppages. At times it is impossible for employees to travel to the workplace owing to storms, floods, hurricanes, etc. In a virtual office, much of the work can continue.

Formal communication network. Since the telecommuters must be kept informed about and provided with specific instructions, more attention is given to the communication network than is ordinarily the case.

Social contribution. The virtual office makes it possible for the organisation to employ persons who would not otherwise have an opportunity to work. The virtual office therefore provides the organisation an opportunity to express its social conscience.

15.2.2. Disadvantages of the Virtual Office

Following are the disadvantages of the virtual office:

Sense of not belonging. When employees do not come in direct daily contact with their co-workers, they lose the feeling of being an important part of an organisation.

Fear of job loss. Since the employees' work is done independently of the operations of the organisation, the employees can easily get the idea that they are expendable. They can conclude that anybody with a computer and a modem can do the work and that they may become the victims of an 'electronic layoff'.

Low morale. Low morale of employees may exist due to a number of factors. One is the absence of positive feedback that comes from face-to-face interaction with superiors and peers. Another is the fact that salaries paid to telecommuters tend to be lower than those paid to workers at fixed office locations.

Family tension. When tensions exist at home, the telecommuter is never able to escape for a few hours. Tension can also be increased because spouses may view the job as a working arrangement being used to avoid family responsibilities.

Since these are the disadvantages to the employee, the organisation must make a special effort to ensure that the employees do not suffer.

15.2.3. A Recommended Virtual Office Strategy

Linda T. Risse, a management consultant with Synergy Planning Incorporated, has identified several policies that the organisation should follow for maximum benefit from the virtual office. Following list is the basis of her recommendations:

- *Provide computer resources.* When employees cannot afford to furnish their own hardware and software, the organisation should make them available.
- *Provide access to information sources.* When research is a part of the telecommuter's responsibilities, the organisation should make available such required information sources as commercial databases and the Internet.
- *Provide non-computer supplies.* Although the virtual office is built on electronic data, employees should be given non-computer items that they need to do their jobs — calculators, staplers, overnight mailing envelopes, phone lists, procedure manuals, and so on.
- *Forwarding phone calls.* Someone at the fixed office should have the responsibility for forwarding phone calls to the telecommuters. Also, the organisation should make use of a voice messaging system that enables telecommuters to access it from their homes.
- *Use conference calls.* A conference call enables more than two persons to participate in a telephone conversation at the same time. Management should schedule conference calls at regular intervals to give telecommuters an opportunity to engage in live, two-way interaction.
- *Schedule regular meetings.* The organisation should schedule meetings that everyone is expected to attend. Such meetings are intended to foster a sense of community among the employees and should be scheduled as frequently as necessary to achieve that result.
- *Follow a work routine.* Telecommuters should commit to work certain hours a day, ideally in a room away from distractions. Friends and relatives should understand that the telecommuter is working even though he/she is at home.

The final strategy is one that the employee must enforce. The others are the responsibility of the organisation. The virtual office, therefore, demands cooperation by both the

organisation and the employees if it is to succeed. In fact, the virtual office demands more dedication from employees than does the fixed office. The virtual office is not for everyone. Unless the employee can discipline himself/herself to do the work without supervision and motivation, the work will likely to suffer.

The concept of 'virtual office' captures the way that computers and data communications networks are changing office work. The organisation of the future will be one that supports both virtual and fixed offices. Both types provide the setting for the office automation applications.

15.3. OFFICE AUTOMATION MODEL

A model of an OA system is shown in Fig. 15.1. Information is collected from the physical system of the organisation and entered in the database. For example, a marketing manager in the field uses a laptop computer to transmit a daily report. Information also enters the database from the organisation's environment. An example is business intelligence that is purchased from a supplier in the form of compact disks.

The information serves as input to such computer-based OA applications as word processing, electronic mail, and computer conferencing. The computer-based applications enable problem-solvers to communicate with each other using their computers.

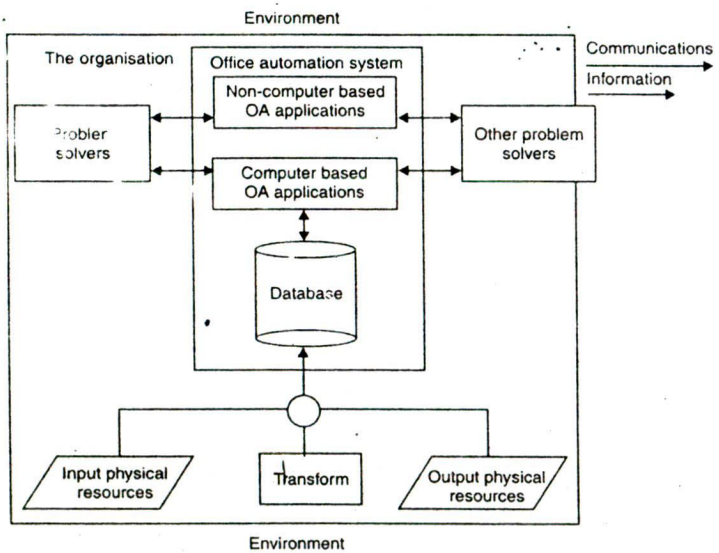


Fig. 15.1. An office automation model

The model also reflects the use of such non-computer based OA applications as video and audio conferencing.

15.4. TYPES OF OFFICE AUTOMATION SYSTEMS

Office automation systems can be broadly divided into two categories. Electronic publishing and processing systems, and electronic meeting systems. Figure 15.2 classifies OA systems into electronic publishing and processing systems, which facilitate written and voice

communication, and electronic meeting systems, which focus primarily on communications within groups.

15.4.1. Electronic Publishing and Processing Systems

Electronic publishing and processing systems include word processing, desktop publishing, document management systems (multimedia and imaging), and communication systems (E-mail, fax, and voice mail).

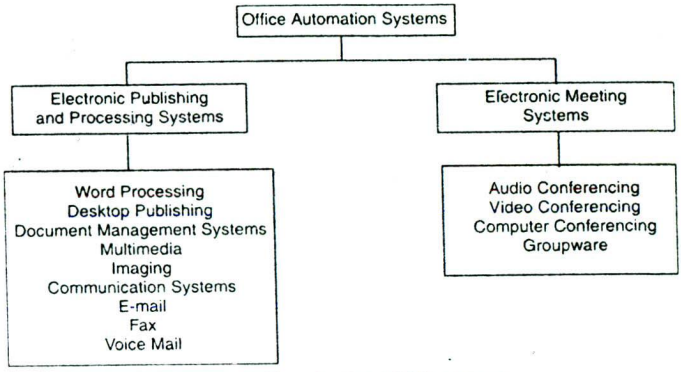


Fig. 15.2. Classification of OA systems

Word Processing

Word processing is the use of a computer to automatically perform many of the tasks necessary to prepare typed or printed documents. Figure 15.3 shows the primary units of a word processing system.

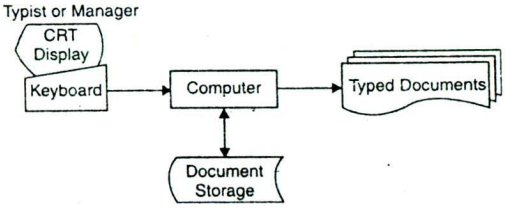


Fig. 15.3. A word processing system

Word processing contributes to problem solving by enabling the manager to prepare more effective written communication. The manager realises a return on his/her investment when the persons with whom the manager communicates use word processing in preparing memos, letters, and reports that are directed to the manager.

Desktop Publishing

Desktop publishing (DTP) is the use of a computer to prepare output that is very close in quality to that produced by a typesetter. A DTP system consists of the microcomputer configuration as shown in Fig. 15.3. The configuration includes a high-resolution CRT screen and a laser printer, and it is driven by DTP software. The software permits the selection of type fonts and sizes, hyphenation and right-margin justification, addition of horizontal and vertical lines, and layout of pages (including graphics).

DTP applications fall into *three* areas:

- *Administrative applications*. These include documents intended for such internal use as correspondence, reports, and newsletters.
- *Technical applications*. These include such training materials as slides overhead projectors and transparencies, and manuals.
- *Corporate graphics*. These include advertisements, brochures, and other documents intended for use outside the organisation.

Document Management Systems

Document management systems are computer-based tools that provide users to access text and graphics, regardless of where they are located and in what form the data are stored. The retrieved documents can be displayed in a wide variety of formats, edited, distributed, and integrated using other communication systems, such as word processing and DTP. Such systems allow knowledge workers to better control, coordinate, and manage the myriad electronic documents used in decision making. For example, Encyclopaedia Britannica, in Chicago, uses document management systems to allow users to create new products by mixing and matching various documents, thus promoting the idea of 'reusable text'. Two popular types of document management systems are *multimedia* and *imaging*.

Multimedia

Multimedia encompasses a group of computer-based technologies that integrate different types of media, such as text, graphics, animation, audio and video, to generate information. Multimedia is made possible by integrating audio and video capabilities into personal computers. Besides the usual CPU and peripherals, for a computer to have multimedia capabilities it should also have a CD-ROM drive, stereo speakers, a microphone for voice input, a sound card, a video card, and a video compression card.

Multimedia is very useful in corporate training and business presentations. Further, it is found that a multimedia system sometimes costs less than a single traditional training programme. Besides, multimedia is interactive and provides consistency in the content and the delivery of training material. Another advantage of using multimedia for training is that the material can be accessed by trainees at their convenience and can be viewed at any desired pace.

Imaging

Imaging digitizes documents, that is, it converts paper, microfilm, mircofiche, and electronic documents into digital form so that the original documents can be discarded. Thus, instead of tacking up space in cabinets and storehouses, all the necessary information is stored in digital form. Once digitised, documents can be readily accessed, regardless of their physical location, and can be edited, printed, faxed, or viewed on a computer screen.

An imaging system consists of scanners that digitise the images, workstations or PCs for processing the images, imaging software, and high-capacity storage devices, such as optical disks. For example, one kind of imaging software that is essential for most applications is the index server, which keeps an index of all documents in the system for easy retrieval. Therefore, when a document is digitised, it is immediately indexed according to its name, date, and any other relevant criteria.

Imaging systems help considerably in decreasing the storage costs. It also increases employees productivity because it allows instantaneous access to documents. It also allows more than one person to view or analyse a document at the same time. Thus, imaging changes the work flow from sequential to parallel, increasing productivity. In less than 10 sec, the

system can retrieve up to 18 images that match a given criterion and even provide information as to where the originals are filed. For example, the Income-tax Department may use digital imaging to scan income-tax forms, digitise them, and record taxpayer data. An OCR reads the taxpayer data and converts any handwritten numbers on the form into computer data for further processing. The imaging system is expected to result in faster tax refunds.

Communication Systems

Communication systems include E-mail, fax, and voice mail.

Electronic mail (E-mail)

E-mail is the use of a computer network that allows users to send, store, and receive messages using terminals and storage devices. Figure 15.4 shows such a system configuration. A user sends an electronic message over a network; the message is stored in the electronic mailbox of the receiver. The electronic mailbox is usually a file on a server; the messages in it can be retrieved when the recipient is ready to receive them. Users can also edit, sort, save, and classify messages and forward them to other individuals on the network. Some E-mail systems have multimedia capabilities allowing E-mail users to send not only text, but also voice and still pictures.

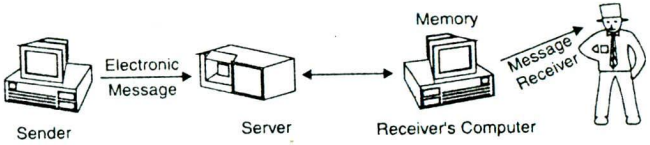


Fig. 15.4 An electronic mail system configuration

There are many kinds of E-mail software, depending on the characteristics of each type used, mail can be sent between different computers or may be restricted to users on one computer. E-mail system allows organisations to be responsive to customer needs. It provides instantaneous access to and dissemination of information, thereby eliminating the time-lag involved in using the postal service. E-mail messages do not get lost or reach the wrong party. Inspite of many benefits, E-mail should be used carefully. Finally, E-mail systems must be carefully selected, since a poor package may not only deter users from using E-mail but can also result in poor communications.

In an organisation, if it is required that everyone must have an access with a terminal to read the message, it is placed on an *electronic bulletin board*. In case everyone is to receive a confirmation when the recipient retrieves the message, it can be sent as *registered mail*. If it is required that the message sent to the recipient should not be routed through others in the network, the same can be sent as *private mail*.

Facsimile Transmission (FAX)

Facsimile transmission is another type of electronic publishing and processing system. Fax technology uses telephones, modems, and scanners to transmit text and graphics to individuals and organisations all over the world having access to telephones.



Fig. 15.5. Facsimile transmission

Figure 15.5 illustrates the use of a fax machine. A scanner in the fax machine scans the

document at one end and a built-in modem sends it; in the fax machine at the other end, a built-in modem receives the message, a scanner scans the document, and a printer prints it.

Fax machines can send the same document to multiple users. They can also be integrated with applications such as word processors, so that faxes can be edited without being rekeyed into the computer. Fax machines can be programmed to send faxes when telephone rates are lowest. An interesting development is the fax modem, which allows users to send or receive faxes using their PCs. Another new feature is fax-on-demand (FOD), which uses touch-tone phones and prerecorded instructions to provide users with customised information by treating the keypad of the touch-tone phone as a data terminal and the fax machine as a printer. FOD technology can reduce operating cost considerably and customers can receive information without human intervention.

Another new development is LAN-based fax systems, also called enterprise-wide fax systems. Such systems help to centralise the receiving and sending of faxes and thereby better monitor and control the cost of fax traffic.

Voice Mail

Voice mail facilitates oral communication. Voice mail system configuration is shown in Fig. 15.6. In this system, the sender dictates a message over phone. A special device, called a *codec*, converts the analog signal of the sender's voice into a digitised message. The message is transmitted over the network and stored in a server at the receiver's end. A blinking light on the receiver's phone indicates that he/she has a voice message. When the receiver chooses, the digitised message is retrieved from the server, reconverted into analog form, using a codec at the receiver's end, and the receiver receives it over the phone. In some systems, users need a password to access voice messages, making voice mail a secure medium for confidential messages.

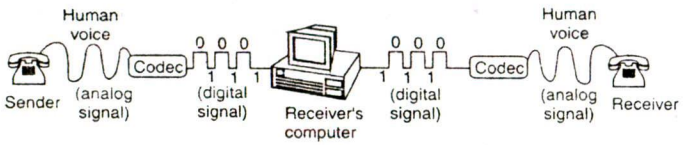


Fig 15.6 Voice mail system configuration

The main advantage of voice mail over E-mail is that the message need not to type. Voice mail also makes it easy to include people in the organisation's environment in a communications network.

15.4.2. Electronic Meeting Systems

Electronic meeting systems consist of tools and techniques that allow a group of people, separated by time and distance, to exchange ideas using audio, video and other electronic media. This system reduces the operating costs considerably and increase productivity as decision makers do not have to travel to attend face-to-face meetings. One type of electronic meeting system is the *group decision support systems (GDSS)*.

Audio Conferencing

Audio conferencing is the use of voice communications equipment to establish an audio link between geographically dispersed persons for conducting a conference. It allows two or more persons to participate in a telephone conversation. Telephone conversations are usually one-to-one; this can be a limitation when more than one person has to participate in the conversation. Audio conferencing is ideal for managing small to medium-sized projects. It is most useful with group of 5 to 15 persons.

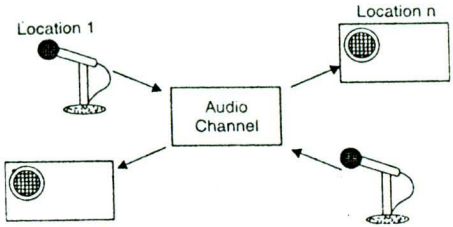


Fig. 15.7 An audio conferencing system.

An audio conferencing system is illustrated in Fig. 15.7

Audio conferencing can be made more efficient if following rules are followed:

1. The person who organises the conference should serve as a *moderator*, making certain that all participants have an opportunity to speak and that the objectives of the conference are achieved.
2. The number of participants should be kept to a manageable size.
3. A copy of the conference agenda should be made available to the participants in advance, perhaps using fax.
4. When participants speak, they should identify themselves.
5. A taped record of the conference should be kept.
6. A hard copy record should be prepared from the taped record, and distributed to all participants.

Video Conferencing

Video conferencing is another type of electronic meeting system that uses telephones, TV monitors, computers, and networks to link geographically separated decision makers to hear and see each other. In this system, a computer digitises sound and video images, then converts them to analog signals and transmits them over the telephone lines to the receiver's computer, which reconverts the analog signals to digital signals. These are then translated into audio and video messages, and presented on the television monitor and sound system.

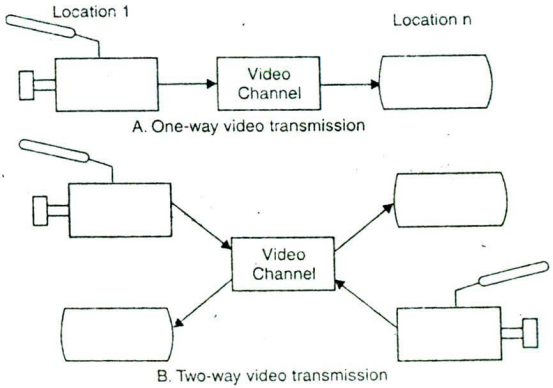


Fig. 15.8. Video conferencing

Three basic video conferencing configurations are possible, based on the signal capabilities:

- *One-way video and one-way audio.* Video and audio signals are sent from a single transmitting site to one or more receiving sites. A good application is use by a project leader to disseminate information to team members at remote sites.
- *One-way video and two-way audio.* The two-way audio capability allows people at the receiving sites to talk to people at the transmitting site while everyone views the same video images. Figure 15.8A illustrates one-way video transmission.
- *Two-way video and two-way audio.* The video and audio communications between all sites are two-way. This setup is the most expensive of the electronically aided conferencing approaches. Figure 15.8B illustrates two-way video transmission.

The organisations regard video conferencing as a way to get more managers involved in decision-making who otherwise would not participate because of their geographic location. A limitation of video conferencing is that it cannot be used for all types of communication. When a meeting is necessary to discuss an especially complex problem, it is best for the members to assemble in person.

Computer Conferencing

Computer conferencing is the use of a networked computer to allow members of a *problem-solving team* to exchange information concerning the *problem* that is being solved. A computer conference group can consist of a large number of participants. It can be used within a single geographic site. A person can use computer conferencing to communicate with someone in the next office.

Teleconferencing includes all three forms of electronically aided conferencing – audio, video, and computer.

Groupware

Groupware is a broad term for communication tools that foster electronic communication and collaboration in groups. Some examples of groupware are:

- Electronic calendars and schedulers
- E-mail
- Voice mail
- Fax
- Electronic bulletin board systems
- Video conferencing
- Project management
- Group decision support systems.

Figure 15.9 shows some electronic tools that are classified as groupware. It allows groups to work together; exchange ideas, information and knowledge. Linking group members who are geographically separated requires networks, which are among the essential components of groupware. It eliminates the barriers of time and space and allows different group members to work on the same document at the same time. Electronic bulletin boards allow members to post their ideas and elicit responses from other group members. Primary benefits are increased responsiveness to market forces and significant improvement in the quality of business processes such as product development, account management and customer service. In an organisation, various applications can be integrated to facilitate the free flow of information among them. Security issues must be carefully addressed in the

early stages of groupware development so that valuable information will not be lost or fall in the wrong hands. Groupware also reduces bureaucracy and requires people to work in teams. In some organisations, this may require fostering a new mindset. 'Command, control and communication are no longer the driving forces in organisations. Coordination, cooperation and collaboration are what's in store for the future'.

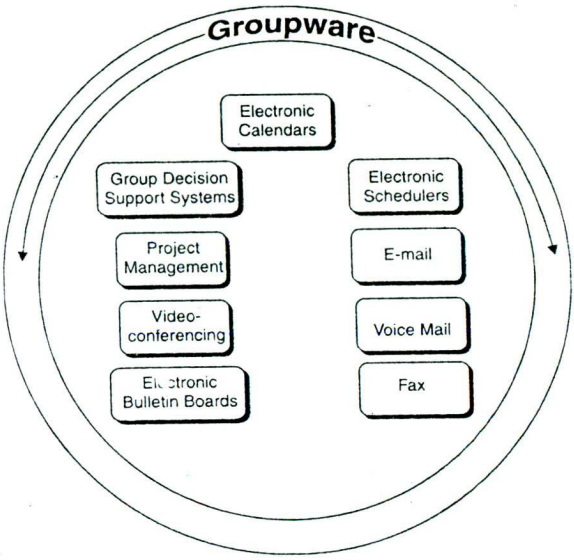


Fig. 15.9. Groupware classification of tools

One of the most popular groupware packages is *Notes*, from Lotus. Notes is a distributed database that allows users to share a wide variety of documents in different forms and from different sources such as text, spreadsheet, data, graphics, and images. With some hardware and software enhancements, this application can even support sound and video. It is menu-driven and icon-based so that even people with little or no programming background can use it effectively.

15.5. INTERNET

15.5.1. History

The advent of the Internet was somewhat accidental. The Internet was originally the outcome of a strategic study during the cold war years, into how US authorities could communicate and command centres function after a nuclear strike. The answer was that a special command-and-control network was needed. According to conventional wisdom, such a network would be unreliable because of its lack of a central command system; nevertheless, it is capable of functioning in a decentralised fashion. It would be designed from the start to transcend its own unreliability. All the nodes in the network would be equal in status to all other nodes, each node with its own authority to originate, receive and pass on messages. It would also be possible to add or remove nodes without disturbing the overall functioning of the network. Such a network would serve the primary purpose of continuing to run as a network despite local malfunctions.

The first such network was created in 1969 for the US Department of Defense. Called the ARPAnet (ARPA stands for Advanced Research Project Agency), it was designed to be decentralised so that even if part of the network malfunctioned, the remainder would keep functioning. However, it was soon revealed that ARPAnet's users had 'warped' the computer-sharing network into a dedicated, high-speed, federally subsidised electronic postoffice. The main traffic on ARPAnet was not long-distance computing; instead, it was news and personal messages. Researchers were using ARPAnet to collaborate on projects, to trade notes on work, and eventually, to barter gossip and rumours. People had their personal addresses for electronic mail on the ARPAnet computers. Their enthusiasm for using ARPAnet as an interpersonal communication tool greatly overshadowed the system's original objective of being a long-distance computational tool. It was not long before the invention of the mailing-list, an ARPAnet broadcasting technique in which an identical message could be sent automatically to a large number of network subscribers. Throughout the 1970s, ARPA's network grew. Its decentralised structure made expansion easy. Unlike standard corporate computer networks, the ARPA network could accommodate many different kinds of machine.

In 1984, the National Science Foundation (NSF) of the US government recognised the great potential of networking for scientific research and built its own network, known as NSFnet. The new NSFnet set a blistering pace for the Internet's technical advancement. Other US government agencies such as NASA soon joined the race. There were also other major networks, including the Australian Academic and Research Network (AARNet), the NASA Science Internet (NSI) and the Swiss Academic and Research Network (SWITCH). As the networks grew, different Internet 'domains' were generated and standardised. They are **gov** for government institutions, **mil** for military institutions, **edu** for educational institutions, **com** for commercial institutions, **net** for network operations and services, and **org** for non-profit organisations. This list is due to be expanded.

15.5.2. What is the Internet?

The Internet, and the name by which it is sometimes known, the Information Superhighway, have become popular buzz words. Whether one is already an avid Internet user or still a novice, the growing relevance of the Internet to work, study, research, social life and recreational activities is irresistible. In particular, the Internet is becoming more meaningful for engineers, scientists and technologists.

The Internet is a computer network interconnecting numerous computer hardware or local computer networks. These networks are linked by a set of common technical 'protocol' that enable users of any one of the networks to communicate with or use the services located on other networks. A protocol can be seen as a formal description of message formats and the rules that two communicating computers must follow to exchange data. Protocols can describe the low-level details of machine-to-machine interfaces (i.e., the order in which bits and bytes are sent across a wire) or high-level exchanges between allocation programs (i.e., the way in which two programs transfer a file across the Internet). These protocols are referred to as the Transmission Control Protocol/Internet Protocol, or TCP/IP protocol, suite. The number of these computer networks for the Internet is growing astronomically. A Internet domain survey shows that there are 6.6 million computers connected to the Internet. The rapid growth of the Internet in the next decade or two is widely anticipated.

The TCP protocol for data transmission is a group of over 100 data-communications protocols currently being used to link computers and data-communications equipment into computer networks. Originally designed to interconnect hosts on ARPAnet, an early computer network located at the US Department of Defense, TCP/IP has now been adopted

by a large number of universities, research institutions, government agencies and corporations as their computer communication network protocol for the Internet.

The Inernet would not have developed without a technology called packet-switching. The technology preceding packet-switching was capable of transmitting large quantities of data from one computer to another only in a single continuous stream. This is like a telephone call in which only one line can be connected to one telephone at a time. Therefore, it was not possible for different network segments to 'share' a common transmission medium. Packet-switching technology breaks the data into small fragments — each fragment is 'packed', marked with its source and destination, and pushed into the transmission line. This allows packets from different sources to pour into the transmission and arrive at each destination. After reaching its destination, each packet is unpacked and re-assembled with other packets to recover the original data.

Though information provided by the Internet can be of many kinds, the Internet only transmits data. Of course, the data can be of texts, pictures, graphs, publications, sounds, video and more. In order for your computer to interpret correctly different types of data, one needs not only the right hardware such as a modem, sound card, etc., but also the right software. There are plenty of books that address the hardware side of linking to the Internet. For people engaged in engineering study and research, the computer hardware is often readily available. What one requires now are the skills to exploit the Internet to his own maximum benefit.

The Internet provides various services for interacting with computers, and data are transferred by different protocols. Among the most commonly used protocols between computers on the Internet are:

- mailing and receiving electronic text to each other (E-mail)
- retrieving and sending documents or groups of documents to and from each other (FTP or file transfer protocol)
- browsing each other's collections of documents (Gopher, FTP)
- requesting a search for a document on a remote system (WAIS or Wide Area Information Server)
- exchanging messages on electronic message boards (Usenet newsgroups).

There are also other more complex interactions that link computers, but many are actually combinations of these basic services.

The Internet is a worldwide communication medium. It is an information pool from where people all over the world exchange, retrieve, disseminate and store information. Never before have people had such a boundless and dynamic environment in which to do so. The information floating on the Internet can be of any kind. Given time and effort, one will be impressed by what the Internet can offer.

15.6. WORLD WIDE WEB

The World Wide Web, also known as the Web, the WWW or W3, was initially developed by the European Laboratory for Particle Physics, otherwise known as CERN, in order to build a distributed hypermedia system. It is an interconnected assembly of resources available on the Internet that can be displayed by means of a type of software program known as a 'Web browser' or 'Web navigator'. This system aims to provide global access to a universe of documents, most of them in the form of 'hypertext'. Hypertext is akin to regular text in many aspects — it can be stored, read, searched or edited. However, unlike regular text, it contains pointers to other text. In this way, it generates 'hyperlinks' which form a web and continue to provide information and documents on a particular topic.

To access the World Wide Web, one needs a Web browser to read documents and retrieve them onto his computer from other sources. Information providers set up hypermedia 'servers' from which browsers access documents. The browsers can, in addition, access files by FTP, NNTP (the Internet news protocol), Gopher and a range of other protocols. Better still, if the server has search capabilities, the browsers permit a person to search documents and databases.

There are over 30 different Web browsers currently in use, but only a few have gained widespread acceptance. Netscape Navigator from Netscape Communications, Internet Explorer from Microsoft and Modaic are perhaps still the most popular browsers on the Web, and are highly recommended. They are compatible with PC, Macintosh and Unix platforms.

Both the terms 'World Wide Web' and 'Internet' are frequently used, but it is important to understand that the terms are not synonymous. The Web refers to a body of information — an abstract space of knowledge — while the Internet refers to the physical side of the global network, a giant mass of cables and computers. Viewed as a hierarchy, the Web is a subset of the Internet.

Hyperlinks connect documents created by HTML, enabling a person to move to another part of the document or to a different document. The Uniform Resource Locator (URL) is the address of a resource on the Web. Included in the URL are the tool to access the resource, the address of the computer where the resource is located, and the optical path name of the resource on the computer where the resource resides. HTTP (Hyper Text Transfer Protocol) is the protocol of the World Wide Web. Java is the programming language used on the Web to add animation, sound, and interactive programming to a resource. Research on the Web can be done by searching or browsing. Searching involves looking for key terms and identifying Web locations that use those terms. Browsing involves looking around for interesting items. Search engines are categorised as subject directories on Web databases. Web databases, such as Alta Vista and Infoseek Guide, are built by software 'robots' that roam the Web looking for key terms and placing them in databases.

Web pages are documents, with hyperlinks, that are stored on the Web and retrieved by browsers. A home page is the page first encountered when a Web browser finds a Web site. The best computer to use for getting on the Internet and the World Wide Web is capable of using a graphics interface, a fast modem, a telephone line, and either an Internet Service Provider (ISP) or an online service.

In essence, the World Wide Web is the document technology on the Internet.

15.7. ELECTRONIC COMMERCE

The electronic commerce, popularly known as E-commerce, is the new buzzword, for doing business electronically over the Internet. It can be defined as the use of the computer to facilitate all the operations of an organisation. Many of the operations are internal — they are performed within the organisation by the functional areas of finance, human resources, information services, manufacturing, and marketing. Other operations involve the organisation's interfaces with *eight* environmental elements – customers, suppliers, stockholders and owners, labour unions, government, financial community, global community, and competitors. E-commerce is not just buying and selling things on the computer, as is commonly believed. It is a paradigm shift and calls for a huge change in the way it was thought. The *first* function is the creation of virtual shops and shopping malls which exhibits the products and services they want to sell. To facilitate selling, there has to be a way of paying electronically, which then calls for security and identification systems. Electronic banks are required to be created to allow people to deposit, withdraw and pay

money just by pressing keys on a computer at their homes. Then online delivery system in case of information products or a physical delivery system in case of hard products has to be evolved.

15.7.1. The Virtual Front

E - Commerce can be divided into:

- E- tailing or 'virtual storefronts' on websites with on-line catalogues, sometimes gathered into a 'virtual mall'.
- Collecting and using demographic data through the Internet.
- *Electronic Data Interchange,* the business-to -business exchange of data.
- E-mail and fax as media to reach customers.
- Business-to-business buying and selling.
- The security of business transactions.

Bill Gates, Chairman of Microsoft, has given following *twelve* steps for organisations to turn digital:

1. Communicate through E-mail.
2. Sales data should be online.
3. Inculcate high level thinking.
4. Motivate employees.
5. Reduce paperwork.
6. Eliminate single task jobs.
7. Feedback.
8. Handle customer complaints immediately.
9. Redefine the boundaries.
10. Just-in-time delivery systems.
11. Eliminate middle-man.
12. Help customers.

15.7.2. Anticipated Benefits from E-Commerce

Organisations engage in E-commerce to achieve improvements are expected to result in three main benefits:

- Improved customer service
- Improved relationships with suppliers and financial community.
- Increased return on stockholder and owner investments.

15.7.3. E - Commerce Constraints

Three main constraints that organisations do not want to implement E- commerce are:

- High costs
- Security concerns
- Immature or unavailable software.

15.7.4. Pathway to E-Commerce

When the organisation decides to implememnt E-commerce, the strategic business plan embodies the commitment to use E-commerce to achieve competitive advantage. The organisation first collects business intelligence to understand the potential role played by each environmental element. Then there is need to establish an interorganisational

system (IOS) by means of EDI. The IO is achieved by following the system development life cycle or engaging in business process redesing. The result is a network-oriented system that employs direct connectivity, value-added networks, Internet, or some combination. Figure 15.10 shows the major steps for selection of E-commerce.

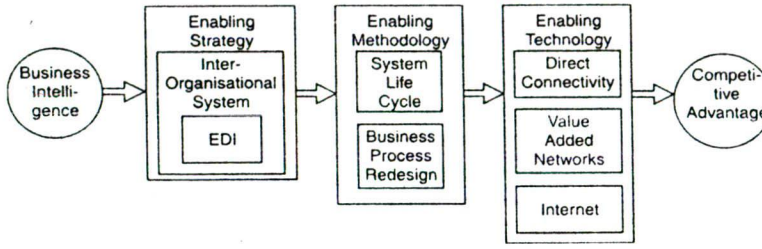


Fig. 15.10. Selection of E-commerce strategy, methodology and technology

15.7.5. Impact of Internet on Business

E-commerce is truly the future of business. While it is difficult to say that people will stop going out to shop, the Internet does promise to offer a new channel of trade and commerce. It will definitely change the business processes. The Internet is seen as the beginning of *a National Information Infrastructure* (NII). In case each country has an NII, it will affect commerce in different ways. In developed countries like Canada, Japan, US, Germany, which make quick use of innovation in technology, the effect can be swift and dramatic. Within a country, the effect can vary from one industry to another.

REVIEW QUESTIONS SET

1. What is office automation ? What are some tools that can be classified as office automation system?
2. What is virtual office and what are some reasons for its increasing popularity?
3. What are some managerial implications at the virtual office?
4. Outline some policies that an organisation should follow to derive maximum benefits from the virtual office.
5. Illustrate with a neat sketch an office automation model.
6. What specific tools can be categorised as electronic publishing and processing systems?
7. Describe a multimedia system and its components. What are some uses of multimedia?
8. What is E-mail? Explain with a neat sketch. What are some of its advantages and disadvantages?
9. Why are fax machines sometimes referred to as long-distance copiers? What is the difference, if any, between stand-alone fax and fax-on-demand?
10. Explain the difference between audio conferencing and video conferencing? Explain how they work.
11. What is an imaging system? What are some of its components?
12. Illustrate with sketch the voice mail working system and its advantages.
13. Define the term teleconferencing. Why teleconferencing is more useful than audio conferencing, video conferencing and computer conferencing.
14. What is groupware? How does it facilitate group decision making?
15. Outline the history of Internet.

16. What is the Internet? What are the most commonly used protocols between computers on the Internet?
17. What is world wide web. How is it useful in an organisation?
18. Outline the essential components of world wide web.
19. What is a browser? Is Intranet the same as Internet? Explain.
20. What is a location on WWW called? What name is used to describe the information that the location makes available to web users.
21. Outline the anticipated benefits and constraints of E-commerce.
22. What do you mean by E-commerce?
23. Mention the steps essential for organisations to turn digital.
24. Explain the strategic business plan for implementing the E-commerce with a neat sketch.
25. Is the Internet same as the world wide web? Explain.